

UNIVERSIDADE FEDERAL DO PARANÁ

ANDRÉA APARECIDA SOARES

MEMORIAL DE PROJETOS: DA ANÁLISE ÉTICA À ENGENHARIA DE SISTEMAS
DE INTELIGÊNCIA ARTIFICIAL CENTRADOS NO HUMANO

CURITIBA

2026

ANDRÉA APARECIDA SOARES

MEMORIAL DE PROJETOS: DA ANÁLISE ÉTICA À ENGENHARIA DE SISTEMAS
DE INTELIGÊNCIA ARTIFICIAL CENTRADOS NO HUMANO

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

Orientadora: Profa. Dra. Rafaela Mantovani Fontana

CURITIBA

2026



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

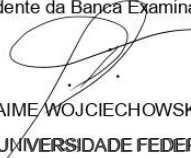
ATA Nº162

**ATA DE SESSÃO PÚBLICA DE DEFESA DE ESPECIALIZAÇÃO PARA A
OBTENÇÃO DO GRAU DE ESPECIALISTA EM INTELIGÊNCIA ARTIFICIAL
APLICADA**

No dia cinco de março de dois mil e vinte e seis às 17:30 horas, na sala Auditório A, Rua Dr. Alcides Vieira Arcoverde, 1225, Jardim das Américas, foram instaladas as atividades pertinentes ao rito de defesa de monografia da discente **ANDRÉA APARECIDA SOARES**, intitulada: **MEMORIAL DE PROJETOS: DA ANÁLISE ÉTICA À ENGENHARIA DE SISTEMAS DE INTELIGÊNCIA ARTIFICIAL CENTRADOS NO HUMANO**. A Banca Examinadora, designada pelo Colegiado do Programa de Pós-Graduação em Inteligência Artificial Aplicada da Universidade Federal do Paraná, foi constituída pelos seguintes Membros: **RAFAELA MANTOVANI FONTANA** (UNIVERSIDADE FEDERAL DO PARANÁ), **JAIME WOJCIECHOWSKI** (UNIVERSIDADE FEDERAL DO PARANÁ). A presidência iniciou os ritos definidos pelo Colegiado do Programa e, após exarados os pareceres dos membros do comitê examinador e da respectiva contra argumentação, ocorreu a leitura do parecer final da banca examinadora, que decidiu pela aprovação. Este resultado deverá ser homologado pelo Colegiado do programa, mediante o atendimento de todas as indicações e correções solicitadas pela banca dentro dos prazos regimentais definidos pelo programa. A outorga de título de especialista está condicionada ao atendimento de todos os requisitos e prazos determinados no regimento do Programa de Pós-Graduação. Nada mais havendo a tratar a presidência deu por encerrada a sessão, da qual eu, **RAFAELA MANTOVANI FONTANA**, lavrei a presente ata, que vai assinada por mim e pelos demais membros da Comissão Examinadora.

Curitiba, 05 de Março de 2026.


RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora


JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

AGRADECIMENTOS

À minha família, pelo amor incondicional que nutre minha alma e me dá coragem para continuar, mesmo quando o caminho exige reinvenção.

Aos amigos, colegas e mestres que cruzaram meu caminho e que fizeram parte da minha história, aos que ficaram e aos que partiram, pois cada um foi essencial para a minha evolução.

À vida, por demonstrar que o propósito é o antídoto para a estagnação.

Destino este trabalho à resiliência e ao compromisso de desenvolver projetos que transcendam a técnica e gerem valor real para a sociedade. Que este memorial simbolize o primeiro passo de uma jornada pautada pela inovação e pela aplicação da tecnologia como instrumento genuíno de transformação social.

*"Olhe bem para o presente
que você está construindo: ele deve
se parecer com o futuro que você
está sonhando."*

Alice Walker

RESUMO

Este memorial analisa a atual inflexão no debate sobre Inteligência Artificial (IA), que vem se deslocando de uma discussão predominantemente ética e abstrata para uma abordagem mais pragmática, centrada na engenharia de sistemas e na geração de valor organizacional. Apesar dos avanços técnicos em capacidades de raciocínio dos Grandes Modelos de Linguagem (LLMs), persiste um descompasso entre o potencial desses modelos e o valor efetivamente capturado pelas organizações, de modo que elevados investimentos não se traduzem, necessariamente, em retorno concreto. O texto discute ainda a chamada "ilusão de pensamento" dos LLMs: são ferramentas poderosas, mas que não substituem o julgamento humano em cenários complexos, reforçando a necessidade de supervisão qualificada, desenho cuidadoso de sistemas e governança adequada. No contexto brasileiro, observa-se uma contradição importante: embora a Estratégia Brasileira de Inteligência Artificial estabeleça diretrizes claras para o desenvolvimento ético e soberano da IA, pesquisas recentes indicam que o uso efetivo de IA em empresas e equipamentos culturais ainda é incipiente, concentrando-se sobretudo na digitalização básica de processos. Defende-se, assim, que o principal gargalo se encontra na engenharia de requisitos e na falta de práticas específicas para sistemas de IA centrados no humano, propondo-se a adoção de um Modelo de Maturidade em IA Centrada no Humano (Human-Centered AI Maturity Model – HCAI-MM), com ênfase em uma perspectiva de design organizacional que incorpore explicitamente a cognição humana no ciclo de processamento da informação. Conclui-se que o sucesso da IA no Brasil dependerá menos da disponibilidade de novos algoritmos e mais da capacidade de projetar sistemas sociotécnicos nos quais a tecnologia amplie, de fato, as capacidades humanas.

Palavras-chave: inteligência artificial; engenharia de sistemas; IA centrada no humano; ética da IA; estratégia organizacional.

ABSTRACT

This paper analyzes the current inflection point in the Artificial Intelligence (AI) debate, which has been shifting from a predominantly ethical and abstract discussion toward a more pragmatic approach focused on systems engineering and organizational value generation. Despite technical advances in the reasoning capabilities of Large Language Models (LLMs), a persistent gap remains between the potential of these models and the value effectively captured by organizations, such that substantial investments do not necessarily translate into concrete returns. The text further discusses the so-called "illusion of thinking" in LLMs: they are powerful tools, yet they do not replace human judgment in complex scenarios, reinforcing the need for qualified oversight, careful system design, and adequate governance. In the Brazilian context, an important contradiction emerges: although the Brazilian Artificial Intelligence Strategy establishes clear guidelines for the ethical and sovereign development of AI, recent research indicates that the effective use of AI in businesses and cultural institutions remains incipient, concentrated mainly in the basic digitalization of processes. It is argued that the main bottleneck lies in requirements engineering and the lack of specific practices for human-centered AI systems, with the adoption of a Human-Centered AI Maturity Model (HCAI-MM) being proposed, emphasizing an organizational design perspective that explicitly incorporates human cognition into the information processing cycle. It is concluded that the success of AI in Brazil will depend less on the availability of new algorithms and more on the capacity to design sociotechnical systems in which technology genuinely enhances human capabilities.

Keywords: artificial intelligence; systems engineering; human-centered AI; AI ethics; organizational strategy.

SUMÁRIO

1	PARECER TÉCNICO.....	9
	REFERÊNCIAS.....	14
APÊNDICE A -	INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL.....	15
APÊNDICE B -	LINGUAGEM DE PROGRAMAÇÃO APLICADA.....	21
APÊNDICE C -	LINGUAGEM R.....	39
APÊNDICE D -	ESTATÍSTICA APLICADA I.....	49
APÊNDICE E -	ESTATÍSTICA APLICADA II.....	64
APÊNDICE F -	ARQUITETURA DE DADOS.....	80
APÊNDICE G -	APRENDIZADO DE MÁQUINA.....	89
APÊNDICE H -	DEEP LEARNING.....	110
APÊNDICE I -	BIG DATA.....	133
APÊNDICE J -	VISÃO COMPUTACIONAL.....	135
APÊNDICE K -	ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	156
APÊNDICE L -	GESTÃO DE PROJETOS DE IA.....	168
APÊNDICE M -	FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	171
APÊNDICE N -	VISUALIZAÇÃO DE DADOS E STORYTELLING.....	191
APÊNDICE O -	TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	197

1 PARECER TÉCNICO

1.1 O DILEMA ENTRE O POTENCIAL TÉCNICO E O VALOR REAL

A discussão sobre Inteligência Artificial encontra-se em um ponto de virada. Ao longo dos últimos anos, o debate concentrou-se, em grande medida, em questões éticas, riscos e implicações sociais da IA. Em 2025, contudo, observa-se uma mudança importante em direção a uma abordagem mais pragmática, voltada para resultados concretos e produtividade organizacional.

O *Hype Cycle* mais recente do Gartner (2025) já sugere que grande parte das tecnologias de IA está saindo da fase de expectativas infladas e entrando em um estágio no qual se exige demonstração efetiva de valor. Não basta mais enfatizar apenas o caráter inovador ou "generativo" da tecnologia. Relatório do Boston Consulting Group (2024), por exemplo, formula de maneira direta a questão central deste momento: "*Where's the value in AI?*". A consultoria mostra que muitas organizações adotam soluções de IA sem que haja clareza sobre os casos de uso prioritários, os indicadores de desempenho ou a forma de integrar tais soluções aos processos existentes.

Nesse contexto, o presente parecer parte da premissa de que a resposta para esse dilema não está apenas no aprimoramento do código ou dos modelos, mas, sobretudo, na maneira como se projeta o sistema sociotécnico na totalidade. Há um descompasso evidente entre o que os modelos de IA podem realizar, em termos técnicos, e o que as organizações conseguem efetivamente absorver e incorporar às suas rotinas. A Deloitte (2025), em seu relatório sobre tendências tecnológicas, denomina esse fenômeno de "dilema orçamentário da IA": mesmo diante de orçamentos significativos destinados a projetos de IA, o retorno sobre o investimento (ROI) frequentemente se mostra limitado ou pouco mensurável.

Essa constatação sugere a necessidade de deslocar o foco sem abandonar a perspectiva ética para uma abordagem fortemente ancorada na engenharia de sistemas. Mais do que discutir abstrações sobre "a IA em geral", parece urgente compreender como projetar, de maneira sistemática, soluções que articulem tecnologia, processos organizacionais, governança e participação humana, de modo a transformar o potencial técnico em valor real.

1.2 A FRONTEIRA TÉCNICA: RACIOCÍNIO, NOÇÃO E COMPLEXIDADE

Do ponto de vista computacional, os avanços recentes são expressivos. Estudos como os de Gao *et al.* (2025) e Xie *et al.* (2025) apontam para melhorias na capacidade de raciocínio abstrato e de raciocínio encadeado em grandes modelos de linguagem. A literatura evidencia que, com técnicas específicas de *prompting* e treinamento, esses modelos passam a resolver tarefas que exigem certo nível de abstração e decomposição de problemas, o que indica aumento de sofisticação na forma como "manipulam" estruturas conceituais.

No entanto, esses avanços não eliminam limitações relevantes. Shojaei *et al.* (2025) chamam atenção para a chamada "ilusão de pensamento" (*illusion of thinking*). Os autores demonstram que, em problemas de alta complexidade, os modelos tendem a produzir respostas que aparentam coerência e profundidade, mas que, sob análise rigorosa, não sustentam o mesmo nível de consistência lógica ou factual. Em outras palavras, há um descompasso entre a fluência textual e a robustez do raciocínio subjacente.

Esse fenômeno reforça dois pontos centrais deste memorial. Primeiro, embora os *LLMs* configurem ferramentas cada vez mais poderosas, eles não substituem o julgamento humano em contextos complexos, ambíguos ou de alta responsabilidade decisória. Segundo a ausência de supervisão qualificada e de um desenho sistêmico adequado, pode levar organizações a confiar em saídas "convincentes, porém equivocadas", com riscos tanto operacionais quanto éticos.

Dessa forma, torna-se evidente que a tecnologia, isoladamente, não resolve os problemas organizacionais. A introdução de IA demanda uma engenharia centrada no humano, capaz de incorporar essas limitações ao projeto do sistema, definindo claramente quando, como e sob quais salvaguardas os modelos devem ser utilizados.

1.3 O CENÁRIO BRASILEIRO: ESTRATÉGIA, DADOS E O “GAP” DE ENGENHARIA

Ao transpor essa discussão para o contexto nacional, observa-se um quadro de contrastes. De um lado, o Brasil conta com uma diretriz clara por meio da Estratégia Brasileira de Inteligência Artificial 2020–2029 (Brasil, 2025), que estabelece objetivos em torno de soberania tecnológica, inovação, ética e desenvolvimento sustentável

mediado por IA. De outro lado, estudos empíricos indicam que a implementação prática dessas diretrizes ainda é limitada.

Pesquisas do Comitê Gestor da Internet no Brasil (CGI.br) ilustram bem esse cenário. A TIC Empresas 2024/2025 (CGI.br, 2025a) e a TIC Cultura 2024 (CGI.br, 2025b) mostram que houve avanços significativos em infraestrutura digital básica. Empresas e equipamentos culturais encontram-se majoritariamente conectados, com processos e acervos em acelerado processo de digitalização. Contudo, o uso de IA propriamente dita permanece incipiente: a tecnologia é pouco explorada para apoio à decisão, automação inteligente de processos ou personalização de serviços.

Esse quadro sugere a existência de um "gap de engenharia de requisitos" em sistemas de IA. Ahmad *et al.* (2023), ao analisarem práticas de requisitos em projetos de IA centrada no humano, identificam lacunas importantes na definição, documentação e validação de requisitos que contemplem aspectos éticos, contextuais e sociotécnicos. No Brasil, observa-se fenômeno semelhante: digitalizam-se registros e processos burocráticos, mas nem sempre se definem com clareza os requisitos para que esses dados possam ser efetivamente transformados em inteligência útil.

Em outras palavras, há uma adoção significativa de infraestrutura (*hardware*, conectividade, digitalização), mas uma carência de práticas de engenharia que orientem como, onde e para quê aplicar IA. O resultado é um cenário em que muitas organizações "adquirem a ferramenta" impulsionadas, em parte, pelo *hype* descrito pelo Gartner sem desenvolver a "lição de casa" de engenharia de sistemas, tal como enfatizado por Ahmad *et al.* (2023).

1.4 A SOLUÇÃO METODOLÓGICA: ENGENHARIA DE SISTEMAS CENTRADA NO HUMANO (HCAI)

Diante desses desafios tanto no plano global quanto no contexto brasileiro, este memorial adota a hipótese de que a saída passa por uma solução metodológica: a adoção de modelos de maturidade em IA centrada no humano. Wray e Xu (2025) propõem o *Human-Centered AI Maturity Model (HCAI-MM)*, que se destaca por conceber a IA não apenas como um artefato tecnológico, mas como parte de um arranjo organizacional mais amplo.

A proposta desses autores não se limita a "testar" a IA em termos de acurácia ou desempenho algorítmico. Trata-se de representar a organização como um sistema sociotécnico, em que tecnologias, pessoas, processos e valores institucionais

interagem continuamente. Sob essa perspectiva de *design* organizacional, a cognição humana é entendida como componente estrutural do ciclo de processamento: o humano atua não apenas como usuário final, mas como coautor, avaliador e validador.

Essa abordagem dialoga diretamente com a preocupação de Shojaee *et al.* (2025) em relação à "ilusão de pensamento". Ao reconhecer o humano como elemento essencial na validação lógica e contextual das respostas geradas por modelos, o *HCAI-MM* fornece um arcabouço para mitigar riscos e alinhar a tecnologia às necessidades reais das organizações. Em vez de buscar a substituição plena do julgamento humano, propõe-se sua integração deliberada ao projeto dos sistemas.

Ahmad *et al.* (2023) reforçam que as práticas tradicionais de engenharia de *software* não são suficientes para lidar com sistemas de IA. A definição de requisitos passa a demandar a consideração de nuances éticas, sociais e contextuais, como transparência, explicabilidade, vieses algorítmicos e impacto sobre o trabalho humano. Essas dimensões são, inclusive, enfatizadas na Estratégia Brasileira de Inteligência Artificial (Brasil, 2025), mas ainda não se encontram completamente operacionalizadas pelas organizações brasileiras, como sugerem os dados da TIC Empresas e da TIC Cultura (CGI.br, 2025a; 2025b).

Dessa forma, a adoção de um modelo de maturidade em IA centrada no humano configura tanto um desafio quanto uma oportunidade. Desafio, porque exige revisão de práticas estabelecidas de gestão de TI, governança de dados e desenvolvimento de *software*. Oportunidade, porque oferece um caminho estruturado para transformar investimentos em IA em resultados sustentáveis, alinhando tecnologia, estratégia e valores organizacionais.

1.5 CONSIDERAÇÕES FINAIS

A análise realizada neste memorial indica que a transição de um debate predominantemente ético para uma abordagem orientada à engenharia de sistemas constitui um imperativo do momento atual no campo da IA. As evidências técnicas, como as de Gao *et al.* (2025), Xie *et al.* (2025) e Shojaee *et al.* (2025), demonstram que os modelos evoluíram significativamente em termos de capacidade de raciocínio e geração de linguagem. Contudo, os relatórios de mercado de Gartner (2025), Boston Consulting Group (2024) e Deloitte (2025) evidenciam que a captura de valor concreto pelas organizações ainda é limitada e desigual.

No contexto brasileiro, observa-se um cenário promissor, mas incompleto: de um lado, uma estratégia nacional robusta e ambiciosa (Brasil, 2025); de outro, um nível ainda modesto de uso de IA em empresas e instituições culturais, segundo o CGI.br (2025a; 2025b). A digitalização de processos e acervos avançou, mas a transformação desses dados em inteligência efetiva permanece como desafio central.

Conclui-se que o sucesso da IA no Brasil dependerá menos da mera adoção de novos algoritmos e mais da capacidade de projetar sistemas sociotécnicos centrados no humano. Isso envolve práticas sólidas de engenharia de requisitos, governança de dados, modelos de maturidade em IA e, principalmente, um olhar organizacional que reconheça a tecnologia como meio para ampliar capacidades humanas, e não como fim em si mesma ou, ainda, como simples justificativa para expansão de orçamentos de TI.

Em síntese, a consolidação de uma IA verdadeiramente centrada no humano requer a articulação entre avanço técnico, *design* organizacional, políticas públicas e formação de profissionais capazes de atuar nesse espaço interdisciplinar. Trata-se de um desafio expressivo, mas que se apresenta, ao mesmo tempo, como uma oportunidade estratégica para o desenvolvimento tecnológico e social do país.

REFERÊNCIAS

AHMAD, L. et al. Requirements practices and gaps when engineering human-centered Artificial Intelligence systems. **Applied Soft Computing**, v. 143, 2023. DOI: 10.1016/j.asoc.2023.110421.

BOSTON CONSULTING GROUP. **Where's the Value in AI?** Boston: BCG, 2024.

BRASIL. Ministério da Ciência, Tecnologia e Inovação. **Estratégia Brasileira de Inteligência Artificial 2020–2029**. Brasília: MCTI, 2025.

CGI.br. **Pesquisa sobre o uso das Tecnologias de Informação e Comunicação nas Empresas Brasileiras – TIC Empresas 2024/2025**. São Paulo: Comitê Gestor da Internet no Brasil, 2025a.

CGI.br. **Uso de inteligência artificial em equipamentos culturais brasileiros é incipiente, mas digitalização avança, revela pesquisa TIC Cultura 2024**. São Paulo: Comitê Gestor da Internet no Brasil, 2025b.

DELOITTE. **Tech Trends 2025: The AI Budget Dilemma**. Deloitte Insights, 2025.

GAO, S. et al. Augmenting LLMs' Reasoning by Reinforcing Abstract Thinking. **arXiv preprint** arXiv:2500.07751, 2025.

GARTNER. **Hype Cycle for Artificial Intelligence 2025**. Stamford: Gartner, 2025.

SHOJAEI, P. et al. The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Loss of Problem Complexity. **arXiv preprint** arXiv:2500.08041, 2025.

WRAY, S.; XU, W. Human-Centered AI Maturity Model (HCAI-MM): An Organizational Design Perspective. **arXiv preprint** arXiv:2512.14977, 2025. DOI: 10.48550/arXiv.2512.14977.

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA

A – ENUNCIADO

1 ChatGPT

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

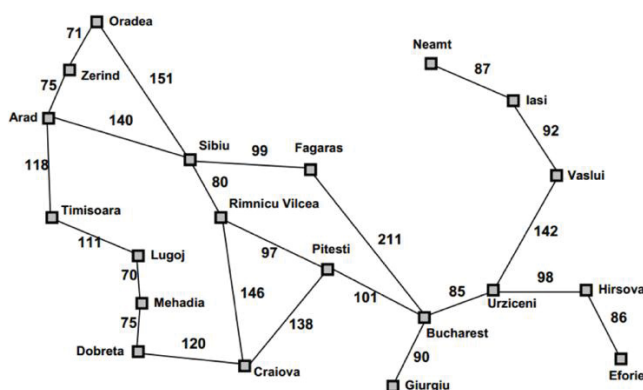
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

Figura 3.22 Valores de *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: $\alpha \rightarrow \beta, \beta \rightarrow \gamma \vdash \alpha \rightarrow \gamma$

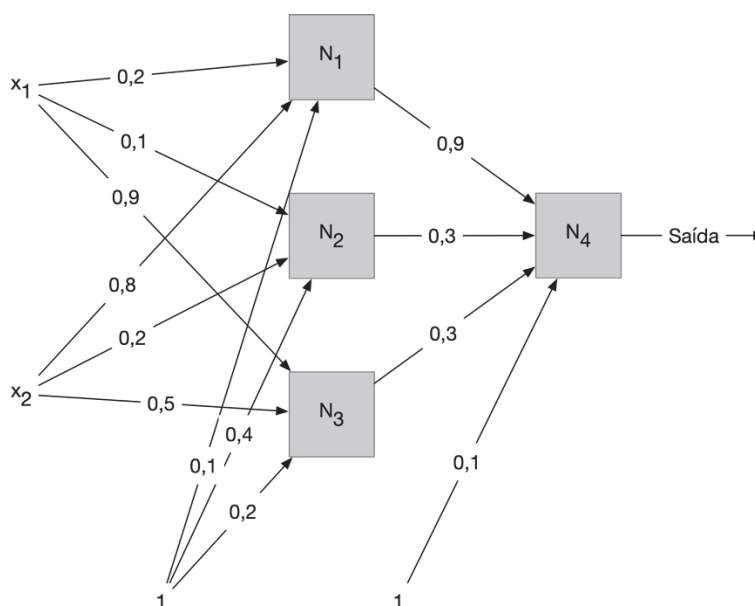
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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 N1, N2 e N3 possuem função de ativação linear. Já N4 possui função de ativação tangente hiperbólica (pesquise a fórmula e aplique).

Dada a entrada $x_1 = -3$, $x_2 = 1$, dê:

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

B – RESOLUÇÃO

1 ChatGPT

- a) Inteligência Artificial (IA) refere-se à capacidade das máquinas de realizar tarefas que normalmente exigiriam inteligência humana. Isso inclui a capacidade de aprender com dados, adaptar-se a novas situações, realizar raciocínio, tomar decisões, resolver problemas e realizar tarefas específicas.

As técnicas de IA podem ser categorizadas de várias maneiras, incluindo aprendizado de máquina, onde os algoritmos são treinados com grandes conjuntos de dados para reconhecer padrões e fazer previsões ou tomar decisões; e aprendizado profundo (deep learning), uma subcategoria do aprendizado de máquina que utiliza redes neurais artificiais com muitas camadas (daí o termo "profundo") para aprender representações complexas dos dados. As aplicações da IA são diversas e incluem reconhecimento de fala, reconhecimento de imagem, processamento de linguagem natural, sistemas de recomendação, veículos autônomos, robótica, diagnóstico médico, jogos, finanças e muitos outros campos. A IA está cada vez mais presente em nossas vidas, impulsionando avanços tecnológicos e transformando a maneira como trabalhamos, vivemos e interagimos com o mundo ao nosso redor.

- b) A definição de Inteligência Artificial dada pelo ChatGPT, em geral, classifica-se dentro da abordagem "agir como os humanos", que se caracteriza por ver a IA como a capacidade de reproduzir ou imitar as ações humanas. Observa-se que, segundo o ChatGPT, a IA "refere-se à capacidade das máquinas de **realizar tarefas** que normalmente exigiriam inteligência humana". Nota-se que o critério usado para definir a IA refere-se a uma ação humana (realizar tarefas). Embora o texto mencione atributos do pensamento humano, como raciocínio e aprendizado, esses são tratados no contexto das técnicas usadas para atingir o objetivo, que é o de realizar tarefas (agir) como humanos. Além disso, os exemplos citados de aplicações da IA referem-se a **atividades/ações** desempenhadas com a inteligência humana, como reconhecimento de imagens, processamento de linguagem natural, etc.

- c) ChatGPT é um chatbot online de inteligência artificial desenvolvido pela OpenAI, lançado em novembro de 2022. O nome "ChatGPT" combina "Chat", referindo-se à sua funcionalidade de chatbot, e "GPT", que significa Generative Pre-trained Transformer, um tipo de modelo de linguagem grande. Fonte: ChatGPT - <https://pt.m.wikipedia.org/wiki/ChatGPT> em 27/02/2024. O ChatGPT é um produto que surge da convergência de duas importantes revoluções na área de Inteligência Artificial (IA) e Processamento de Línguas Naturais (PLN). Em sua essência, ele emprega a tecnologia de modelagem de transformers, que consiste em conjuntos de redes neurais artificiais projetadas para serem altamente "atentas" ao que estão aprendendo. Esses modelos se baseiam em representações numéricas das palavras da língua, conhecidas como word embeddings, as quais são fundamentadas em hipóteses linguísticas clássicas. Segundo essas hipóteses, os significados das palavras podem ser deduzidos a partir dos contextos em que são usadas.

Ao combinar os transformers com as word embeddings, sistemas como o ChatGPT conseguem aprender com sucesso estruturas da língua, padrões de escrita e associações entre diferentes padrões. Isso é viabilizado por meio de um treinamento intensivo em uma grande quantidade de textos "crus", ou seja, não processados, sem a necessidade de intervenção humana para análise linguística dos dados. O ChatGPT se destaca entre outras soluções de IA ao incorporar uma pequena quantidade de dados "curados" fornecidos por humanos. Essa abordagem tem o propósito de reforçar o aprendizado e realizar moderação de conteúdo para identificar discursos de ódio, sexistas ou violentos, permitindo ao sistema lidar de forma mais eficaz com conteúdos indesejados.

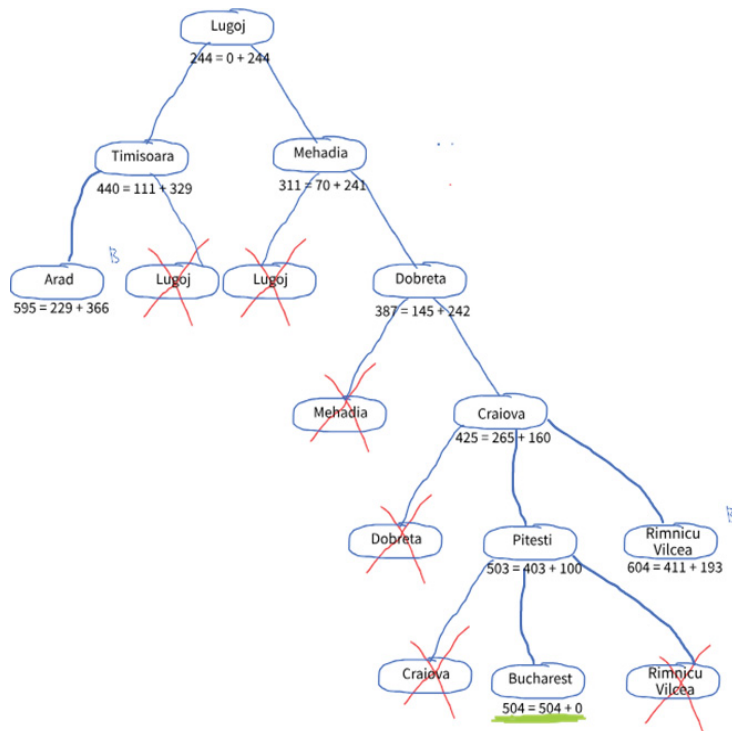
A contribuição do ChatGPT reside na sua capacidade de compreender e gerar linguagem natural de forma contextualmente relevante e coerente. Ele é capaz de extrair significados a partir dos contextos fornecidos, permitindo interações mais naturais e inteligentes com os usuários. Essa habilidade é resultado direto da sofisticação dos modelos de transformers e das word embeddings, os quais capacitam o sistema a capturar nuances semânticas e sintáticas da linguagem.

Além disso, a utilização de uma pequena quantidade de dados "curados" introduzidos por humanos confere ao ChatGPT uma vantagem adicional na detecção e moderação de conteúdos prejudiciais. Essa abordagem híbrida combina a eficiência dos algoritmos de IA com o discernimento humano, resultando em uma maior capacidade de filtrar e gerenciar informações. Dessa forma, o ChatGPT não apenas oferece interações mais naturais e inteligentes, mas também se torna mais responsável e apto a lidar com os desafios éticos e sociais relacionados ao conteúdo online. Fonte: ChatGPT: o robô que mostra como a inteligência artificial pode revolucionar nossas vidas – [ChatGPT: o robô que mostra como a inteligência artificial pode revolucionar nossas vidas – Jornal da USP](#) em 27/02/2024.

- d) Podemos classificar o ChatGPT como um agente que busca “Agir como os Humanos”, pois através de uma simples conversa com o chat ele nos responde como um humano do outro lado, não sendo possível distinguir a máquina do humano. Além disso, ele busca aprender com as nossas preferências e oferece dicas como respostas das solicitações. Percebe-se que se você solicitar algo que seja moralmente e eticamente incorretos na visão humana, o ChatGPT está preparado, pelo menos nessas últimas versões, para nos informar que a pesquisa que queremos realizar não condiz com uma conduta correta.

2 Busca Heurística

a)



3 Lógica

a) Encontrar $q \leftrightarrow p$

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

R4: $r \rightarrow p$

R5: $q \rightarrow p$

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

R7: $q \leftrightarrow p$

EI, R3

SH, R2, R4

CONJ, R5, R1

EB, R6

4 Redes Neurais Artificiais

	A	B	
1	x1	-3	
2	wx1n1	0,2	
3	wx1n2	0,1	
4	wx1n3	0,9	
5	x2	1	
6	wx2n1	0,8	
7	wx2n2	0,2	
8	wx2n3	0,5	
9	b	1	
10	wbn1	0,1	
11	wbn2	0,4	
12	wbn3	0,2	
13	u1	0,3	'u1=B1*B2+B5*B6+B9*B10'
14	N1	0,3	'f(u)= u1'
15	u2	0,3	'u2=B1*B3+B5*B7+B9*B11'
16	N2	0,3	'f(u)= u2'
17	u3	-2	'u3=B1*B4+B5*B8+B9*B12'
18	N3	-2	'f(u)= u3'
19	wn1n4	0,9	
20	wn2n4	0,3	
21	wn3n4	0,3	
22	wbn4	0,1	
23	u4	-0,14	'u4=B14*B19+B16*B20+B18*B21+B9*B22'
24	N4	-0,139092448	'f(u)=(EXP(2*B23) - 1)/(EXP(2*B23) + 1)'

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

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

B - RESOLUÇÃO

1 Análise Exploratória dos dados

```
#Importações das bibliotecas a serem utilizadas
# Biblioteca Pandas - Manipulação de dados
import pandas as pd
# Biblioteca Seaborn - Criação de gráficos
import seaborn as sns
# Biblioteca Matplotlib - Criação de gráficos
import matplotlib.pyplot as plt
# Biblioteca para ignorar mensagens de warning (aviso) ao rodar uma célula de código
import warnings
warnings.filterwarnings('ignore')
```

a)

```
#Carrega informações a partir do CSV
dados = pd.read_csv('precos_carros_brasil.csv')
```

b)

```
#Verifica a existência/quantidade de valores faltantes
dados.isna().sum()
year_of_reference      65245
month_of_reference     65245
fipe_code              65245
authentication         65245
brand                 65245
model                 65245
fuel                  65245
gear                  65245
engine_size           65245
year_model            65245
avg_price_brl         65245
dtype: int64
```

```
#Remove as linhas com valores faltantes e verifica se funcionou
dados.dropna(inplace=True)
dados.isna().sum()
# verifica se funcionou
year_of_reference      0
month_of_reference     0
fipe_code              0
authentication         0
brand                 0
model                 0
fuel                  0
gear                  0
engine_size           0
year_model            0
avg_price_brl         0
dtype: int64
```

c)

```
#Verifica a existência/quantidade de linhas duplicadas
dados.duplicated().sum()
2
#Remove as duplicidades e verifica se funcionou
dados.drop_duplicates(inplace=True)
dados.duplicated().sum()
0
```

d)

```
#Verifica o tipo de dados das colunas
dados.dtypes
year_of_reference    float64
month_of_reference   object
fipecode             object
authentication       object
brand               object
model               object
fuel                object
gear               object
engine_size         object
year_model          float64
avg_price_brl       float64
dtype: object
```

```
#Passando 'year_of_reference' e 'year_model' para serem tratadas como objeto
e não numérico
```

```
dados['year_of_reference'] = dados['year_of_reference'].astype('object')
dados['year_model'] = dados['year_model'].astype('object')
```

```
#Criando categorias para separar colunas numéricas e categóricas
numericas_cols = [col for col in dados.columns if dados[col].dtype !=
'object']
categoricas_cols = [col for col in dados.columns if dados[col].dtype ==
'object']
```

```
# Resumo das variáveis numéricas
dados[numericas_cols].describe()
```

	avg_price_brl
count	202295.000000
mean	52756.765713
std	51628.912116
min	6647.000000
25%	22855.000000
50%	38027.000000
75%	64064.000000
max	979358.000000

```
# Resumo das variáveis categóricas
dados[categoricas_cols].describe()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model
count	202295.0	202295	202295	202295	202295	202295	202295	202295	202295	202295.0
unique	3.0	12	2091	202295	6	2112	3	2	29	24.0
top	2022.0	January	003281-6	cfzlcztzfwrcp	Fiat	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6	2015.0
freq	97797.0	24260	425	1	44962	425	168684	161883	47420	10900.0

e)

```
# Contagem de valores por categoria de Modelo
dados['model'].value_counts()
```

model

```
Palio Week. Adv/Adv TRYON 1.8 mpi Flex      425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p      425
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: count, Length: 2112, dtype: int64
```

```
# Contagem de valores por categoria de Marca
dados['brand'].value_counts()
```

brand

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

f) Realizada a importação das bibliotecas para podermos utilizar o método "read" para importar a base de dados. Depois removemos as linhas em branco dos dados importados e verificamos as duplicidades e as removemos. Verificamos os tipos de objetos para ajustarmos, criando duas categorias: a numérica e a categórica. Depois criamos os resumos das variáveis numéricas e categóricas. Por fim, fizemos a contagem por categoria de Modelo e a contagem por categoria de Marca.

2 Visualização dos Dados

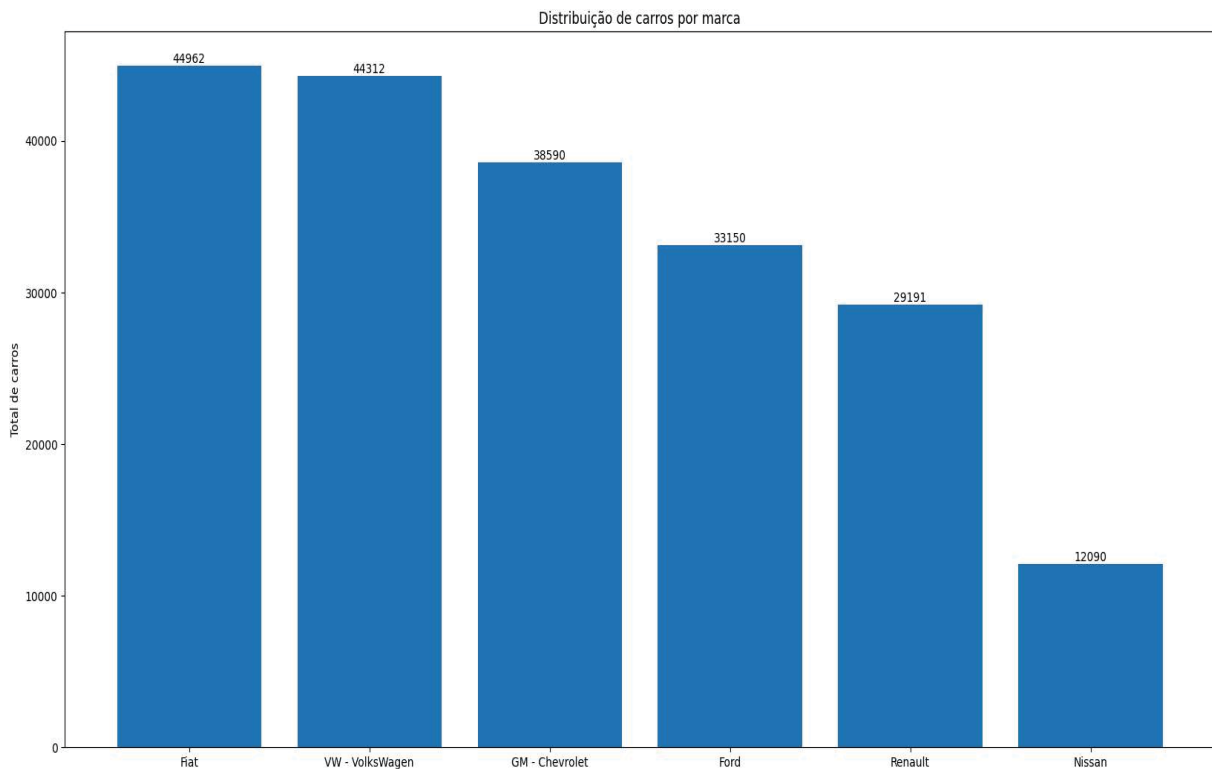
a)

```
#Seleção dos dados para o gráfico
valores_contados_p_marca = dados['brand'].value_counts()

#Geração do gráfico de distribuição
plt.figure(figsize=(20,10))
grafico_carros_p_marca = plt.bar(valores_contados_p_marca.index,
valores_contados_p_marca.values)

#Adição de detalhamento ao gráfico
plt.title('Distribuição de carros por marca')
plt.ylabel('Total de carros')
plt.bar_label(grafico_carros_p_marca, size=10)

[Text(0, 0, '44962'),
Text(0, 0, '44312'),
Text(0, 0, '38590'),
Text(0, 0, '33150'),
Text(0, 0, '29191'),
Text(0, 0, '12090')]
```

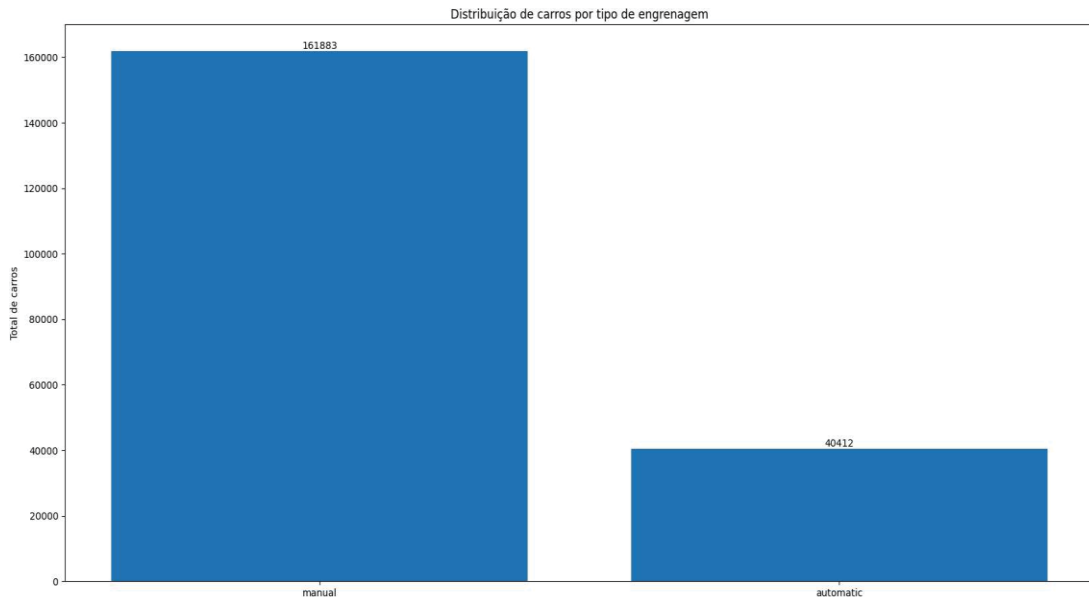


b)

```
#Seleção dos dados para o gráfico
valores_contados_p_gear = dados['gear'].value_counts()
#Geração do gráfico de distribuição
plt.figure(figsize=(20,10))
grafico_carros_p_gear = plt.bar(valores_contados_p_gear.index,
valores_contados_p_gear.values)

#Adição de detalhamento ao gráfico
```

```
plt.title('Distribuição de carros por tipo de engrenagem')
plt.ylabel('Total de carros')
plt.bar_label(grafico_carros_p_gear, size=10)
[Text(0, 0, '161883'), Text(0, 0, '40412')]
```



c)

```
# Filtrando os carros pelo ano de 2022
carros_ano = dados[dados['year_of_reference'] == 2022]
# Calculando a média por mês (coluna month_of_reference) para saber se a
média de preço ao longo dos meses dos carros no ano de 2022
avg_per_mouth = carros_ano.groupby(['month_of_reference',
'year_of_reference'])['avg_price_brl'].mean().round(2) # round para
arredondar
# Verificando a alteração
avg_per_mouth.head()
```

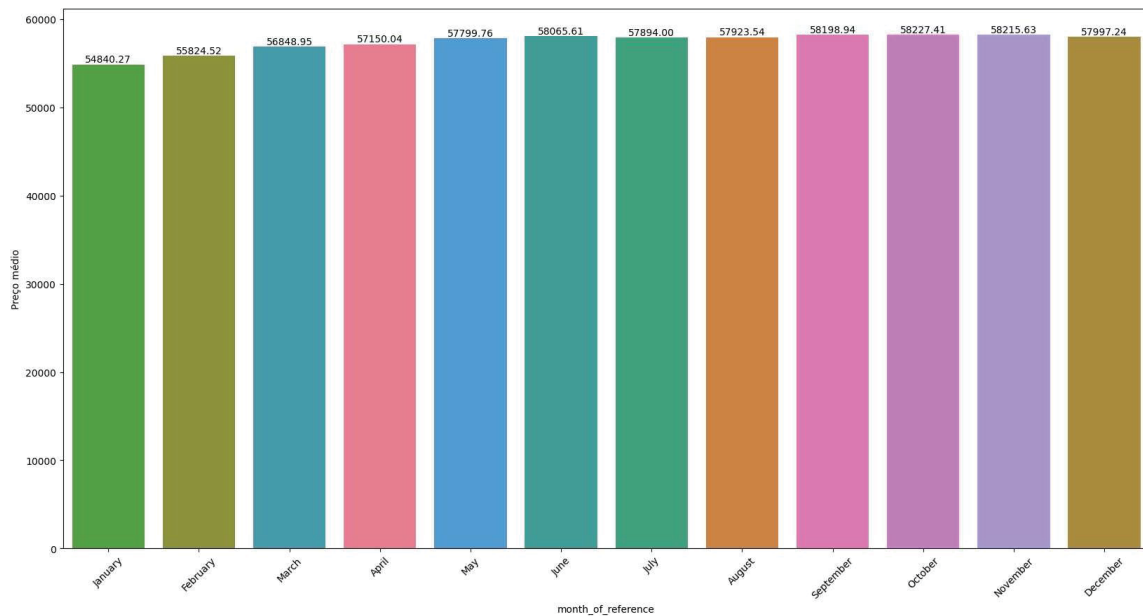
```
month_of_reference    year_of_reference
April                2022.0      57150.04
August               2022.0      57923.54
December             2022.0      57997.24
February             2022.0      55824.52
January              2022.0      54840.27
Name: avg_price_brl, dtype: float64
```

```
# Utilizando a função reset_index para criar uma ordem e facilitar a
criação do gráfico
avg_per_mouth = avg_per_mouth.reset_index(name='Preço médio')
avg_per_mouth.head()
```

	month_of_reference	year_of_reference	Preço médio
0	April	2022.0	57150.04
1	August	2022.0	57923.54
2	December	2022.0	57997.24

	month_of_reference	year_of_reference	Preço médio
3	February	2022.0	55824.52
4	January	2022.0	54840.27

```
# Visualizando a média mensal dos preços dos carros em 2022
plt.figure(figsize=(20,10))
# Criação do gráfico com a média mensal dos preços dos carros em 2022
ax = sns.barplot(x='month_of_reference', y='Preço médio',
hue='month_of_reference', data=avg_per_month,
order=['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'S
eptember', 'October', 'November', 'December'])
# Apresentação dos valores nas barras do gráfico
for bars in ax.containers:
    ax.bar_label(bars, fmt='%.2f')
# Rotação do label da barra para ficar em 45°
plt.xticks(rotation=45);
```



d)

```
# Agrupamento por marca e tipo de engrenagem com a média de preço dos carros
# Cálculo da média por marca e tipo de engrenagem
media_marca_engrenagem = dados.groupby(['brand',
'gear'])['avg_price_brl'].mean().round(2)
media_marca_engrenagem.head()
```

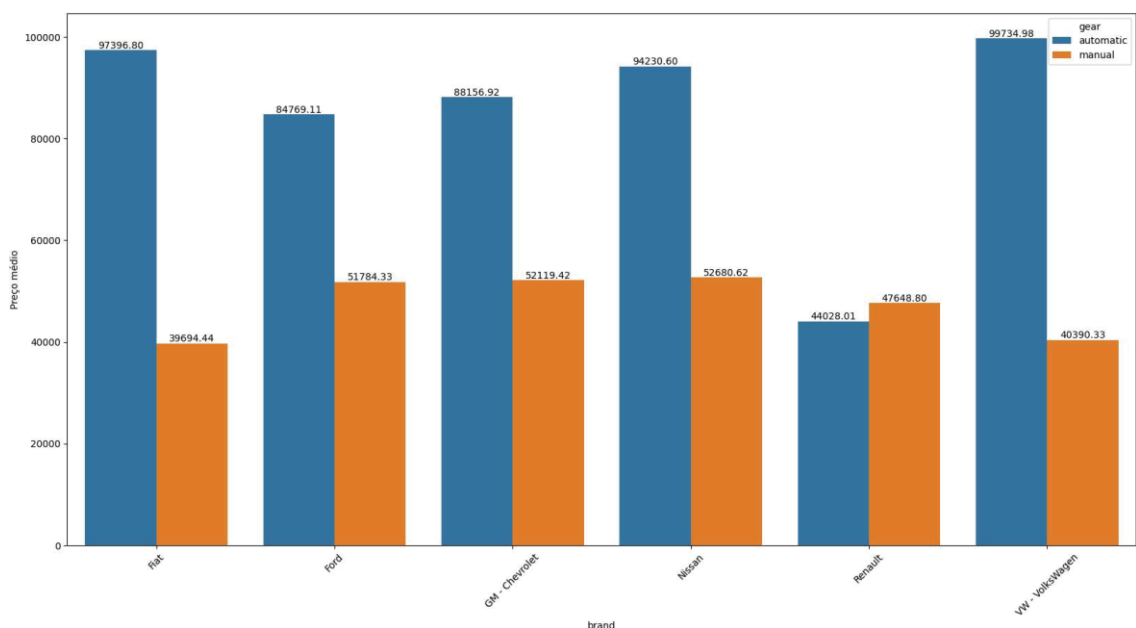
```
brand      gear
Fiat      automatic    97396.80
          manual      39694.44
Ford      automatic    84769.11
          manual      51784.33
GM - Chevrolet automatic    88156.92
Name: avg_price_brl, dtype: float64
```

```
# Utilizando a função reset_index para criar uma ordem e facilitar a
criação do gráfico
media_marca_engrenagem = media_marca_engrenagem.reset_index(name='Preço
médio')
```

```
media_marca_engrenagem.head()
```

	brand	gear	Preço médio
0	Fiat	automatic	97396.80
1	Fiat	manual	39694.44
2	Ford	automatic	84769.11
3	Ford	manual	51784.33
4	GM - Chevrolet	automatic	88156.92

```
# Visualizando a média de preço dos carros por marca e tipo de engrenagem
plt.figure(figsize=(20,10))
# Criação do gráfico com a média de preço dos carros por marca e tipo de engrenagem
mt = sns.barplot(x='brand', y='Preço médio', hue='gear',
data=media_marca_engrenagem, hue_order=['automatic', 'manual']);
# Apresentação dos valores nas barras do gráfico
for bars in mt.containers:
    mt.bar_label(bars, fmt='%.2f')
# Rotação do label da barra para ficar em 45°
plt.xticks(rotation=45);
```



e) Realizamos o agrupamento por marca e tipo de engrenagem (utilizando as colunas "brand" e "gear") para encontrar a média de preço dos carros e arredondamos para duas casas decimais. Logo após, foi utilizada uma função para ordenação de índice para facilitar a visualização do gráfico de barras, separando as marcas por preço e o tipo de engrenagem. Para uma melhor visualização, decidimos utilizar o laço "for" para escrever nas barras o preço médio de cada tipo de engrenagem por marca.

f)

```
# Agrupamento por marca e tipo de combustível com a média de preço dos carros
# Cálculo da média por marca e tipo de combustível
```

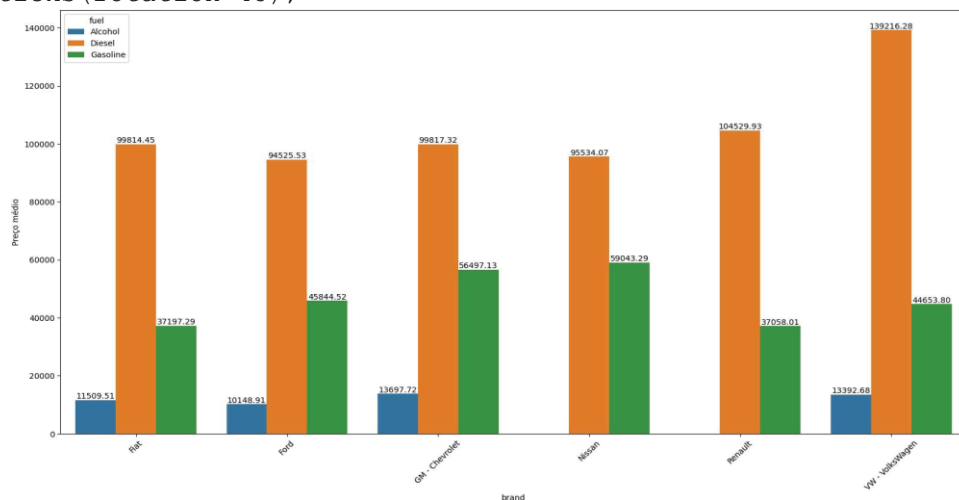
```
media_marca_fuel =
dados.groupby(['brand', 'fuel'])['avg_price_brl'].mean().round(2)
media_marca_fuel.head()
```

```
brand    fuel
Fiat     Alcohol    11509.51
         Diesel     99814.45
         Gasoline   37197.29
Ford     Alcohol    10148.91
         Diesel     94525.53
Name: avg_price_brl, dtype: float64
```

```
# Utilizando a função reset_index para criar uma ordem e facilitar a
criação do gráfico
media_marca_fuel = media_marca_fuel.reset_index(name='Preço médio')
media_marca_fuel.head()
```

	brand	fuel	Preço médio
0	Fiat	Alcohol	11509.51
1	Fiat	Diesel	99814.45
2	Fiat	Gasoline	37197.29
3	Ford	Alcohol	10148.91
4	Ford	Diesel	94525.53

```
# Visualizando a média de preço dos carros por marca e tipo de engrenagem
plt.figure(figsize=(20,10))
# Criação do gráfico com a média de preço dos carros por marca e tipo de
engrenagem
mf = sns.barplot(x='brand', y='Preço médio', hue='fuel',
data=media_marca_fuel, hue_order=['Alcohol', 'Diesel', 'Gasoline']);
# Apresentação dos valores nas barras do gráfico
for bars in mf.containers:
    mf.bar_label(bars, fmt='%.2f')
# Rotação do label da barra para ficar em 45°
plt.xticks(rotation=45);
```



g) Realizamos o agrupamento por marca e tipo de combustível (utilizando as colunas "brand" e "fuel") para encontrar a média de preço dos carros e arredondamos para duas casas decimais. Logo após, foi utilizada uma função

para ordenação de índice para facilitar a visualização do gráfico de barras, separando as marcas por preço e os tipos de combustível, pois nesse temos três barras distintas. Para uma melhor visualização, decidimos utilizar o laço "for" para escrever nas barras o preço médio de cada tipo de combustível por marca.

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

a)

```
# Bibliotecas de machine learning
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import LabelEncoder

# Métricas de avaliação dos modelos
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
#Verificando as colunas e seus tipos de dados
dados.dtypes

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

# Variável dados_num contém apenas variáveis numéricas de interesse (exclui
o restante)
dados_num = dados.drop(['authentication', 'fipecode'],axis = 1)
dados_num.head()
```

	year_of_ reference	month_of_ reference	brand	model	fuel	gear	engine _size	year_ model	avg_price _brl
0	2021.0	January	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasol ine	manual	1	2002.0	9162.0
1	2021.0	January	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasol ine	manual	1	2001.0	8832.0

2	2021.0	January	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000.0	8388.0
3	2021.0	January	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000.0	8453.0
4	2021.0	January	GM - Chevrolet	Corsa Pick-Up GL/Cham p 1.6 MPFI / EFI	Gasoline	manual	1,6	2001.0	12525.0

#Coluna gear tem dois valores: automático e manual. Após aplicar a função abaixo, ficará com valores 0 e 1 (1 manual e 0 automático)

```
dados_num['gear'] = LabelEncoder().fit_transform(dados_num['gear'])
```

#Coluna fuel tem três valores: gasolina, álcool e diesel. Após aplicar a função abaixo, ficará com valores 0, 1 e 2 (2 gasolina, 1 diesel e 0 álcool)

```
dados_num['fuel'] = LabelEncoder().fit_transform(dados_num['fuel'])
```

#Convertendo marca para variável numérica

```
dados_num['brand'] = LabelEncoder().fit_transform(dados_num['brand'])
```

#Convertendo modelo para variável numérica

```
dados_num['model'] = LabelEncoder().fit_transform(dados_num['model'])
```

#deve-se remover as virgulas antes da conversão

```
dados_num['engine_size'] = dados_num['engine_size'].str.replace(',', '.')
```

#Convertendo 'engine_size', avg_price_brl, year_of_reference e year_model para inteiro

```
dados_num['engine_size'] = dados_num['engine_size'].astype('float64')
```

```
dados_num['year_of_reference'] =
```

```
dados_num['year_of_reference'].astype('int32')
```

```
dados_num['year_model'] = dados_num['year_model'].astype('int32')
```

```
dados_num['avg_price_brl'] = dados_num['avg_price_brl'].astype('int32')
```

Dicionário de mapeamento

```
meses_dict = {
    'January': 1,
    'February': 2,
    'March': 3,
    'April': 4,
    'May': 5,
    'June': 6,
```

```

    'July':      7,
    'August':    8,
    'September': 9,
    'October':   10,
    'November':  11,
    'December':  12
}

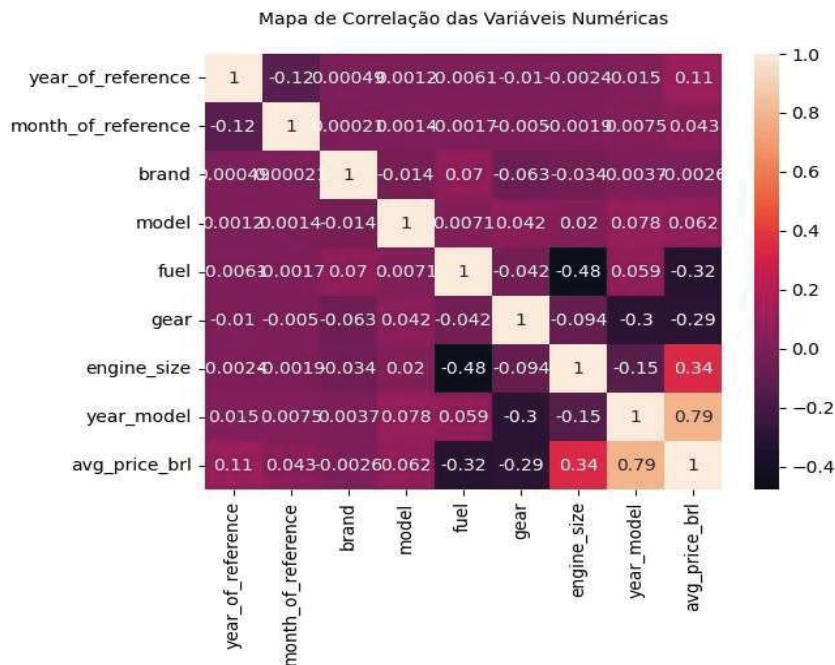
# Aplicar a conversão na coluna month_of_reference para números inteiros
dados_num['month_of_reference'] =
dados_num['month_of_reference'].map(meses_dict)

#Verificando o sucesso das conversões
dados_num.dtypes

year_of_reference      int32
month_of_reference     int64
brand                  int32
model                  int32
fuel                   int32
gear                   int32
engine_size            float64
year_model              int32
avg_price_brl          int32
dtype: object

# Mapa de correlação das variáveis numéricas com variável Target
sns.heatmap(dados_num.corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 10)
plt.show()

```



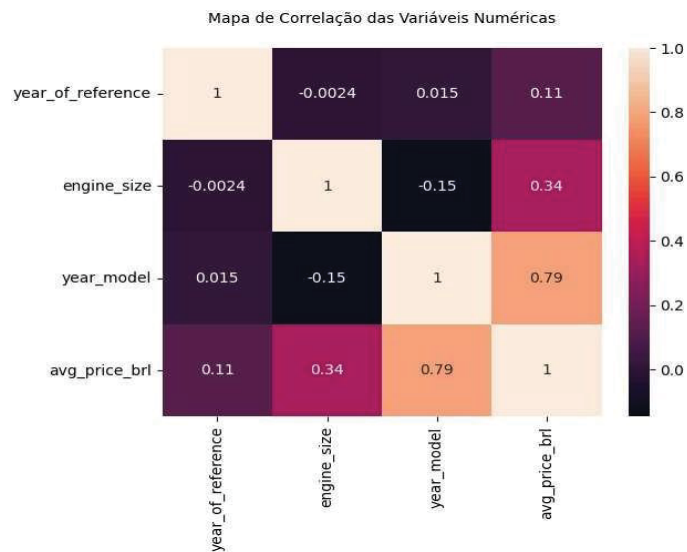
```

#removendo variáveis fracas, selecionando variáveis numéricas de interesse
para a análise
dados_num = dados_num.drop(['month_of_reference', 'brand', 'model', 'fuel',
'gear'],axis = 1)

# Mapa de correlação das variáveis numéricas com variável Target
sns.heatmap(dados_num.corr("spearman"), annot = True)

```

```
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 10)
plt.show()
```



```
# Variável X contém apenas variáveis numéricas de interesse para a análise,
excluindo a variável target
X = dados_num.drop(['avg_price_brl'],axis = 1)
X.head()
```

	year_of_reference	engine_size	year_model
0	2021	1.0	2002
1	2021	1.0	2001
2	2021	1.0	2000
3	2021	1.0	2000
4	2021	1.6	2001

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

```
0    9162
1    8832
2    8388
3    8453
4   12525
```

```
Name: avg_price_brl, dtype: int32
```

b)

```
# Divisão: 25% dos dados são de teste e 75% de treinamento
```

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

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

```
(151721, 3)
```

	year_of_reference	engine_size	year_model
156364	2022	2.3	2020
93758	2021	1.6	2011
43313	2021	2.0	2008
114972	2022	2.3	2013
69993	2021	1.0	2018

```
# Observando a variável target
Y_test.head()
```

```
180633    42595
13130     10989
163315     9087
121464    26965
14044     57102
Name: avg_price_brl, dtype: int32
```

c)

```
##### C.1 - Algoritmo Random Forest, sem especificar nenhum parâmetro
```

```
# Algoritmo Random Forest, sem especificar nenhum parâmetro
model_rf = RandomForestRegressor()
```

```
# Ajuste do modelo, de acordo com as variáveis de treinamento
model_rf.fit(X_train, Y_train)
```

```
▼ RandomForestRegressor ⓘ ?
RandomForestRegressor()
```

```
##### C.2 - Algoritmo Random Forest, com parâmetros especificados
```

```
# Algoritmo Random Forest, com parâmetros
model_rf_parametros = RandomForestRegressor(n_estimators=300,
min_samples_split=6)
```

```
# Ajuste do modelo, de acordo com as variáveis de treinamento
model_rf_parametros.fit(X_train, Y_train)
```

```
▼ RandomForestRegressor ⓘ ?
RandomForestRegressor(min_samples_split=6, n_estimators=300)
```

```
##### C.3 - XGBoost
```

```
# Algoritmo XGBRegressor, sem especificar nenhum parâmetro
model_xgboost = XGBRegressor()

# Ajuste do modelo, de acordo com as variáveis de treinamento
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_round
s=None,
              enable_categorical=False, eval_metric=None, feature_type
s=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bi
n=None,
```

d)

```
# Predição dos valores com base nos dados de teste
valores_preditos_rf = model_rf.predict(X_test)
valores_preditos_rf
```

```
##### D.1 - Algoritmo Random Forest, sem especificar nenhum parâmetro
```

```
# Predição dos valores com base nos dados de teste
valores_preditos_rf = model_rf.predict(X_test)
valores_preditos_rf
```

```
array([ 47442.35159723, 13375.99359041, 15048.77951718, ...,
        116937.91789323, 14208.70739379, 22781.71018086])
```

```
##### D.2 - Algoritmo Random Forest, com parâmetros especificados
```

```
# Predição dos valores de salário com base nos dados de teste
valores_preditos_rfc = model_rf_parametros.predict(X_test)
valores_preditos_rfc
```

```
array([ 47422.98041018, 13354.30359893, 15027.85045561, ...,
        116948.32572092, 14166.53711641, 22754.38059332])
```

```
##### D.3 - XGBoost
```

```
# Predição dos valores de salário com base nos dados de teste
valores_preditos_xgboost = model_xgboost.predict(X_test)
valores_preditos_xgboost
```

```
array([ 47451.33 , 13334.936, 15008.233, ..., 116279.016, 14095.22 ,
        22785.436], dtype=float32)
```

e)

```
##### E.1 - Algoritmo Random Forest, sem especificar nenhum parâmetro
```

```
#Análise de importância das variáveis
model_rf.feature_importances_
pd.DataFrame(model_rf.feature_importances_, index = X_train.columns,
columns=['importance']).sort_values('importance', ascending = False)
```

```

                importance
engine_size      0.544408
year_model       0.440935
year_of_reference 0.014657
```

```
##### E.2 - Algoritmo Random Forest, com parâmetros especificados
```

```
#Análise de importância das variáveis
model_rf_parametros.feature_importances_
pd.DataFrame(model_rf_parametros.feature_importances_, index =
X_train.columns, columns=['importance']).sort_values('importance',
ascending = False)
```

```

                importance
engine_size      0.544850
year_model       0.440614
year_of_reference 0.014535
```

```
##### E.3 - XGBoost
```

```
#Análise de importância das variáveis
model_xgboost.feature_importances_
pd.DataFrame(model_xgboost.feature_importances_, index = X_train.columns,
columns=['importance']).sort_values('importance', ascending = False)
```

```

                importance
engine_size      0.533692
year_model       0.439572
year_of_reference 0.026735
```

f) Foram utilizadas as bibliotecas "Random Forest" e "XGBoost" para a aprendizagem de máquina com os modelos. Vimos que com o "Random Forest sem parâmetros" e "com os parâmetros" houve leve melhora nas importâncias das acurácias, mas não tão significativas. Depois, foi utilizado o "XGBoost" para ver a possibilidade de melhora entre as variáveis, o que fez melhorar a variável "year_of_reference" em quase 100%, mas ela não é uma variável tão relevante para o modelo.

g)

```
##### G.1 - Algoritmo Random Forest, sem especificar nenhum parâmetro
```

```
#MSE - calcula o erro quadrático médio das predições do nosso modelo. Quanto maior o MSE, pior é o modelo.
```

```
mean_squared_error(Y_test, valores_preditos_rf)
```

```
231721177.30131346
```

```
#O MAE calcula a média da diferença absoluta entre o valor predito e o valor real. Nesse caso, os erros são penalizados linearmente, ou seja, todos terão o mesmo peso na média.
```

```
mean_absolute_error(Y_test, valores_preditos_rf)
```

8599.640199577498

#O R^2 é uma métrica que varia entre 0 e 1 e é uma razão que indica o quão bom o nosso modelo. Quanto maior seu valor, melhor é o modelo
`r2_score(Y_test, valores_preditos_rf)`
 0.9138985427696947

G.2 - Algoritmo Random Forest, com parâmetros especificados

#MSE - calcula o erro quadrático médio das predições do nosso modelo. Quanto maior o MSE, pior é o modelo.
`mean_squared_error(Y_test, valores_preditos_rfc)`

230888820.97491515

#O MAE calcula a média da diferença absoluta entre o valor predito e o valor real. Nesse caso, os erros são penalizados linearmente, ou seja, todos terão o mesmo peso na média.
`mean_absolute_error(Y_test, valores_preditos_rfc)`

8597.140534628832

#O R^2 é uma métrica que varia entre 0 e 1 e é uma razão que indica o quão bom o nosso modelo. Quanto maior seu valor, melhor é o modelo
`r2_score(Y_test, valores_preditos_rfc)`

0.9142078243531582

G.3 - XGBoost

#MSE - calcula o erro quadrático médio das predições do nosso modelo. Quanto maior o MSE, pior é o modelo.
`mean_squared_error(Y_test, valores_preditos_xgboost)`

231329738.4531412

#O MAE calcula a média da diferença absoluta entre o valor predito e o valor real. Nesse caso, os erros são penalizados linearmente, ou seja, todos terão o mesmo peso na média.
`mean_absolute_error(Y_test, valores_preditos_xgboost)`

8592.144669950685

#O R^2 é uma métrica que varia entre 0 e 1 e é uma razão que indica o quão bom o nosso modelo. Quanto maior seu valor, melhor é o modelo
`r2_score(Y_test, valores_preditos_xgboost)`

0.9140439910866621

h) Foram testadas as métricas de acurácia dos modelos MSE, MAE, R^2 para encontrar qual teria o melhor resultado. Mesmo com a melhoria das variáveis utilizada no modelo de aprendizagem do "XGBoost", todos as acurácias ficaram com o mesmo resultado entre elas não havendo melhora significativa. Ficando a melhor avaliação de métrica em 91% de acurácia em todos os modelos para R^2 .

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

Coeficiente de determinação: R^2

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

onde y_i é o valor observado, $\hat{y}_i$ é o valor predito e $\bar{y}$ é a média dos valores y_i observados.

Quanto mais perto de 1 melhor é o modelo;

Erro padrão da estimativa: S_{yx}

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

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

$S_{yx}\%$

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

Instalação/carregamento de pacotes, bibliotecas e semelhantes.

```
install.packages("mlbench")
## package 'mlbench' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Pedro\AppData\Local\Temp\RtmpuoBWMM\downloaded_packages
install.packages("mice")
## package 'mice' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Pedro\AppData\Local\Temp\RtmpuoBWMM\downloaded_packages
library(mlbench)
library(caret)
## Carregando pacotes exigidos: ggplot2
## Carregando pacotes exigidos: lattice
library(mice)
##
## Attaching package: 'mice'
## The following object is masked from 'package:stats':
##
## filter
## The following objects are masked from 'package:base':
##
## cbind, rbind
1
```

Carregamento da base de dados.

```
data(Satellite)
base <- Satellite[, c("x.17", "x.18", "x.19", "x.20", "classes")]
Criação das bases de treino e teste 80% de treino e 20% para teste.
```

```
set.seed(7)
indicesSat <- createDataPartition(base$classes, p=0.80, list=FALSE)
treinoSat <- base[indicesSat,]
testeSat <- base[-indicesSat,]
```

Treinamento dos modelos RandomForest, SVM e RNA. Utilizando a base de treino.

```
## # weights: 17
## initial value 9342.956165
## iter 10 value 8928.014836
## iter 10 value 8928.014835
## final value 8928.014835
## converged
....
## # weights: 61
## initial value 10332.109654
## iter 10 value 7176.337000
## iter 50 value 4476.201332
## iter 100 value 3170.147484
## final value 3170.147484
## stopped after 100 iterations
```

Predição usando a base de teste.

```
predicoes.rfSat <- predict(rfSat, testeSat)
predicoes.svmSat <- predict(svmSat, testeSat)
predicoes.rnaSat <- predict(rnaSat, testeSat)
```

Matriz de confusão para testar a eficácia dos diferentes modelos.

```
confusionMatrix(predicoes.rfSat, testeSat$classes)
## Confusion Matrix and Statistics
##
## Reference
## Prediction red soil cotton crop grey soil damp grey soil
## red soil 296 1 5 2
## cotton crop 0 123 0 0
## grey soil 3 0 238 29
## damp grey soil 1 0 21 62
## vegetation stubble 6 10 0 0
## very damp grey soil 0 6 7 32
## Reference
## Prediction vegetation stubble very damp grey soil
## red soil 8 0
## cotton crop 2 1
## grey soil 1 10
## damp grey soil 2 40
## vegetation stubble 120 8
## very damp grey soil 8 242
##
## Overall Statistics
##
## Accuracy : 0.8419
## 95% CI : (0.8208, 0.8614)
## No Information Rate : 0.2383
## P-Value [Acc > NIR] : < 2.2e-16
##
## Kappa : 0.8047
##
```

```

## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
## Class: red soil Class: cotton crop Class: grey soil
## Sensitivity 0.9673 0.87857 0.8782
## Specificity 0.9836 0.99738 0.9576
## Pos Pred Value 0.9487 0.97619 0.8470
## Neg Pred Value 0.9897 0.98532 0.9671
## Prevalence 0.2383 0.10903 0.2111
## Detection Rate 0.2305 0.09579 0.1854
## Detection Prevalence 0.2430 0.09813 0.2188
48
## Balanced Accuracy 0.9755 0.93797 0.9179
## Class: damp grey soil Class: vegetation stubble
## Sensitivity 0.49600 0.85106
## Specificity 0.94478 0.97900
## Pos Pred Value 0.49206 0.83333
## Neg Pred Value 0.94560 0.98158
## Prevalence 0.09735 0.10981
## Detection Rate 0.04829 0.09346
## Detection Prevalence 0.09813 0.11215
## Balanced Accuracy 0.72039 0.91503
## Class: very damp grey soil
## Sensitivity 0.8040
## Specificity 0.9461
## Pos Pred Value 0.8203
## Neg Pred Value 0.9403
## Prevalence 0.2344
## Detection Rate 0.1885
## Detection Prevalence 0.2298
## Balanced Accuracy 0.8750
confusionMatrix(predicoes.svmSat, testeSat$classes)
## Confusion Matrix and Statistics
##
## Reference
## Prediction red soil cotton crop grey soil damp grey soil
## red soil 298 1 4 2
## cotton crop 1 120 0 0
## grey soil 4 0 260 29
## damp grey soil 0 1 7 69
## vegetation stubble 3 14 0 2
## very damp grey soil 0 4 0 23
## Reference
## Prediction vegetation stubble very damp grey soil
## red soil 7 0
## cotton crop 4 0
## grey soil 1 12
## damp grey soil 2 32
## vegetation stubble 117 3
## very damp grey soil 10 254
##
## Overall Statistics
##
## Accuracy : 0.8707
## 95% CI : (0.8511, 0.8886)
## No Information Rate : 0.2383
## P-Value [Acc > NIR] : < 2.2e-16
##
## Kappa : 0.8399
##

```

```

## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
49
## Class: red soil Class: cotton crop Class: grey soil
## Sensitivity 0.9739 0.85714 0.9594
## Specificity 0.9857 0.99563 0.9546
## Pos Pred Value 0.9551 0.96000 0.8497
## Neg Pred Value 0.9918 0.98274 0.9888
## Prevalence 0.2383 0.10903 0.2111
## Detection Rate 0.2321 0.09346 0.2025
## Detection Prevalence 0.2430 0.09735 0.2383
## Balanced Accuracy 0.9798 0.92639 0.9570
## Class: damp grey soil Class: vegetation stubble
## Sensitivity 0.55200 0.82979
## Specificity 0.96376 0.98075
## Pos Pred Value 0.62162 0.84173
## Neg Pred Value 0.95226 0.97904
## Prevalence 0.09735 0.10981
## Detection Rate 0.05374 0.09112
## Detection Prevalence 0.08645 0.10826
## Balanced Accuracy 0.75788 0.90527
## Class: very damp grey soil
## Sensitivity 0.8439
## Specificity 0.9624
## Pos Pred Value 0.8729
## Neg Pred Value 0.9527
## Prevalence 0.2344
## Detection Rate 0.1978
## Detection Prevalence 0.2266
## Balanced Accuracy 0.9031
confusionMatrix(predicoes.rnaSat, testeSat$classes)
## Confusion Matrix and Statistics
##
## Reference
## Prediction red soil cotton crop grey soil damp grey soil
## red soil 292 2 11 2
## cotton crop 7 124 0 0
## grey soil 3 0 259 67
## damp grey soil 0 0 0 0
## vegetation stubble 3 5 0 0
## very damp grey soil 1 9 1 56
## Reference
## Prediction vegetation stubble very damp grey soil
## red soil 12 1
## cotton crop 13 0
## grey soil 1 25
## damp grey soil 0 0
## vegetation stubble 100 12
## very damp grey soil 15 263
##
## Overall Statistics
##
## Accuracy : 0.8084
## 95% CI : (0.7858, 0.8296)
## No Information Rate : 0.2383
50
## P-Value [Acc > NIR] : < 2.2e-16
##
## Kappa : 0.7595

```

```
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
## Class: red soil Class: cotton crop Class: grey soil
## Sensitivity 0.9542 0.88571 0.9557
## Specificity 0.9714 0.98252 0.9052
## Pos Pred Value 0.9125 0.86111 0.7296
## Neg Pred Value 0.9855 0.98596 0.9871
## Prevalence 0.2383 0.10903 0.2111
## Detection Rate 0.2274 0.09657 0.2017
## Detection Prevalence 0.2492 0.11215 0.2765
## Balanced Accuracy 0.9628 0.93412 0.9305
## Class: damp grey soil Class: vegetation stubble
## Sensitivity 0.00000 0.70922
## Specificity 1.00000 0.98250
## Pos Pred Value NaN 0.83333
## Neg Pred Value 0.90265 0.96478
## Prevalence 0.09735 0.10981
## Detection Rate 0.00000 0.07788
## Detection Prevalence 0.00000 0.09346
## Balanced Accuracy 0.50000 0.84586
## Class: very damp grey soil
## Sensitivity 0.8738
## Specificity 0.9166
## Pos Pred Value 0.7623
## Neg Pred Value 0.9595
## Prevalence 0.2344
## Detection Rate 0.2048
## Detection Prevalence 0.2687
## Balanced Accuracy 0.8952
```

Conclusão.

Após realizar o treinamento e os testes utilizando os modelos RandomForest, SVM (Support Vector Machine) e RNA (Redes Neurais Artificiais), vimos que o melhor modelo foi o SVM tendo uma Acurácia (métrica) de 0.8707 em comparação com o RF que foi de 0.8419 e de RNA de 0.8084.

Instalação dos pacotes e carregamento das bibliotecas.

```
install.packages("e1071")
## package 'e1071' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

install.packages("randomForest")
## package 'randomForest' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

install.packages("kernlab")
## package 'kernlab' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

install.packages("caret")
## package 'caret' successfully unpacked and MD5 sums checked
```

```
## Warning: cannot remove prior installation of package 'caret'
## Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
## C:\Program Files\R\R-4.3.3\library\00LOCK\caret\libs\x64\caret.dll to
## C:\Program Files\R\R-4.3.3\library\caret\libs\x64\caret.dll: Permission
denied
## Warning: restored 'caret'
##
## The downloaded binary packages are in
## C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages
```

```
install.packages("RSNNS")
## package 'RSNNS' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
## C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages
```

```
library("caret")
## Carregando pacotes exigidos: ggplot2
## Carregando pacotes exigidos: lattice
```

```
library(RSNNS)
## Carregando pacotes exigidos: Rcpp
##
## Attaching package: 'RSNNS'
## The following objects are masked from 'package:caret':
##
## confusionMatrix, train
```

Caregamento do CSV.

```
dadosVolume <- read.csv("http://www.razer.net.br/datasets/Volumes.csv",
sep=";", dec=",")
```

Eliminação da coluna NR.

```
summary(dadosVolume)
## NR DAP HT HP
## Min. : 1.00 Min. :27.50 Min. :22.80 Min. :1.000
## 1st Qu.: 25.75 1st Qu.:32.50 1st Qu.:25.50 1st Qu.:1.700
## Median : 50.50 Median :35.25 Median :26.40 Median :2.225
## Mean : 50.50 Mean :35.86 Mean :26.24 Mean :2.135
## 3rd Qu.: 75.25 3rd Qu.:39.05 3rd Qu.:27.15 3rd Qu.:2.562
## Max. :100.00 Max. :49.00 Max. :28.40 Max. :3.400
## VOL
## Min. :0.6982
## 1st Qu.:1.0553
## Median :1.3006
## Mean :1.3583
## 3rd Qu.:1.6191
## Max. :2.5245
```

```
dadosVolume$NR <- NULL
```

```
summary(dadosVolume)
## DAP HT HP VOL
## Min. :27.50 Min. :22.80 Min. :1.000 Min. :0.6982
## 1st Qu.:32.50 1st Qu.:25.50 1st Qu.:1.700 1st Qu.:1.0553
## Median :35.25 Median :26.40 Median :2.225 Median :1.3006
## Mean :35.86 Mean :26.24 Mean :2.135 Mean :1.3583
## 3rd Qu.:39.05 3rd Qu.:27.15 3rd Qu.:2.562 3rd Qu.:1.6191
## Max. :49.00 Max. :28.40 Max. :3.400 Max. :2.5245
```

Criação das bases de treino e teste 80% de treino e 20% para teste.

```
set.seed(7)
indicesVol <- createDataPartition(dadosVolume$VOL, p=0.80, list=FALSE)
treinoVol <- dadosVolume[indicesVol,]
testeVol <- dadosVolume[-indicesVol,]
```

Treinamento dos modelos RandomForest, SVM, RNA e alom.

Utilizando a base de treino.

```
rfVol <- caret::train(VOL~., data=treinoVol, method="rf")
## note: only 2 unique complexity parameters in default grid. Truncating
the grid to 2 .
svmVol <- caret::train(VOL~., data=treinoVol, method="svmRadial")
rnaVol <- caret::train(VOL~., data=treinoVol, method="nnet")
## # weights: 6
## initial value 73.902614
## final value 27.985637
## converged
...
## # weights: 26
## initial value 71.559886
## iter 10 value 24.090531
## iter 50 value 24.074771
## iter 80 value 24.073714
## final value 24.073709
## converged
```

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

Predição usando a base de teste.

```
predicoes.rfVol <- predict(rfVol, testeVol)
predicoes.svmVol <- predict(svmVol, testeVol)
predicoes.rnaVol <- predict(rnaVol, testeVol)
predicoes.alomVol <- predict(alomVol, testeVol)
```

Função para calcular o Coeficiente de determinação: R2.

Quanto mais perto de 1 melhor é o modelo.

```
R2 <- function(X, Y) { 1 - (sum((X-Y)^2)/sum((X - mean(X))^2)) }
```

Função para calcular o Erro padrão da estimativa: Syx.

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

```
Syx <- function (X, Y, n) {sqrt(sum((X - Y)^2)/(n-2))}
```

Função para calcular Erro padrão da estimativa como porcentagem: Syx%.

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

```
SyxPerc <- function (X, Y) {(Syx(X, Y, length(X))/mean(X))*100}
```

Avaliação dos resultados usando R^2.

```
R2.rfVol <- R2(testeVol$VOL, predicoes.rfVol)
R2.svmVol <- R2(testeVol$VOL, predicoes.svmVol)
R2.rnaVol <- R2(testeVol$VOL, predicoes.rnaVol)
R2.alomVol <- R2(testeVol$VOL, predicoes.alomVol)
```

Avaliação dos resultados usando Syx.

```
Syx.rfVol <- Syx(dadosVolume$VOL, predicoes.rfVol, length(dadosVolume$VOL))
Syx.svmVol <- Syx(dadosVolume$VOL, predicoes.svmVol,
length(dadosVolume$VOL))
Syx.rnaVol <- Syx(dadosVolume$VOL, predicoes.rnaVol,
length(dadosVolume$VOL))
Syx.alomVol <- Syx(dadosVolume$VOL, predicoes.alomVol,
length(dadosVolume$VOL))
```

Avaliação dos resultados usando Syx%.

```
SyxPerc.rfVol <- SyxPerc(dadosVolume$VOL, predicoes.rfVol)
SyxPerc.svmVol <- SyxPerc(dadosVolume$VOL, predicoes.svmVol)
SyxPerc.rnaVol <- SyxPerc(dadosVolume$VOL, predicoes.rnaVol)
SyxPerc.alomVol <- SyxPerc(dadosVolume$VOL, predicoes.alomVol)
```

Impressão dos resultados: R².

```
str(R2.rfVol)
## num 0.854
str(R2.svmVol)
## num 0.848
str(R2.rnaVol)
## num -0.724
str(R2.alomVol)
## num 0.836
```

Impressão dos resultados: Syx.

```
str(Syx.rfVol)
## num 0.531
str(Syx.svmVol)
## num 0.511
str(Syx.rnaVol)
## num 0.548
str(Syx.alomVol)
## num 0.54
```

Impressão dos resultados: Syx%.

```
31
str(SyxPerc.rfVol)
## num 39.1
str(SyxPerc.svmVol)
## num 37.6
str(SyxPerc.rnaVol)
## num 40.3
str(SyxPerc.alomVol)
## num 39.8
```

Conclusão.

O melhor modelo utilizando as funções de R2 (onde aquele que estiver mais perto de 1), Erro padrão da estimativa (onde aquele que estiver mais próximo de 0) e a métrica do percentual do erro (onde aquele que estiver mais próximo de 0), foi o modelo SVM (Support Vector Machine) depois de realizarmos os treinos e os testes, aplicando assim as funções utilizando os valores observados e os valores preditos.

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

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

B – RESOLUÇÃO

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
options(repos = c(CRAN = "https://cloud.r-project.org/"))
install.packages('tinytex')
tinytex::install_tinytex(force = TRUE)
```
```

```
# 1 - Gráficos e tabelas.
```

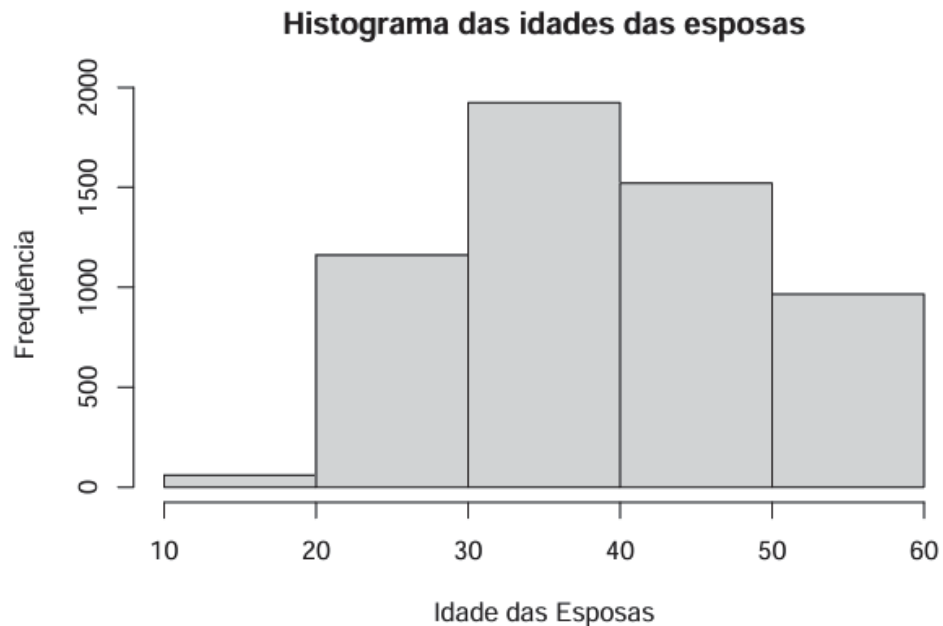
Elaborar os gráficos box-plot e histograma das variáveis "age" (idade da esposa) e "husage" (idade do marido) e comparar os resultados.

```
```{r cars0}
Carregando a base de dados "salarios"
load("salarios.RData")

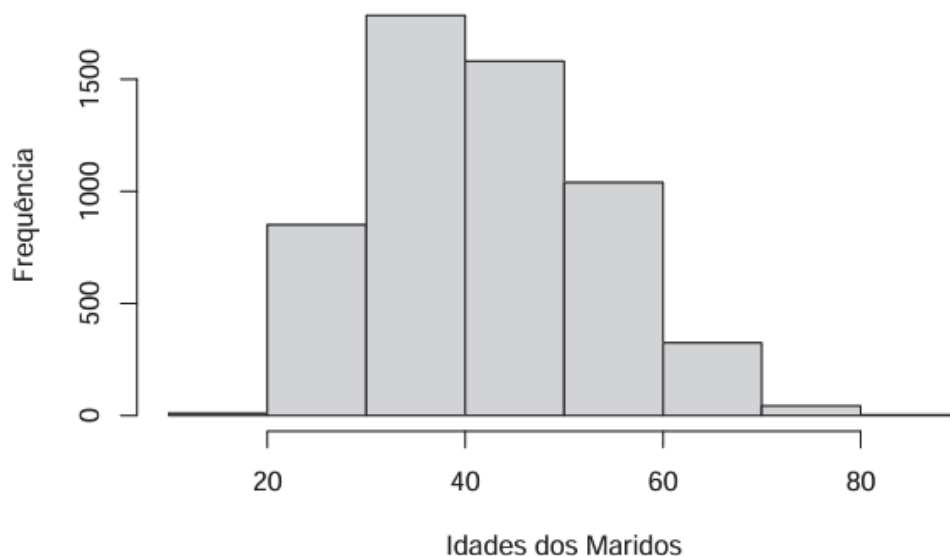
visualizando algumas estatisticas e informacoes das variaveis
View(salarios)
```

## Histogramas.

```{r cars1}
Desenho do histograma das variáveis "age" (idade das esposas)
e "husage" (idade dos maridos)
hAge = hist(salarios$age, main = "Histograma das idades das esposas",
 xlab="Idade das Esposas", ylab = "Frequência", breaks = 5)
```
```



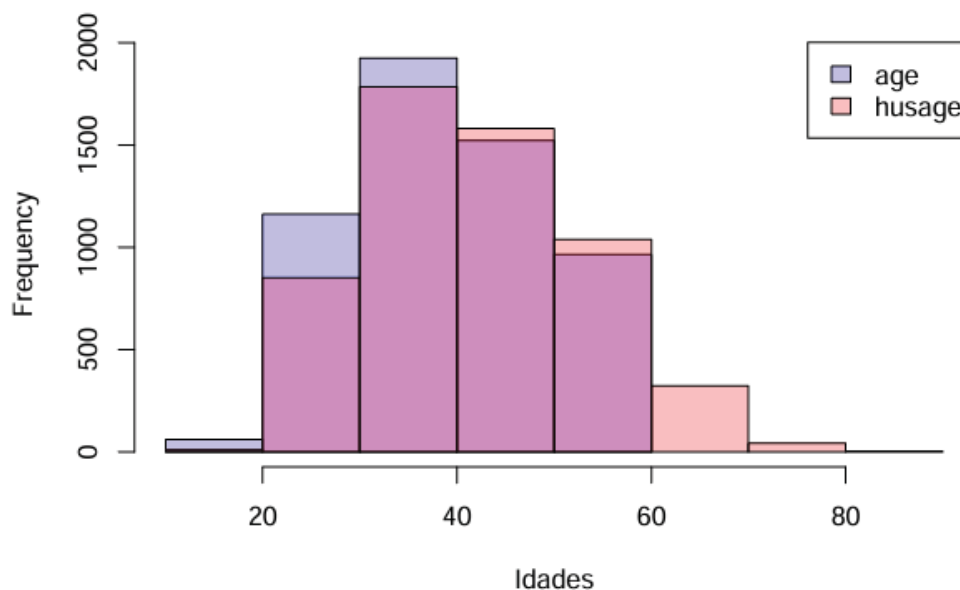
```
```{r cars2}
Histograma para 5 quebras com a mesma variavel
hHusage = hist(salarios$husage, main = "Histograma das idades dos maridos",
 xlab = "Idades dos Maridos", ylab = "Frequência", breaks = 5)
```
```

Histograma das idades dos maridos

```

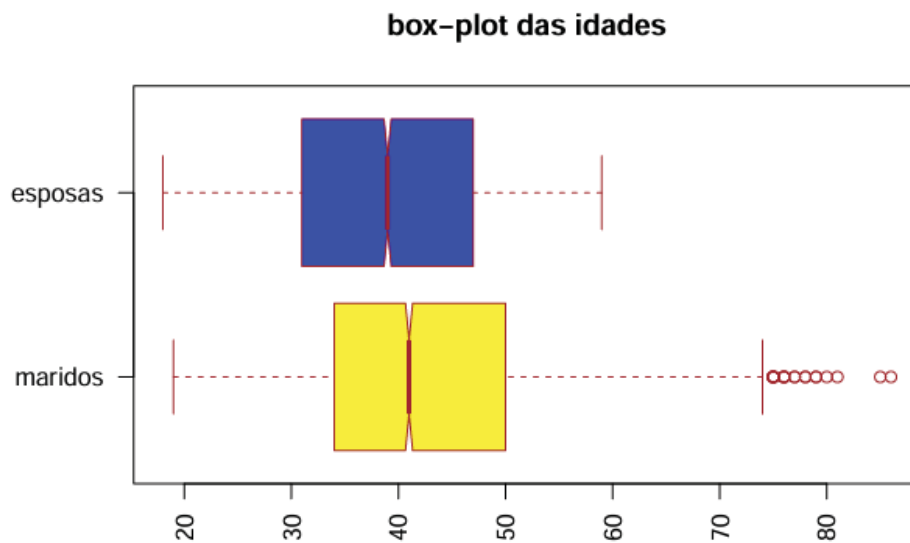
```{r cars3}
Plotando os dois histograma para a comparação entre as idades das esposas
e dos maridos
plot(hAge, col=rgb(0,0,1,1/4),
 main = "Histogramas",
 xlab = "Idades",
 ylim = c(min(min(hAge$counts), min(hHusage$counts)),
 max(max(hAge$counts, max(hHusage$counts))))),
 xlim=c(min(min(hAge$breaks), min(hHusage$breaks)),
 max(max(hAge$breaks, max(hHusage$breaks))))))
plot(hHusage, col=rgb(1,0,0,1/4), add=T)
legend("topright",
 c("age", "husage"),
 fill=c(rgb(0,0,1,1/4), rgb(1,0,0,1/4))
)
```

```

Histogramas

```
## box-plot.

```{r cars4}
Box-plot da variavel "age" e "husage"
boxplot(salarios$husage, salarios$age,
 main = "box-plot das idades",
 at = c(1,2),
 names = c("maridos", "esposas"),
 las = 2,
 col = c("yellow", "blue"),
 border = "brown",
 horizontal = TRUE,
 notch = TRUE)
```
```



```
## Tabela de Distribuicao de frequência.

```{r cars5}
Instalando pacote que gera a distribuicao de frequencia
install.packages("fdth")
library(fdth)
```

```{r cars6}
Calculando a distribuicao de frequencia e guardando no objeto
"table"
tableAge <- fdt(salarios$age,
 start=floor(min(salarios$age)),
 end=floor(max(salarios$age) + 1))

tableHusAge <- fdt(salarios$husage,
 start=floor(min(salarios$husage)-1),
 end=floor(max(salarios$husage) + 1))
```

```{r cars7}
Mostrando a distribuicao de frequencia
print (tableAge)
print (tableHusAge)
```
```

```

Class limits      f    rf rf(%)    cf cf(%)
[18,24)    306 0.05  5.43   306  5.43
[24,31)    917 0.16 16.28  1223 21.71
[31,37)   1371 0.24 24.33  2594 46.04
[37,44)   1069 0.19 18.97  3663 65.02
[44,50)   1006 0.18 17.86  4669 82.87
[50,57)    662 0.12 11.75  5331 94.62
> print (tableHusAge)
Class limits      f    rf rf(%)    cf cf(%)
[18,26)    347 0.06  6.16   347  6.16
[26,35)   1226 0.22 21.76  1573 27.92
[35,43)   1438 0.26 25.52  3011 53.44
[43,51)   1328 0.24 23.57  4339 77.01
[51,60)    846 0.15 15.02  5185 92.03
[60,68)    369 0.07  6.55  5554 98.58
[68,76)     70 0.01  1.24  5624 99.82
[76,84)     8 0.00  0.14  5632 99.96

```

Conclusão

Percebe-se que a mediana entre as idades das esposas e maridos estão bem próximas. A Maior frequência entre as idades está entre os 18 e 60 anos.

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.

Cálculo da média.

```

```{r cars8}
Calculando a média das variaveis age e husage
mediaAge <- mean(salarios$age)
mediaHusAge <- mean(salarios$husage)

Mostrando as médias das variaveis age e husage
print(mediaAge)
print(mediaHusAge)
```

[1] 39.42758
[1] 42.45296

```

A media da idade das esposas na amostra é 39.42758.
A media da idade dos maridos na amostra é 42.45296.

```

```{r cars10}
se dividirmos a idade média das esposas com dos maridos temos:
mediaHusAge/mediaAge
```

[1] 1.076733

```

Portanto, a idade média dos maridos são 7,67% maior que das esposas.

```
## Cálculo da médiana.
```

```
```{r cars11}
Calculando a médiana das variaveis age e husage
medianaAge <- median(salarios$age)
medianaHusAge <- median(salarios$husage)

Mostrando as médias das variaveis age e husage
print(medianaAge)
print(medianaHusAge)
```
```

```
[1] 39
```

```
[1] 41
```

A mediana da idade das esposas é de 39.

A mediana dos maridos é de 41.

```
```{r cars12}
Se dividirmos a mediana das idades dos maridos pelo das
esposas temos:
((medianaHusAge/medianaAge)-1)*100
```
```

```
[1] 5.128205
```

Em termos de mediana a idade dos maridos é 5,128 % maior que das esposas.

```
## Cálculo da moda.
```

```
```{r cars13}
Calculando a moda das variaveis age e husage
modaAge <- table(salarios$age)
subset(table(salarios$age),
 table(salarios$age) == max(table(salarios$age)))
```

```
modaHusAge <- table(salarios$husage)
subset(table(salarios$husage),
 table(salarios$husage) ==
 max(table(salarios$husage)))
```

```
Mostrando as modas das variaveis age e husage
print(modaAge)
print(modaHusAge)
```
```

```
37
```

```
217
```

```
44
```

```
201
```

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

```
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45 46 47 48
```

```

 5   6  25  31  35  71  59 115 113 108 137 157 172 174 169 196 161 174 195
169 200 175 192 172 178 201 149 149 130 147
 49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67
68  69  70  71  72  73  74  75  76  77  78
135 127 112 128 102 104 126 103 106  86  91  81  74  62  45  45  30  18  14
14  13   8   7   9   6   2   7   4   2   2
 79  80  81  85  86
  2   1   1   1   1

```

A moda da idade dos maridos é de 44 anos, com 201 pessoas.

A moda da idade das esposas é de 37 anos, com 217 pessoas.

```

```{r cars14}
Portanto, a moda da idade dos maridos eh maior que a das
esposas:
((44/37)-1)*100
eh aproximadamente 18,92% maior.
```

```

```
[1] 18.91892
```

Portanto, a moda da idade dos maridos eh maior que a das esposas: é aproximadamente 18,92% maior.

Cálculo da variância.

```

```{r cars15}
Calculando a variancia das variaveis age e husage
varianciaAge <- var(salarios$age)
varianciaHusAge <- var(salarios$husage)

Mostrando as variancias das variaveis age e husage
print(varianciaAge)
print(varianciaHusAge)
```

```

```
[1] 99.75234
```

```
[1] 126.0717
```

Portanto, a variancia das idades dos maridos é 26,38 % maior que a variância das idades das esposas.

Cálculo do desvio padrão.

```

```{r cars17}
Calculando o desvio padrão das variaveis age e husage
desvioAge <- sd(salarios$age)
desvioHusAge <- sd(salarios$husage)

Mostrando os desvios padrões das variaveis age e husage
print(desvioAge)
print(desvioHusAge)
```

```

```
[1] 9.98761
```

```
[1] 11.22817
```

O desvio padrao das idades das esposas é 9.98761.

O desvio padrao das idades dos maridos é 11.22817.

```
```{r cars18}
Se dividirmos o desvio padrao das idades dos maridos pelo
das esposas:
((desvioHusAge/desvioAge)-1)*100
```
```

```
[1] 12.42096
```

O desvio padrao das idades dos maridos é mais que 12,42 % superior ao das esposas.

Cálculo do coeficiente de variação.

```
```{r cars19}
Calculando o coeficiente de variacao das variaveis age e husage
coeficienteAge <- (desvioAge/mediaAge)*100
coeficienteHusAge <- (desvioHusAge/mediaHusAge)*100
```

```
Mostrando os coeficientes de variacao das variáveis age e husage
print(coeficienteAge)
print(coeficienteHusAge)
```
```

```
[1] 25.33153
```

```
[1] 26.44849
```

coeficiente de variacao das idades das esposas é 25.33%.

O coeficiente de variacao das idades dos maridos é 26.45%.

Isso quer dizer que a idade dos maridos e esposas não variam muito na amostra.

3 - Testes paramétricos ou não paramétricos

Testes de hipoteses paramétricos e intervalos de confiança

Teste de duas amostras independentes - Unpaired Z test

```
```{r cars20}

install.packages("skimr")
library(skimr)
Carregando os pacotes:
library("BSDA")
library("onewaytests")
library("sjPlot")
#library("tigerstats")
library("misty")
library("stats")
library("carData")
library("datasets")
library("BSDA")
library("nortest")
library("stats")
library("rcompanion")
library("dplyr")
#library("tigerstats")
library("misty")
library("tsoutliers")
library("DescTools")
library("sjPlot")
```

```
library("tidyr")
```

```

Vamos fazer checagens preliminares para verificar as exigências do teste: Amostras independentes, normalidade e homogeneidade das variâncias entre grupos.

Premissa 1: As duas amostras são independentes?

Sim, pois o grupo Age não tem nada haver com o grupo HusAge não se trata de uma amostra ou grupos emparelhados.

Premissa 2: Os dados de cada amostra/grupo possuem distribuição normal?

Vamos usar o teste de normalidade kolmogorov-Smirnov e Jarque Berra com o seguinte teste de hipóteses:

- H0: os dados são normalmente distribuídos
- Ha: os dados não são normalmente distribuídos

Primeiro vamos fazer o teste de normalidade Kolmogorov-Smirnov para "age".

```
```{r cars21}
Executando teste de normalidade de Jarque-Bera
JarqueBeraTest(salarios$age, robust = TRUE)
```
```

Robust Jarque Bera Test

```
data: salarios$age
X-squared = 158.49, df = 2, p-value < 2.2e-16
```

```
p-value = 2.2e-16 -> p-value < 0.05
```

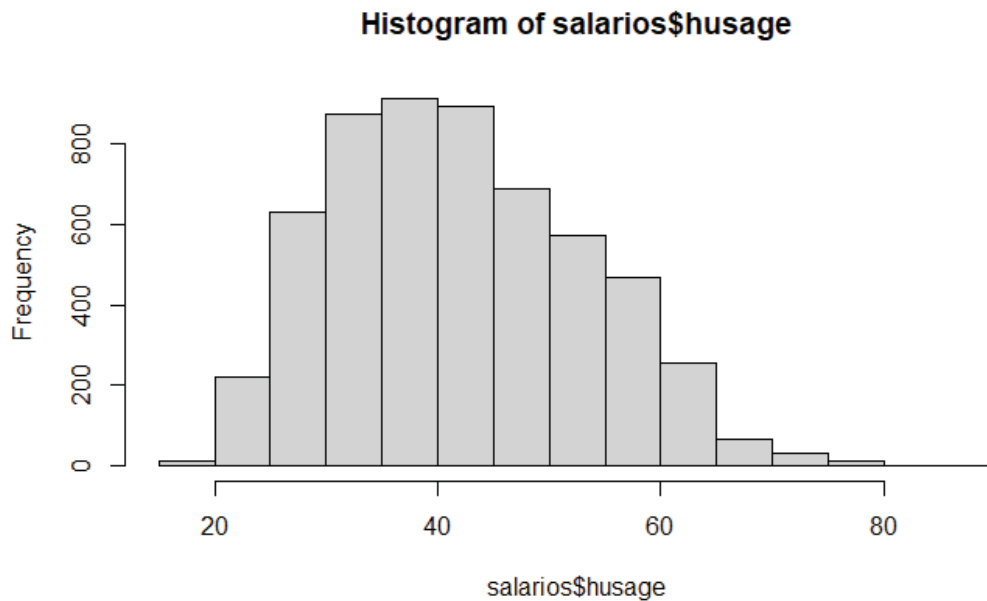
```
```{r cars22}
Executando teste de normalidade de Kolmogorov-Smirnov
lillie.test(salarios$age)
```
```

Lilliefors (Kolmogorov-Smirnov) normality test

```
data: salarios$age
D = 0.058909, p-value < 2.2e-16
```

```
p-value = 2.2e-16 -> p-value < 0.05
```

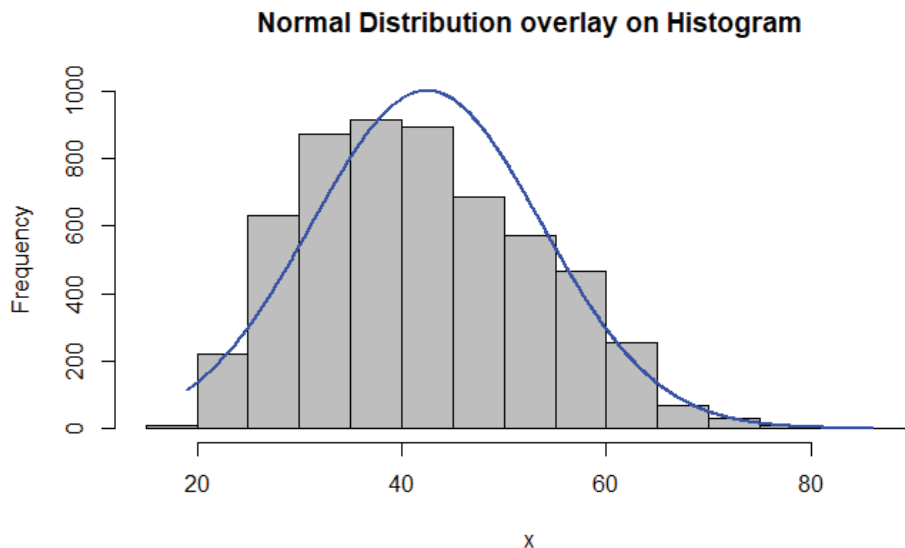
```
```{r cars23}
Vamos gerar o histograma:
hist(salarios$husage)
```
```



```

```{r cars24}
Vamos plotar a curva normal sobre o histograma:
plotNormalHistogram(salarios$husage, prob = FALSE,
 main = "Normal Distribution overlay on Histogram",
 length = 1000)
```

```



Conclusão: Rejeita-se a hipótese de que as duas amostras são normalmente distribuídas. Assume-se que a hipótese de que ambas as amostras não são normalmente distribuídas.

Premissa 3.

As duas populações/amostras/grupos possuem homogeneidade das variâncias?
O teste de hipóteses é:

- H_0 : As variâncias são estatisticamente iguais (homogêneas)
- H_A : As variâncias não são estatisticamente iguais (heterogêneas)

```

```{r cars25}

```

```

instalacao de PairedData
install.packages("PairedData")
library(PairedData)
```


```

```{r cars26}
# Usaremos o teste F para testar a homogeneidade nas variancias.
Var.test(salarios$age, salarios$husage, data = salarios, conf.level = 0.95,
         alternative = c("two.sided"), paired=FALSE)
```

F test to compare two variances

data: x and y
F = 0.79123, num df = 5633, denom df = 5633, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.7509664 0.8336625
sample estimates:
ratio of variances
 0.7912349

p-value = 2.2e-16 -\> p-value \< 0.05.

```{r cars27}
# Obtendo o valor tabelado da distribuicao F
qf(0.95, 5633, 5633)
```

[1] 1.04481

Temos F=1.04481.

```{r cars28}
# para a outra cauda temos:
1/1.04481
```

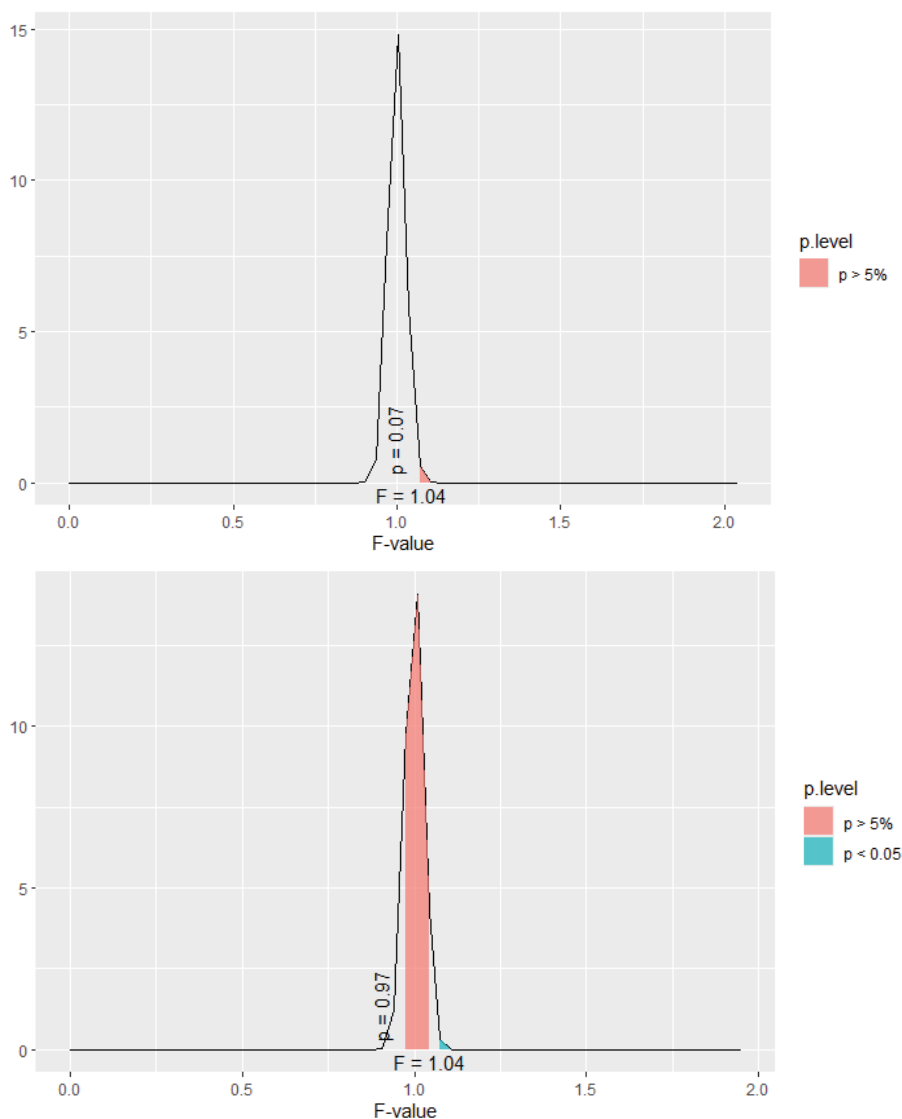
[1] 0.9571118

F = 0.9571118.

```{r cars29}
# Vamos construir o grafico:
dist_f(f = 1.04, deg.f1 = 5633, deg.f2 = 5633)
dist_f(f = 0.95, deg.f1 = 5633, deg.f2 = 5633)
```

```


```



O teste de F tem valor crítico entre 0.95 e 1.04 (região de não rejeição de H_0), os valores acima de 1,04 e abaixo de 0.95 estão na região de rejeição de H_0 (área azul do gráfico).

O valor da estatística F calculada é 0.7912349.

Como esse valor se encontra na região de não rejeição de H_0 , então rejeitamos a hipótese de que as variâncias são estatisticamente iguais.

Observe que se as observacoes são independentes trata-se de uma amostra emparelhada.

Se os dados não forem normalmente distribuídos, e as variâncias não forem homogêneas é recomendável usar o teste de duas amostras independentes não paramétrico.

```
## Testes não Paramétricos, Mann-Whitney "U" test
```

```
` `{r cars30}
#instalação dos pacotes
install.packages("dplyr")
install.packages("ggpubr")
```

```
# Carregando os pacotes
library("dplyr")
```

```

library("ggpubr")
```

```{r cars31}
# Vamos calcular um sumario estatistico
summary(salarios$age)
summary(salarios$husage)
```

 Min. 1st Qu. Median Mean 3rd Qu. Max.
 18.00 31.00 39.00 39.43 47.00 59.00
 Min. 1st Qu. Median Mean 3rd Qu. Max.
 19.00 34.00 41.00 42.45 50.00 86.00

```{r cars32}
# Pegar apenas os dados de idades das esposas (age) e maridos (husage)
# com que vamos trabalhar para realizar os testes
dadosAge <- salarios[c(1,10)]
```

```{r cars33}
# Vamos empilhar os dados para fazer tudo em um comando
age_husage_emp <- dadosAge %>%
  pivot_longer(1:2, names_to = "tipo_rel",
               values_to = "idades")
```

```{r cars34}
# Ordenando o dataframe
age_husage_emp <- age_husage_emp[order(age_husage_emp$tipo_rel),]
head(age_husage_emp)
```

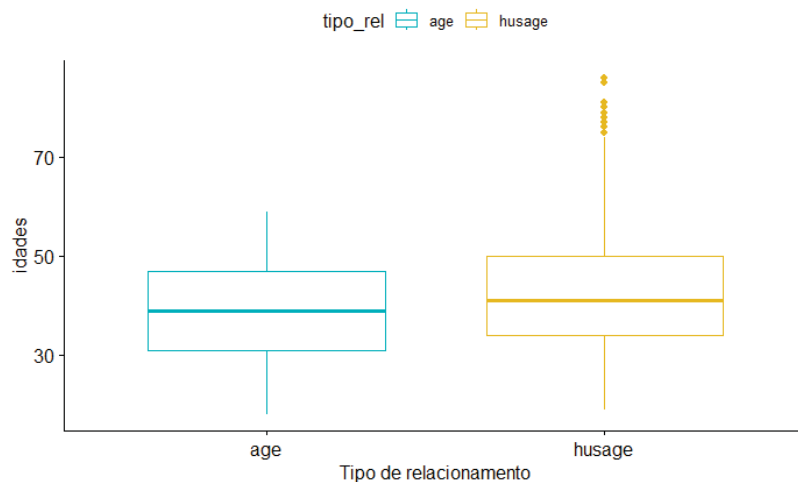

A tibble: 6 x 2

tipo_rel <chr>	idades <dbl>
age	43
age	26
age	49
age	35
age	43
age	58

6 rows


```{r cars35}
# Vamos visualizar os dados usando box-plots
# Plotaremos o "weight" por grupo
ggboxplot(age_husage_emp, x = "tipo_rel", y = "idades",
           color = "tipo_rel", palette=c("#00AFBB", "#E7B800"),
           ylab = "idades", xlab = "Tipo de relacionamento")
```

```



Temos valores extremos, mas não há problema

Vamos calcular o One-sample Wilcoxon Signed Rank Test (teste bicaudal).  
As hipóteses do teste são:

- $H_0$ : idade mediana das esposas é estatisticamente igual a idade mediana dos maridos
- $H_A$ : idade mediana das esposas não é estatisticamente igual a idade mediana dos maridos

```
```{r carsx}
# Vamos calcular o teste
wilcox.test(idades ~ tipo_rel, data = age_husage_emp,
             conf.int=TRUE)
```
```

Wilcoxon rank sum test with continuity correction

```
data: idades by tipo_rel
W = 13619912, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -3.000024 -2.000033
sample estimates:
difference in location
 -2.999966
```

p-value = 2.2e-16 -> p-value < 0.05.

O p-value do teste é 2.2e-16, que é menor que o nível de significância  $\alpha = 0,05$ .

Podemos concluir que as idades são significativamente diferentes entre os maridos e as esposas.

O intervalo de confiança da diferença das medianas está entre -3.000024 e -2.000033 com uma mediana de -2.999966.

Portanto, existe diferença estatística entre os valores das medianas.

```
```{r cars36}
median(salarios$husage)/median(salarios$age)
```
```

```
[1] 1.051282
```

## Conclusão

A diferença entre as medianas é de que os maridos têm idade mediana 5.12% maior do que a mediana idade das esposas, e que essa diferença é estatisticamente diferente.

## APÊNDICE E - ESTATÍSTICA APLICADA II

### A – ENUNCIADO

#### Regressões Ridge, Lasso e ElasticNet

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

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

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

### B – RESOLUÇÃO

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

```
install.packages("glmnet")
```

```
#Carregando os pacotes
```

```
library(plyr)
```

```
library(readr)
```

```
library(dplyr)
```

##

```
Anexando pacote: 'dplyr'
```

```
Os seguintes objetos são mascarados por 'package:plyr':
```

##

```
arrange, count, desc, failwith, id, mutate, rename, summarise,
```

```
summarize
```

```
Os seguintes objetos são mascarados por 'package:stats':
```

##

```
filter, lag
```

```
Os seguintes objetos são mascarados por 'package:base':
```

##

```
intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
library(repr)
```

```
library(glmnet)
```

```
Carregando pacotes exigidos: Matrix
```

```
Loaded glmnet 4.1-10
```

```
library(caret)
```

```
Carregando pacotes exigidos: lattice
```

Abrindo o dataset

```
load("trabalhosalarios.RData")
```

Vamos guardar este dataset em um objeto chamado "dataTrabalho"

```
dataTrabalho <- trabalhosalarios
```

```
Vamos excluir a coluna earns
```

```
dataTrabalho$earns = NULL
```

```
Vamos visualizar parte do dataset
```

```
glimpse (dataTrabalho)
```

```
Rows: 2,574
```

```
Columns: 16
```

```
$ husage <dbl> 56, 31, 33, 34, 42, 45, 33, 31, 31, 44, 45, 22, 66, 43,
26, 3...
```

```
$ husunion <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0...
```

```
$ husearns <dbl> 1500, 800, 950, 1000, 730, 1154, 1350, 769, 340, 750, 1
200, 2...
```

```
$ huseduc <dbl> 14, 17, 13, 14, 14, 16, 16, 18, 12, 12, 12, 12, 16, 12,
14, 1...
```

[illegible][illegible]

```
$ hushrs <dbl> 40, 40, 60, 50, 40, 38, 40, 55, 40, 40, 50, 40, 40, 50,
36, 5...
```

```
$ kidge6 <dbl> 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0...
```

```
$ age <dbl> 49, 29, 30, 31, 41, 45, 32, 27, 30, 42, 42, 23, 55, 31,
27, 3...
```

```
$ black <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```
$ educ <dbl> 12, 14, 12, 12, 12, 18, 12, 14, 15, 12, 12, 13, 12, 12,
14, 1...
```

```
$ hispanic <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```
$ union <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1
```

```
$ exper <dbl> 31, 9, 12, 13, 23, 21, 14, 7, 9, 24, 24, 4, 37, 13, 7,
12, 31...
$ kidlt6 <dbl> 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1
, 0, 1...
$ lwage <dbl> 1.897120, 2.484907, 2.431418, 1.629241, 2.302585, 2.496
741, 2...
Vamos jogar uma semente para gerar numeros aleatórios.
Essa semente de numeros aleatorios serve para
particionar o dataset aleatoriamente
set.seed(302)
Particionando o dataset para Treinamento e Teste

Criando o index para os 80% de treinamento
index = sample(1:nrow(dataTrabalho), 0.8*nrow(dataTrabalho))
Vamos criar a base de dados de treinamento
train = dataTrabalho[index,]
Vamos criar a base de dados de teste com os outros 20%
test = dataTrabalho[-index,]
Vamos checar as dimensoes das bases de treinamento e
teste
dim(train)
[1] 2059 16
dim(test)
[1] 515 16
A base de treinamento possui 2059 linhas (familias) e
16 colunas (variaveis)
A base de teste possui 515 linhas (familias) e 16
colunas (variaveis)
Padronização das variáveis

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

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

Vamos ver o sumario estatistico das variaveis
padronizadas de cada dataset
summary(train)
husage husunion husearns huseduc
Min. :-2.1039 Min. :0.0000 Min. : -1.7145 Min. : -5.0191
1st Qu. :-0.8087 1st Qu. :0.0000 1st Qu. : -0.6615 1st Qu. : -0.5568
Median :-0.1113 Median :0.0000 Median : -0.2166 Median : -0.1849
Mean : 0.0000 Mean :0.2205 Mean : 0.0000 Mean : 0.0000
3rd Qu. : 0.6857 3rd Qu. :0.0000 3rd Qu. : 0.4212 3rd Qu. : 0.9306
Max. : 2.8775 Max. :1.0000 Max. : 3.9006 Max. : 1.6744
husblk hushisp hushrs kidge6
Min. :0.00000 Min. :0.00000 Min. : -3.3291 Min. :0.0000
1st Qu. :0.00000 1st Qu. :0.00000 1st Qu. : -0.1961 1st Qu. :0.0000
Median :0.00000 Median :0.00000 Median : -0.1961 Median :0.0000
Mean :0.06217 Mean :0.04905 Mean : 0.0000 Mean :0.3439
3rd Qu. :0.00000 3rd Qu. :0.00000 3rd Qu. : 0.5871 3rd Qu. :1.0000
Max. :1.00000 Max. :1.00000 Max. : 4.4250 Max. :1.0000
age black educ hispanic
Min. :-2.12418 Min. :0.00000 Min. : -5.6264 Min. :0.00000
1st Qu. : -0.83944 1st Qu. :0.00000 1st Qu. : -0.6265 1st Qu. :0.00000
```

```

Median :-0.09001 Median :0.00000 Median :-0.2098 Median :0.00000
Mean : 0.00000 Mean :0.06022 Mean : 0.0000 Mean :0.05148
3rd Qu.: 0.65943 3rd Qu.:0.00000 3rd Qu.: 1.0401 3rd Qu.:0.00000
Max. : 2.26536 Max. :1.00000 Max. : 1.8734 Max. :1.00000
union exper kidlt6 lwage
Min. :0.0000 Min. : -1.89545 Min. :0.0000 Min. : -4.78007
1st Qu.:0.0000 1st Qu.: -0.86181 1st Qu.:0.0000 1st Qu.: -0.69165
Median :0.0000 Median : -0.03489 Median :0.0000 Median : -0.05973
Mean :0.1481 Mean : 0.00000 Mean :0.2545 Mean : 0.00000
3rd Qu.:0.0000 3rd Qu.: 0.68866 3rd Qu.:1.0000 3rd Qu.: 0.65188
Max. :1.0000 Max. : 2.65259 Max. :1.0000 Max. : 4.11492
summary(test)
husage husunion husearns huseduc
Min. : -1.80501 Min. :0.00000 Min. : -1.72343 Min. : -5.0191
1st Qu.: -0.70910 1st Qu.:0.00000 1st Qu.: -0.72231 1st Qu.: -0.5568
Median : -0.01171 Median :0.00000 Median : -0.20173 Median : -0.1849
Mean : 0.05078 Mean :0.2272 Mean : -0.03584 Mean : -0.0167
3rd Qu.: 0.68568 3rd Qu.:0.00000 3rd Qu.: 0.42118 3rd Qu.: 0.9306
Max. : 2.47898 Max. :1.0000 Max. : 3.90062 Max. : 1.6744
husblck hushisp hushrs kidge6
Min. :0.00000 Min. :0.00000 Min. : -3.32913 Min. :0.000
1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.: -0.19614 1st Qu.:0.000
Median :0.00000 Median :0.00000 Median : -0.19614 Median :0.000
Mean :0.07961 Mean :0.06214 Mean : -0.02625 Mean :0.365
3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.: 0.46963 3rd Qu.:1.000
Max. :1.00000 Max. :1.00000 Max. : 4.42503 Max. :1.000
age black educ hispanic
Min. : -1.91006 Min. :0.00000 Min. : -5.62637 Min. :0.00000
1st Qu.: -0.73238 1st Qu.:0.00000 1st Qu.: -0.62650 1st Qu.:0.00000
Median : 0.01706 Median :0.00000 Median : -0.62650 Median :0.00000
Mean : 0.06404 Mean :0.07767 Mean : -0.09496 Mean :0.07184
3rd Qu.: 0.76649 3rd Qu.:0.00000 3rd Qu.: 0.62347 3rd Qu.:0.00000
Max. : 2.26536 Max. :1.00000 Max. : 1.87343 Max. :1.00000
union exper kidlt6 lwage
Min. :0.0000 Min. : -1.79209 Min. :0.0000 Min. : -11.060109
1st Qu.:0.0000 1st Qu.: -0.65508 1st Qu.:0.0000 1st Qu.: -0.635121
Median :0.0000 Median : -0.03489 Median :0.0000 Median : -0.041678
Mean :0.1379 Mean : 0.08533 Mean :0.2544 Mean : 0.008121
3rd Qu.:0.0000 3rd Qu.: 0.74034 3rd Qu.:1.0000 3rd Qu.: 0.651875
Max. :1.0000 Max. : 2.44586 Max. :1.0000 Max. : 4.027740
Vamos criar um objeto com as variaveis que usaremos no
modelo
cols_reg = c('husage', 'husunion', 'husearns', 'huseduc', 'husblck',
 'hushisp', 'hushrs', 'kidge6', 'age', 'black',
 'educ', 'hispanic', 'union', 'exper', 'kidlt6', 'lwage')

Vamos gerar variaveis dummies para organizar os datasets
em objetos tipo matriz
Estamos interessados em estimar o salario-hora (lwage)
dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+husblck+
 hushisp+hushrs+kidge6+age+black+
 educ+hispanic+union+exper+kidlt6,
 data = dataTrabalho[,cols_reg])
train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])
print(dim(train_dummies)); print(dim(test_dummies))
[1] 2059 15
[1] 515 15
Vamos guardar a matriz de dados de treinamento das
variaveis explicativas para o modelo em um objeto
chamado "x"

```

```

x = as.matrix(train_dummies)

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

Vamos guardar a matriz de dados de teste das variaveis
explicativas para o modelo em um objeto chamado
"x_test"
x_test = as.matrix(test_dummies)

Vamos guardar o vetor de dados de teste da variavel
dependente para o modelo em um objeto chamado "y_test"
y_test = test$lwage
Calculo do R^2 e RMSE para usar nas Regressões Ridge, Lasso e ElasticNet

Vamos calcular o R^2 dos valores verdadeiros e
preditos conforme a seguinte funcao:
eval_results <- function(true, predicted, df) {
 SSE <- sum((predicted - true)^2)
 SST <- sum((true - mean(true))^2)
 R_square <- 1 - SSE / SST
 RMSE = sqrt(SSE/nrow(df))
As metricas de performace do modelo:
 data.frame(
 RMSE = RMSE,
 Rsquare = R_square
)
}

```

Exemplo para a predição

```

Como os valores do dataset sao padronizados, nos temos
de padronizar tambem os dados que vamos fazer a predicao
Note que as variaveis dummies nao devem ser padronizadas

Vamos fazer uma predicao para:
husage = 40 anos (idade do marido)
husage = (40-pre_proc_val[["mean"]][["husage"]])/
 pre_proc_val[["std"]][["husage"]]

husunion = 0 (marido não possui união estável)
husunion = 0

husearns = 600 (US$ renda do marido por semana)
husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
 pre_proc_val[["std"]][["husearns"]]

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

husblck = 1 (o marido é preto)
husblck = 1

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

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

```

```

kidge6 = 1 (possui filhos maiores de 6 anos)
kidge6 = 1

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

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

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

hispanic = 1 (esposa é hispanica)
hispanic = 1

union = 0 (esposa não possui união estável)
union = 0

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

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

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

our_pred
husage husunion husearns huseduc husblk hushisp hush
[1,] -0.0117095 0 -0.02375752 -0.1849375 1 0 -0.19613
53
kidge6 age black educ hispanic union exper kidlt
6
[1,] 1 0.01705504 0 -0.2098454 1 0 -0.03488992
1

Regressão RIDGE

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

```

```

best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge
[1] 0.03162278
Estimando o modelo Ridge

ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
 family = 'gaussian',
 lambda = best_lambda_ridge)

Vamos ver o resultado (valores) da estimativa
(coeficientes)
ridge_reg[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"
s0
husage -0.004745469
husunion 0.001301385
husearns 0.230241265
huseduc 0.051370292
husblck 0.217056671
hushisp 0.086144980
hushrs -0.069507967
kidge6 -0.181910965
age 0.069266121
black -0.277110253
educ 0.327096706
hispanic -0.011406010
union 0.365669412
exper -0.016268996
kidlt6 -0.042460770
Calculando o exemplo com a Regressão Ridge

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

As metricas da base de treinamento sao:
eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.8411446 0.292132
Predicao e avaliacao nos dados de teste:
predictions_test <- predict(ridge_reg,
 s = best_lambda_ridge,
 newx = x_test)

As metricas da base de teste sao:
eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.9893328 0.259084
Se compararmos as metricas de treinamento e teste
vemos que o tamanho dos erros de treinamento e teste
e os R^2 sao muito parecidos, o que descarta a hipotese
de overfitting e underfitting.

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

O resultado da predicao é:
predict_our_ridge
s=0.03162278
[1,] -0.06843352
O resultado é um valor padronizado, vamos converte-lo
para o valor nominal, consistente com o dataset original

```

```

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

O resultado é:
lwage_pred_ridge
s=0.03162278
[1,] 2.161212
Este é o valor predito do salario por hora (US$ 2.16),
segundo as características que atribuímos

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

Os valores sao:
CIlwr_ridge
s=0.03162278
[1,] 2.139354
CIupr_ridge
s=0.03162278
[1,] 2.18307
Entao, segundo as características que atribuímos o
salario-hora da esposa é em media US$2.16 e pode
variar entre US$2.14 e US$2.18

#####
Regressão LASSO

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

Vamos guardar o lambda "otimo" em um objeto chamado
best_lambda_lasso
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso
[1] 0.01258925
Estimando o modelo LASSO

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

Vamos visualizar os coeficientes estimados
lasso_model[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"
##
s0
husage .
husunion .
husearns 0.23002775
huseduc 0.03010233
husblck .
hushisp .
hushrs -0.06025028

```

```
kidge6 -0.14252320
age 0.04689191
black -0.03169581
educ 0.33915744
hispanic .
union 0.34330443
exper .
kidlt6 .
Perceba que alguns coeficientes estao zerados pois
nao sao significativos
Calculando o exemplo com a Regressão LASSO

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

As metricas da base de treinamento sao:
eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.8423523 0.2900978
Vamos fazer as predicoes na base de teste
predictions_test <- predict(lasso_model,
 s = best_lambda_lasso,
 newx = x_test)

As metricas da base de teste sao:
eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.9899383 0.2581767
Perceba que os erros nao sao muito diferentes, assim
como os valores de R2.

Vamos fazer uma predicao para o nosso exemplo

Vamos fazer a predicao com base nos mesmos parametros
como fizemos anteriormente na regressao Ridge

Vamos para a predicao
predict_our_lasso <- predict(lasso_model,
 s = best_lambda_lasso,
 newx = our_pred)

predict_our_lasso
s=0.01258925
[1,] -0.2120462
Novamente, o resultado esta padronizado, nos temos de
converte-lo para valor compativel com o dataset original
wage_pred_lasso=(predict_our_lasso*
 pre_proc_val[["std"]][["lwage"]])+
 pre_proc_val[["mean"]][["lwage"]]
wage_pred_lasso
s=0.01258925
[1,] 2.088536
Portanto, o salario-hora da esposa é US$2.09

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

```

Cilwr_lasso <- m + (qnorm(0.025))*dam
CIupr_lasso <- m - (qnorm(0.025))*dam

O intervalo de confianca é:
Cilwr_lasso
s=0.01258925
[1,] 2.066677
CIupr_lasso
s=0.01258925
[1,] 2.110394
Entao, o salario medio é de US$2.09 e pode variar
entre US$2.07 e US$2.11

#####
Regressão ElasticNet

A regressao ElasticNet combina aspectos das regressoes
Ridge e Lasso. Ela penaliza o modelo usando ambas
funcoes de penalidade, l2-norm and l1-norm.
O modelo pode ser customizado pelo pacote "caret"
que encontra os valores otimos dos parametros
automaticamente.
Vamos configurar o treinamento do modelo por
cross validation, com 10 folders, 5 repeticoes
e busca aleatoria dos componentes das amostras
de treinamento, o "verboseIter" é soh para
mostrar o processamento.
train_cont <- trainControl(method = "repeatedcv",
 number = 10,
 repeats = 5,
 search = "random",
 verboseIter = FALSE)

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

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

Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trai
nInfo,
: There were missing values in resampled performance measures.
O melhor parametro alpha escolhido é:
elastic_reg$bestTune
alpha lambda
7 0.7429763 0.01233549
E os parametros sao:
elastic_reg[["finalModel"]][["beta"]]
15 x 77 sparse Matrix of class "dgCMatrix"
[[suppressing 77 column names 's0', 's1', 's2' ...]]
##
husage
husunion

```

```

husearns . . . 0.005265248 0.02487154 0.0429
6476
huseduc
husblck
hushisp
hushrs
kidge6
age
black
educ . 0.03507813 0.06778955 0.0982384 0.125145010 0.14617055 0.1655
8804
hispanic
union
exper
kidlt6
##
husage
husunion
husearns 0.05963892 0.0750015 0.08913756 0.1021342 0.1140741 0.12505948
huseduc
husblck
hushisp
hushrs
kidge6
age
black
educ 0.18350187 0.2000085 0.21520547 0.2291845 0.2420330 0.25318972
hispanic
union 0.01387615
exper
kidlt6
##
husage
husunion
husearns 0.13515562 0.14442264 0.15291828 0.1607027 0.1678319 0.17383344
6
huseduc 0.00205459
6
husblck
hushisp
hushrs
kidge6
age
black
educ 0.26257786 0.27117395 0.27904309 0.2862438 0.2928303 0.29781094
5
hispanic
union 0.04486632 0.07331676 0.09941872 0.1233520 0.1452849 0.16561237
3
exper
kidlt6
##
husage
husunion
husearns 0.179156492 0.1840082886 0.188394608 0.192895267 0.19701187
huseduc 0.004588311 0.0070668547 0.009691862 0.012127439 0.01426328
husblck
hushisp
hushrs -0.003737866 -0.009837365 -0.01541787
kidge6 . . -0.0070599317 -0.021563146 -0.034670629 -0.04664962
age . . 0.0009975204 0.006234541 0.010683673 0.01475270

```

```

black
educ 0.302025726 0.3057289826 0.309131553 0.312206014 0.31506292
hispanic
union 0.184302326 0.2014263463 0.216225946 0.229837789 0.24226937
exper
kidlt6
##
husage
husunion
husearns 0.20076772 0.20420079 0.20733811 0.21020443 0.21282254
huseduc 0.01627001 0.01809984 0.01976777 0.02128807 0.02267374
husblck
hushisp
hushrs -0.02051792 -0.02518001 -0.02944058 -0.03333320 -0.03688880
kidge6 -0.05761321 -0.06763611 -0.07679628 -0.08516577 -0.09281097
age 0.01846904 0.02186304 0.02496201 0.02779107 0.03037329
black
educ 0.31763607 0.31998310 0.32212393 0.32407648 0.32585714
hispanic
union 0.25363908 0.26402769 0.27351750 0.28218434 0.29009797
exper
kidlt6
##
husage
husunion
husearns 0.21521343 0.21739639 0.21938914 0.22120797 0.22286781
huseduc 0.02393668 0.02508771 0.02613673 0.02709276 0.02796402
husblck
hushisp
hushrs -0.04013585 -0.04310055 -0.04580695 -0.04827715 -0.05053142
kidge6 -0.09979302 -0.10616816 -0.11198806 -0.11730018 -0.12214810
age 0.03272985 0.03488016 0.03684202 0.03863175 0.04026426
black
educ 0.32748093 0.32896155 0.33031154 0.33154234 0.33266443
hispanic
union 0.29732250 0.30391684 0.30993501 0.31542658 0.32043698
exper
kidlt6
##
husage
husunion
husearns 0.224255907 0.22548974 0.22661144 0.22763380 0.22856610
huseduc 0.028776704 0.02942079 0.03008110 0.03068577 0.03123698
husblck
hushisp
hushrs -0.052597485 -0.05448836 -0.05621127 -0.05778263 -0.05921578
kidge6 -0.126558307 -0.13056251 -0.13422616 -0.13756844 -0.14061685
age 0.041736540 0.04307692 0.04429700 0.04540955 0.04642395
black -0.005667735 -0.01249777 -0.01873372 -0.02442181 -0.02960940
educ 0.333662297 0.33462474 0.33545846 0.33621681 0.33690793
hispanic
union 0.325283734 0.32977157 0.33387612 0.33762020 0.34103483
exper
kidlt6
##
husage
husunion
husearns 0.229398367 0.230110780 0.230763435 0.23135795 0.23189986
huseduc 0.031865943 0.032632851 0.033397497 0.03409776 0.03473649
husblck
hushisp 0.002352205 0.009404264 0.015816930 0.02166397 0.02699579

```

```

hushrs -0.060499328 -0.061583307 -0.062572081 -0.06347341 -0.06429510
kidge6 -0.143599570 -0.147492078 -0.151073549 -0.15434126 -0.15732181
age 0.047280877 0.047556884 0.047789802 0.04800120 0.04819327
black -0.034179442 -0.037894696 -0.041286264 -0.04437836 -0.04719700
educ 0.337577891 0.338327905 0.338968382 0.33955035 0.34008061
hispanic
union 0.344072673 0.346733717 0.349174360 0.35140012 0.35342937
exper
kidlt6 -0.000576432 -0.004424469 -0.007963405 -0.01119284 -0.01413909
##
husage
husunion
husearns 0.23239380 0.23284401 0.23325249 0.23362736 0.23396847
huseduc 0.03531895 0.03585007 0.03624586 0.03668321 0.03709243
husblck
hushisp 0.03185746 0.03629017 0.04033457 0.04402288 0.04738153
hushrs -0.06504416 -0.06572697 -0.06634821 -0.06691587 -0.06743309
kidge6 -0.16004016 -0.16251917 -0.16471714 -0.16678054 -0.16866489
age 0.04836784 0.04852653 0.04870829 0.04883768 0.04895398
black -0.04976625 -0.05210810 -0.05424275 -0.05618696 -0.05796000
educ 0.34056381 0.34100412 0.34146125 0.34182936 0.34215825
hispanic
union 0.35527935 0.35696582 0.35848302 0.35988388 0.36116276
exper
kidlt6 -0.01682666 -0.01927802 -0.02145192 -0.02349361 -0.02535776
##
husage
husunion
husearns 0.234302266 0.23457017 0.23481646 0.23503756 0.23524176
huseduc 0.037639075 0.03815882 0.03862705 0.03906081 0.03945097
husblck 0.007303927 0.03634914 0.06238414 0.08708612 0.10887828
hushisp 0.050566970 0.05362118 0.05640652 0.05894578 0.06126057
hushrs -0.067898611 -0.06831051 -0.06868618 -0.06902819 -0.06934013
kidge6 -0.170470098 -0.17207863 -0.17354374 -0.17488084 -0.17609841
age 0.049017331 0.04910144 0.04917858 0.04924774 0.04931151
black -0.066513011 -0.09563887 -0.12177370 -0.14652800 -0.16840523
educ 0.342347607 0.34251288 0.34266655 0.34280248 0.34292925
hispanic
union 0.362301820 0.36316857 0.36396097 0.36467762 0.36533518
exper
kidlt6 -0.027123558 -0.02867023 -0.03007946 -0.03136478 -0.03253573
##
husage
husunion
husearns 0.23542487 0.23559448 0.23574622 0.23588724 0.23601306
huseduc 0.03981273 0.04013708 0.04043848 0.04070772 0.04095864
husblck 0.12959517 0.14773623 0.16507035 0.18012435 0.19460156
hushisp 0.06337068 0.06529409 0.06704731 0.06864530 0.07010179
hushrs -0.06962409 -0.06988314 -0.07011891 -0.07033404 -0.07052980
kidge6 -0.17720961 -0.17822118 -0.17914447 -0.17998470 -0.18075174
age 0.04936864 0.04942149 0.04946873 0.04951258 0.04955168
black -0.18916479 -0.20738137 -0.22475090 -0.23987283 -0.25437975
educ 0.34304116 0.34314616 0.34323846 0.34332566 0.34340189
hispanic
union 0.36592954 0.36647567 0.36696875 0.36742255 0.36783170
exper
kidlt6 -0.03360370 -0.03457647 -0.03546370 -0.03627169 -0.03700869
##
husage . . -0.0002718303 -0.002169203 -0.003601924
husunion
husearns 0.23613047 0.23623487 0.2363815518 0.236542238 0.236674763

```

```

huseduc 0.04118166 0.04139036 0.0415539362 0.041689702 0.041814622
husblck 0.20704564 0.21910820 0.2293321929 0.239214612 0.248470115
hushisp 0.07142930 0.07263915 0.0737228316 0.074829439 0.075834295
hushrs -0.07070847 -0.07087100 -0.0710541509 -0.071224848 -0.071376313
kidge6 -0.18144949 -0.18208663 -0.1826100067 -0.183182894 -0.183702319
age 0.04958814 0.04962053 0.0498811006 0.051566606 0.052838906
black -0.26688601 -0.27897420 -0.2892453398 -0.299124208 -0.308363666
educ 0.34347458 0.34353765 0.3435962325 0.343643656 0.343694292
hispanic
union 0.36820898 0.36854862 0.3688584322 0.369113748 0.369346903
exper
kidlt6 -0.03767969 -0.03829183 -0.0388433473 -0.039559188 -0.040186500
##
husage -0.004724857 -0.005797203 -0.006676763 -0.00755068 -0.008376151
husunion
husearns 0.236799496 0.236910757 0.237015727 0.23710877 0.237192181
huseduc 0.041923936 0.042027973 0.042115344 0.04220108 0.042281219
husblck 0.256024205 0.263445339 0.269419740 0.27542215 0.281233716
hushisp 0.076729401 0.077550564 0.078288478 0.07896842 0.079591402
hushrs -0.071516128 -0.071642802 -0.071759508 -0.07186489 -0.071960529
kidge6 -0.184156487 -0.184576524 -0.184947458 -0.18529446 -0.185614017
age 0.053840314 0.054794949 0.055581068 0.05635959 0.057094089
black -0.315947779 -0.323371593 -0.329386272 -0.33539918 -0.341204613
educ 0.343742048 0.343783318 0.343824708 0.34385923 0.343889696
hispanic
union 0.369567839 0.369765277 0.369951109 0.37011634 0.370264333
exper
kidlt6 -0.040727541 -0.041229334 -0.041669220 -0.04208306 -0.042465099
##
husage -0.009139864 -0.009724835 -0.0103166868 -0.010890631 -0.0114345
99
husunion
husearns 0.237267369 0.237339660 0.2374039110 0.237468966 0.2375230
25
huseduc 0.042355579 0.042412216 0.0424715598 0.042515398 0.0425710
26
husblck 0.286740036 0.290830650 0.2949494138 0.298998414 0.3028589
25
hushisp 0.080160469 0.080668287 0.0811359618 0.082836791 0.0841475
38
hushrs -0.072047487 -0.072128124 -0.0722008445 -0.072267051 -0.0723273
11
kidge6 -0.185906674 -0.186159403 -0.1863978210 -0.186619094 -0.1868233
29
age 0.057773244 0.058297158 0.0588246338 0.059329412 0.0598095
73
black -0.346695921 -0.350816733 -0.3549446222 -0.359009493 -0.3628689
02
educ 0.343916759 0.343947560 0.3439711539 0.343981708 0.3439879
01
hispanic . . -0.0004344885 -0.001873699 -0.0029895
39
union 0.370397717 0.370525683 0.3706421244 0.370728442 0.3708095
34
exper
kidlt6 -0.042815228 -0.043114280 -0.0433834492 -0.043652756 -0.0438977
74
##
husage -0.01181193 -0.01220566 -0.012596360
husunion . . .
husearns 0.23757467 0.23762223 0.237664496

```

```
huseduc 0.04260468 0.04264427 0.042683515
husblck 0.30543159 0.30807286 0.310710515
hushisp 0.08515599 0.08617269 0.087127411
hushrs -0.07238293 -0.07243357 -0.072479390
kidge6 -0.18699443 -0.18715784 -0.187310237
age 0.06014661 0.06049525 0.060840486
black -0.36548501 -0.36815157 -0.370801517
educ 0.34400477 0.34401378 0.344020123
hispanic -0.00387763 -0.00475531 -0.005573872
union 0.37089203 0.37096443 0.371028317
exper . . .
kidlt6 -0.04409694 -0.04428998 -0.044471566
Estimando o modelo ElasticNet
```

```
Vamos fazer as predicoes e avaliar a performance do
modelo
Vamos fazer as predicoes no modelo de treinamento:
predictions_train <- predict(elastic_reg, x)
```

```
As metricas de performance na base de treinamento
sao:
```

```
eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.8419625 0.2907546
```

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

```
As metricas de performance na base de teste sao:
```

```
eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.9893722 0.259025
```

Calculando o exemplo com a Regressão ElasticNet

```
Vamos fazer a predicao com base nos parametros que
selecionamos
```

```
predict_our_elastic <- predict(elastic_reg,our_pred)
predict_our_elastic
[1] -0.2292382
```

```
Novamente, o resultado é padronizado, nos temos que
reverte-lo para o nivel dos valores originais do
dataset, vamos fazer isso:
```

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

```
wage_pred_elastic
[1] 2.079836
```

```
Entao o salario-hora medio da esposa predito com base
nas caracteristicas informadas é US$2.08
```

```
Vamos criar o intervalo de confianca para o nosso
exemplo
```

```
n <- nrow(train)
m <- wage_pred_elastic
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_elastic <- m + (qnorm(0.025))*dam
CIupr_elastic <- m - (qnorm(0.025))*dam
```

```
Os valores minimo e maximo sao:
```

```
CIlwr_elastic
[1] 2.057977
CIupr_elastic
```

```
[1] 2.101694
Entao, o salario-hora medio da esposa é de US$2.08
e pode variar entre US$2.06 e US$2.10
Conclusão

1) Mostrar em uma tabela as estatísticas dos modelos (RMSE e R2)
#####
Regressão RMSE (train) R^2 (train) RMSE (test) R^2 (test)
Ridge 0.8411446 0.292132 0.9893328 0.259084
Lasso 0.8423523 0.2900978 0.9899383 0.2581767
ElasticNet 0.8419625 0.2907546 0.9893722 0.259025
#####

2) Concluir qual o melhor modelo entre os três
As estatísticas dos modelos demonstraram que os quadrados dos treinamentos
e dos testes estão bem próximos. Comparando os três modelos, aquele que
se aproxima mais, devido as casas decimais (train e test), é o Lasso.

3) mostrar o resultado da predição com intervalos de confiança
#####
Regressão Predição CIlwr CIupr
Ridge 2.161212 2.139354 2.18307
Lasso 2.088536 2.066677 2.110394
Elastic 2.079836 2.057977 2.101694
#####
```

## APÊNDICE F - ARQUITETURA DE DADOS

### A – ENUNCIADO

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

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

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

Nessa atividade é para "construir características".

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

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

#### 2 Melhore uma base de dados ruim

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

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

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

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

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

### B – RESOLUÇÃO

**Instalação/carregamento de pacotes, bibliotecas e semelhantes.**

```

install.packages("mlbench")
package 'mlbench' successfully unpacked and MD5 sums checked
##
The downloaded binary packages are in
C:\Users\Pedro\AppData\Local\Temp\RtmpuoBWMM\downloaded_packages
install.packages("mice")
package 'mice' successfully unpacked and MD5 sums checked
##
The downloaded binary packages are in
C:\Users\Pedro\AppData\Local\Temp\RtmpuoBWMM\downloaded_packages
library(mlbench)
library(caret)
Carregando pacotes exigidos: ggplot2
Carregando pacotes exigidos: lattice
library(mice)
##
Attaching package: 'mice'
The following object is masked from 'package:stats':
##
filter
The following objects are masked from 'package:base':
##
cbind, rbind
1
Carregamento da base de dados.

data(Satellite)
base <- Satellite[, c("x.17", "x.18", "x.19", "x.20", "classes")]
Criação das bases de treino e teste 80% de treino e 20% para teste.

set.seed(7)
indicesSat <- createDataPartition(base$classes, p=0.80, list=FALSE)
treinoSat <- base[indicesSat,]
testeSat <- base[-indicesSat,]
Treinamento dos modelos RandomForest, SVM e RNA. Utilizando a base de treino.

rfSat <- train(classes~., data=treinoSat, method="rf")
svmSat <- train(classes~., data=treinoSat, method="svmRadial")
rnaSat <- train(classes~., data=treinoSat, method="nnet")
Predição usando a base de teste.
predicoes.rfSat <- predict(rfSat, testeSat)
predicoes.svmSat <- predict(svmSat, testeSat)
predicoes.rnaSat <- predict(rnaSat, testeSat)
Matriz de confusão para testar a eficácia dos diferentes modelos.
confusionMatrix(predicoes.rfSat, testeSat$classes)
Confusion Matrix and Statistics
##
Reference
Prediction red soil cotton crop grey soil damp grey soil
red soil 296 1 5 2
cotton crop 0 123 0 0
grey soil 3 0 238 29
damp grey soil 1 0 21 62
vegetation stubble 6 10 0 0
very damp grey soil 0 6 7 32
Reference
Prediction vegetation stubble very damp grey soil
red soil 8 0
cotton crop 2 1
grey soil 1 10

```

```

damp grey soil 2 40
vegetation stubble 120 8
very damp grey soil 8 242
##
Overall Statistics
##
Accuracy : 0.8419
95% CI : (0.8208, 0.8614)
No Information Rate : 0.2383
P-Value [Acc > NIR] : < 2.2e-16
##
Kappa : 0.8047
##
McNemar's Test P-Value : NA
##
Statistics by Class:
##
Class: red soil Class: cotton crop Class: grey soil
Sensitivity 0.9673 0.87857 0.8782
Specificity 0.9836 0.99738 0.9576
Pos Pred Value 0.9487 0.97619 0.8470
Neg Pred Value 0.9897 0.98532 0.9671
Prevalence 0.2383 0.10903 0.2111
Detection Rate 0.2305 0.09579 0.1854
Detection Prevalence 0.2430 0.09813 0.2188
48
Balanced Accuracy 0.9755 0.93797 0.9179
Class: damp grey soil Class: vegetation stubble
Sensitivity 0.49600 0.85106
Specificity 0.94478 0.97900
Pos Pred Value 0.49206 0.83333
Neg Pred Value 0.94560 0.98158
Prevalence 0.09735 0.10981
Detection Rate 0.04829 0.09346
Detection Prevalence 0.09813 0.11215
Balanced Accuracy 0.72039 0.91503
Class: very damp grey soil
Sensitivity 0.8040
Specificity 0.9461
Pos Pred Value 0.8203
Neg Pred Value 0.9403
Prevalence 0.2344
Detection Rate 0.1885
Detection Prevalence 0.2298
Balanced Accuracy 0.8750
confusionMatrix(predicoes.svmSat, testeSat$classes)
Confusion Matrix and Statistics
##
Reference
Prediction red soil cotton crop grey soil damp grey soil
red soil 298 1 4 2
cotton crop 1 120 0 0
grey soil 4 0 260 29
damp grey soil 0 1 7 69
vegetation stubble 3 14 0 2
very damp grey soil 0 4 0 23
Reference
Prediction vegetation stubble very damp grey soil
red soil 7 0
cotton crop 4 0
grey soil 1 12

```

```

damp grey soil 2 32
vegetation stubble 117 3
very damp grey soil 10 254
##
Overall Statistics
##
Accuracy : 0.8707
95% CI : (0.8511, 0.8886)
No Information Rate : 0.2383
P-Value [Acc > NIR] : < 2.2e-16
##
Kappa : 0.8399
##
McNemar's Test P-Value : NA
##
Statistics by Class:
##
49
Class: red soil Class: cotton crop Class: grey soil
Sensitivity 0.9739 0.85714 0.9594
Specificity 0.9857 0.99563 0.9546
Pos Pred Value 0.9551 0.96000 0.8497
Neg Pred Value 0.9918 0.98274 0.9888
Prevalence 0.2383 0.10903 0.2111
Detection Rate 0.2321 0.09346 0.2025
Detection Prevalence 0.2430 0.09735 0.2383
Balanced Accuracy 0.9798 0.92639 0.9570
Class: damp grey soil Class: vegetation stubble
Sensitivity 0.55200 0.82979
Specificity 0.96376 0.98075
Pos Pred Value 0.62162 0.84173
Neg Pred Value 0.95226 0.97904
Prevalence 0.09735 0.10981
Detection Rate 0.05374 0.09112
Detection Prevalence 0.08645 0.10826
Balanced Accuracy 0.75788 0.90527
Class: very damp grey soil
Sensitivity 0.8439
Specificity 0.9624
Pos Pred Value 0.8729
Neg Pred Value 0.9527
Prevalence 0.2344
Detection Rate 0.1978
Detection Prevalence 0.2266
Balanced Accuracy 0.9031
confusionMatrix(predicoes.rnaSat, testeSat$classes)
Confusion Matrix and Statistics
##
Reference
Prediction red soil cotton crop grey soil damp grey soil
red soil 292 2 11 2
cotton crop 7 124 0 0
grey soil 3 0 259 67
damp grey soil 0 0 0 0
vegetation stubble 3 5 0 0
very damp grey soil 1 9 1 56
Reference
Prediction vegetation stubble very damp grey soil
red soil 12 1
cotton crop 13 0
grey soil 1 25

```

```

damp grey soil 0 0
vegetation stubble 100 12
very damp grey soil 15 263
##
Overall Statistics
##
Accuracy : 0.8084
95% CI : (0.7858, 0.8296)
No Information Rate : 0.2383
50
P-Value [Acc > NIR] : < 2.2e-16
##
Kappa : 0.7595
##
McNemar's Test P-Value : NA
##
Statistics by Class:
##
Class: red soil Class: cotton crop Class: grey soil
Sensitivity 0.9542 0.88571 0.9557
Specificity 0.9714 0.98252 0.9052
Pos Pred Value 0.9125 0.86111 0.7296
Neg Pred Value 0.9855 0.98596 0.9871
Prevalence 0.2383 0.10903 0.2111
Detection Rate 0.2274 0.09657 0.2017
Detection Prevalence 0.2492 0.11215 0.2765
Balanced Accuracy 0.9628 0.93412 0.9305
Class: damp grey soil Class: vegetation stubble
Sensitivity 0.00000 0.70922
Specificity 1.00000 0.98250
Pos Pred Value NaN 0.83333
Neg Pred Value 0.90265 0.96478
Prevalence 0.09735 0.10981
Detection Rate 0.00000 0.07788
Detection Prevalence 0.00000 0.09346
Balanced Accuracy 0.50000 0.84586
Class: very damp grey soil
Sensitivity 0.8738
Specificity 0.9166
Pos Pred Value 0.7623
Neg Pred Value 0.9595
Prevalence 0.2344
Detection Rate 0.2048
Detection Prevalence 0.2687
Balanced Accuracy 0.8952

```

### Conclusão.

Após realizar o treinamento e os testes utilizando os modelos RandomForest, SVM (Support Vector Machine) e RNA (Redes Neurais Artificiais), vimos que o melhor modelo foi o SVM tendo uma Acurácia (métrica) de 0.8707 em comparação com o RF que foi de 0.8419 e de RNA de 0.8084.

### Instalação dos pacotes e carregamento das bibliotecas.

```

install.packages("e1071")
package 'e1071' successfully unpacked and MD5 sums checked
##
The downloaded binary packages are in
C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

```

```
install.packages("randomForest")
package 'randomForest' successfully unpacked and MD5 sums checked
##
The downloaded binary packages are in
C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

install.packages("kernlab")
package 'kernlab' successfully unpacked and MD5 sums checked
##
The downloaded binary packages are in
C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

install.packages("caret")
package 'caret' successfully unpacked and MD5 sums checked
Warning: cannot remove prior installation of package 'caret'
Warning in file.copy(savedcopy, lib, recursive = TRUE): problem copying
C:\Program Files\R\R-4.3.3\library\00LOCK\caret\libs\x64\caret.dll to
C:\Program Files\R\R-4.3.3\library\caret\libs\x64\caret.dll: Permission
denied
Warning: restored 'caret'
##
The downloaded binary packages are in
C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

install.packages("RSNNS")
package 'RSNNS' successfully unpacked and MD5 sums checked
##
The downloaded binary packages are in
C:\Users\Pedro\AppData\Local\Temp\RtmpAjok8g\downloaded_packages

library("caret")
Carregando pacotes exigidos: ggplot2
Carregando pacotes exigidos: lattice

library(RSNNS)
Carregando pacotes exigidos: Rcpp
##
Attaching package: 'RSNNS'
The following objects are masked from 'package:caret':
##
confusionMatrix, train
```

#### Caregamento do CSV.

```
dadosVolume <- read.csv("http://www.razer.net.br/datasets/Volumes.csv",
sep=";", dec=",")
```

#### Eliminação da coluna NR.

```
summary(dadosVolume)
NR DAP HT HP
Min. : 1.00 Min. :27.50 Min. :22.80 Min. :1.000
1st Qu.: 25.75 1st Qu.:32.50 1st Qu.:25.50 1st Qu.:1.700
Median : 50.50 Median :35.25 Median :26.40 Median :2.225
Mean : 50.50 Mean :35.86 Mean :26.24 Mean :2.135
3rd Qu.: 75.25 3rd Qu.:39.05 3rd Qu.:27.15 3rd Qu.:2.562
Max. :100.00 Max. :49.00 Max. :28.40 Max. :3.400
VOL
Min. :0.6982
1st Qu.:1.0553
Median :1.3006
Mean :1.3583
```

```
3rd Qu.:1.6191
Max. :2.5245
```

```
dadosVolume$NR <- NULL
summary(dadosVolume)
DAP HT HP VOL
Min. :27.50 Min. :22.80 Min. :1.000 Min. :0.6982
1st Qu.:32.50 1st Qu.:25.50 1st Qu.:1.700 1st Qu.:1.0553
Median :35.25 Median :26.40 Median :2.225 Median :1.3006
Mean :35.86 Mean :26.24 Mean :2.135 Mean :1.3583
3rd Qu.:39.05 3rd Qu.:27.15 3rd Qu.:2.562 3rd Qu.:1.6191
Max. :49.00 Max. :28.40 Max. :3.400 Max. :2.5245
```

**Criação das bases de treino e teste 80% de treino e 20% para teste.**

```
set.seed(7)
indicesVol <- createDataPartition(dadosVolume$VOL, p=0.80, list=FALSE)
treinoVol <- dadosVolume[indicesVol,]
testeVol <- dadosVolume[-indicesVol,]
```

**Treinamento dos modelos RandomForest, SVM, RNA e alom. Utilizando a base de treino.**

```
rfVol <- caret::train(VOL~., data=treinoVol, method="rf")
note: only 2 unique complexity parameters in default grid. Truncating
the grid to 2 .
svmVol <- caret::train(VOL~., data=treinoVol, method="svmRadial")
rnaVol <- caret::train(VOL~., data=treinoVol, method="nnet")
weights: 6
initial value 73.902614
final value 27.985637
converged
weights: 16
initial value 86.376379
final value 27.985637
converged
weights: 26
initial value 73.912653
final value 27.985637
converged
...
converged
weights: 6
initial value 92.359198
iter 10 value 24.083486
final value 24.083406
converged
weights: 16
initial value 81.100308
iter 10 value 24.083503
final value 24.083424
converged
weights: 26
initial value 71.559886
iter 10 value 24.090531
iter 50 value 24.074771
iter 80 value 24.073714
final value 24.073709
converged
Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
trainInfo,
: There were missing values in resampled performance measures.
weights: 6
```

```
initial value 73.697792
final value 24.970359
converged
alomVol <- nls(VOL ~ b0 + b1*DAP*DAP*HT, dadosVolume, start=list(b0=0.5,
b1=0.5))
```

#### Predição usando a base de teste.

```
predicoes.rfVol <- predict(rfVol, testeVol)
predicoes.svmVol <- predict(svmVol, testeVol)
predicoes.rnaVol <- predict(rnaVol, testeVol)
predicoes.alomVol <- predict(alomVol, testeVol)
```

#### Função para calcular o Coeficiente de determinação: R<sup>2</sup>.

Quanto mais perto de 1 melhor é o modelo.

```
R2 <- function(X, Y) { 1 - (sum((X-Y)^2)/sum((X - mean(X))^2)) }
```

#### Função para calcular o Erro padrão da estimativa: Syx.

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

```
Syx <- function (X, Y, n) {sqrt(sum((X - Y)^2)/(n-2))}
```

#### Função para calcular Erro padrão da estimativa como porcentagem: Syx%.

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

```
SyxPerc <- function (X, Y) {(Syx(X, Y, length(X))/mean(X))*100}
```

#### Avaliação dos resultados usando R<sup>2</sup>.

```
R2.rfVol <- R2(testeVol$VOL, predicoes.rfVol)
R2.svmVol <- R2(testeVol$VOL, predicoes.svmVol)
R2.rnaVol <- R2(testeVol$VOL, predicoes.rnaVol)
R2.alomVol <- R2(testeVol$VOL, predicoes.alomVol)
```

#### Avaliação dos resultados usando Syx.

```
Syx.rfVol <- Syx(dadosVolume$VOL, predicoes.rfVol, length(dadosVolume$VOL))
Syx.svmVol <- Syx(dadosVolume$VOL, predicoes.svmVol,
length(dadosVolume$VOL))
Syx.rnaVol <- Syx(dadosVolume$VOL, predicoes.rnaVol,
length(dadosVolume$VOL))
Syx.alomVol <- Syx(dadosVolume$VOL, predicoes.alomVol,
length(dadosVolume$VOL))
```

#### Avaliação dos resultados usando Syx%.

```
SyxPerc.rfVol <- SyxPerc(dadosVolume$VOL, predicoes.rfVol)
SyxPerc.svmVol <- SyxPerc(dadosVolume$VOL, predicoes.svmVol)
SyxPerc.rnaVol <- SyxPerc(dadosVolume$VOL, predicoes.rnaVol)
SyxPerc.alomVol <- SyxPerc(dadosVolume$VOL, predicoes.alomVol)
```

#### Impressão dos resultados: R<sup>2</sup>.

```
str(R2.rfVol)
num 0.854
str(R2.svmVol)
num 0.848
str(R2.rnaVol)
num -0.724
str(R2.alomVol)
num 0.836
```

#### Impressão dos resultados: Syx.

```
str(Syx.rfVol)
num 0.531
str(Syx.svmVol)
num 0.511
```

```
str(Syx.rnaVol)
num 0.548
str(Syx.alomVol)
num 0.54
```

#### **Impressão dos resultados: Syx%.**

```
31
str(SyxPerc.rfVol)
num 39.1
str(SyxPerc.svmVol)
num 37.6
str(SyxPerc.rnaVol)
num 40.3
str(SyxPerc.alomVol)
num 39.8
```

#### **Conclusão.**

O melhor modelo utilizando as funções de R2 (onde aquele que estiver mais perto de 1), Erro padrão da estimativa (onde aquele que estiver mais próximo de 0) e a métrica do percentual do erro (onde aquele que estiver mais próximo de 0), foi o modelo SVM (Support Vector Machine) depois de realizarmos os treinos e os testes, aplicando assim as funções utilizando os valores observados e os valores preditos.

## APÊNDICE G - APRENDIZADO DE MÁQUINA

### A – ENUNCIADO

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

### B – RESOLUÇÃO

#### CLASSIFICAÇÃO

Para o experimento de Classificação:

- Ordenar pela Acurácia (descendente), ou seja, a técnica de melhor acurácia ficará em primeiro na tabela.
- Após o quadro colocar:
  - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia (criar um arquivo com novos casos à sua escolha)
  - A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

#### Veículo

| Técnica        | Parâmetro            | Acurácia  | Matriz de Confusão |
|----------------|----------------------|-----------|--------------------|
| SVM – CV       | C=100 Sigma=0.01     | 0.8510451 | 0.8095             |
| RNA – CV       | size=6 decay=0.4     | 0.7991805 | 0.8036             |
| SVM – Hold-out | C=1 Sigma=0.05355014 | 0.7571993 | 0.6905             |
| RF – CV        | mtry=9               | 0.7549512 | 0.7202             |
| RF – Hold-out  | mtry=10              | 0.7481988 | 0.7202             |
| RNA – Hold-out | size=5 decay=0.1     | 0.5514466 | 0.4524             |
| KNN            | k=1                  | 0.5046058 | 0.4345             |

- Novos casos para a técnica/parâmetro de maior Acurácia

3 rows

| a | Comp | Circ | DCirc | RadRa | PrAxisRa | MaxLRa | ScatRa | Elong | PrAxisRect | MaxLRect | ScVarMaxis | ScVarmaxis | RaCyr | SkewMaxis | Skewmaxis | Kurtmaxis | KurtMaxis | HollRa | tipo |
|---|------|------|-------|-------|----------|--------|--------|-------|------------|----------|------------|------------|-------|-----------|-----------|-----------|-----------|--------|------|
| 1 | 100  | 42   | 90    | 180   | 55       | 11     | 158    | 33    | 25         | 200      | 170        | 400        | 190   | 73        | 9         | 10        | 185       | 190    | ?    |
| 2 | 95   | 41   | 106   | 200   | 65       | 8      | 178    | 40    | 18         | 155      | 203        | 330        | 158   | 74        | 12        | 14        | 193       | 205    | ?    |
| 3 | 83   | 39   | 69    | 168   | 54       | 5      | 135    | 49    | 18         | 125      | 155        | 274        | 162   | 65        | 16        | 6         | 187       | 190    | ?    |

3 rows

| a | Co... | Circ | DCirc | RadRa | PrAxisRa | MaxLRa | ScatRa | Elong | PrAxisRect | MaxLRect | ScVarMaxis | ScVarmaxis | RaCyr | SkewMaxis | Skewmaxis | Kurtmaxis | KurtMaxis | HollRa | predict.svmSigVeic |
|---|-------|------|-------|-------|----------|--------|--------|-------|------------|----------|------------|------------|-------|-----------|-----------|-----------|-----------|--------|--------------------|
| 1 | 100   | 42   | 90    | 180   | 55       | 11     | 158    | 33    | 25         | 200      | 170        | 400        | 190   | 73        | 9         | 10        | 185       | 190    | van                |
| 2 | 95    | 41   | 106   | 200   | 65       | 8      | 178    | 40    | 18         | 155      | 203        | 330        | 158   | 74        | 12        | 14        | 193       | 205    | van                |
| 3 | 83    | 39   | 69    | 168   | 54       | 5      | 135    | 49    | 18         | 125      | 155        | 274        | 162   | 65        | 16        | 6         | 187       | 190    | opel               |

3 rows

- A lista de comandos emitidos do RStudio com os resultados encontra-se abaixo.

#### Diabetes

| Técnica        | Parâmetro              | Acurácia  | Matriz de Confusão |
|----------------|------------------------|-----------|--------------------|
| RF – CV        | mtry=7                 | 0.7806319 | 0.7434             |
| SVM – CV       | C=50 Sigma=0.01        | 0.7789115 | 0.7697             |
| RF – Hold-out  | mtry=2                 | 0.7591633 | 0.6974             |
| SVM – Hold-out | C=0.5 Sigma=0.09153441 | 0.7532372 | 0.7763             |
| RNA – CV       | size=11 decay=0.7      | 0.7321100 | 0.7039             |
| KNN            | k=9                    | 0.6759342 | 0.7237             |
| RNA – Hold-out | size=3 decay=0.1       | 0.6626534 | 0.6974             |

- **Novos casos para a técnica/parâmetro de maior Acurácia**

RStudio: Notebook Output

data.frame  
3 x 10

Description: df [3 x 10]

| num | preg0nt | glucose | pressure | triceps | insulin | mass | pedigree | age | diabetes |
|-----|---------|---------|----------|---------|---------|------|----------|-----|----------|
| 1   | 14      | 98      | 78       | 25      | 189     | 39.9 | 0.312    | 46  | ?        |
| 2   | 8       | 140     | 85       | 0       | 52      | 28.2 | 0.189    | 41  | ?        |
| 3   | 12      | 130     | 58       | 0       | 156     | 30.3 | 0.480    | 55  | ?        |

3 rows

RStudio: Notebook Output

Description: df [3 x 10]

| num | preg0nt | glucose | pressure | triceps | insulin | mass | pedigree | age | predict.rfMtryDiabet |
|-----|---------|---------|----------|---------|---------|------|----------|-----|----------------------|
| 1   | 14      | 98      | 78       | 25      | 189     | 39.9 | 0.312    | 46  | pos                  |
| 2   | 8       | 140     | 85       | 0       | 52      | 28.2 | 0.189    | 41  | neg                  |
| 3   | 12      | 130     | 58       | 0       | 156     | 30.3 | 0.480    | 55  | pos                  |

3 rows

- **A lista de comandos emitidos do RStudio com os resultados encontra-se abaixo.**

## REGRESSÃO

Para o experimento de Regressão:

- Ordenar por R2 descendente, ou seja, a técnica de melhor R2 ficará em primeiro na tabela.
- Após o quadro, colocar:
  - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior R2 (criar um arquivo com novos casos à sua escolha)
  - O Gráfico de Resíduos para a técnica/parâmetro de maior R2
  - A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

## Admissão

| Técnica        | Parâmetro                  | R2        | Syx        | Pearson   | Rmse       | MAE        |
|----------------|----------------------------|-----------|------------|-----------|------------|------------|
| RF – CV        | mtry=2                     | 0.8642698 | 0.0540429  | 0.9342248 | 0.05376647 | 0.03832247 |
| RF – Hold-out  | mtry=2                     | 0.8613151 | 0.05462798 | 0.9327529 | 0.05434855 | 0.03897055 |
| SVM – CV       | C=50<br>Sigma=0.1<br>5     | 0.8536698 | 0.05611351 | 0.926846  | 0.05582648 | 0.04202136 |
| SVM – Hold-out | C=1<br>Sigma=0.1<br>446028 | 0.8333556 | 0.05988194 | 0.9147223 | 0.05957564 | 0.04489135 |
| RNA – CV       | size=10<br>decay=0.1       | 0.831985  | 0.06012769 | 0.9138058 | 0.05982013 | 0.04670783 |
| RNA – Hold-out | size=5<br>decay=0.1        | 0.7799638 | 0.06880936 | 0.8847604 | 0.0684574  | 0.05318685 |
| KNN            | K=5                        | 0.6984365 | 0.08055452 | 0.8379379 | 0.08014248 | 0.06026531 |

- **Novos casos para a técnica/parâmetro de maior Acurácia**

Description: df [3 × 9]

| num<br><int> | GRE.Score<br><int> | TOEFL.Score<br><int> | University.Rating<br><int> | SOP<br><dbl> | LOR<br><int> | CGPA<br><dbl> | Research<br><int> | ChanceOfAdmit<br><chr> |
|--------------|--------------------|----------------------|----------------------------|--------------|--------------|---------------|-------------------|------------------------|
| 1            | 315                | 112                  | 3                          | 2.8          | 9            | 6.89          | 0                 | ?                      |
| 2            | 368                | 90                   | 5                          | 9.5          | 5            | 7.60          | 0                 | ?                      |
| 3            | 250                | 101                  | 3                          | 2.0          | 4            | 7.67          | 1                 | ?                      |

3 rows

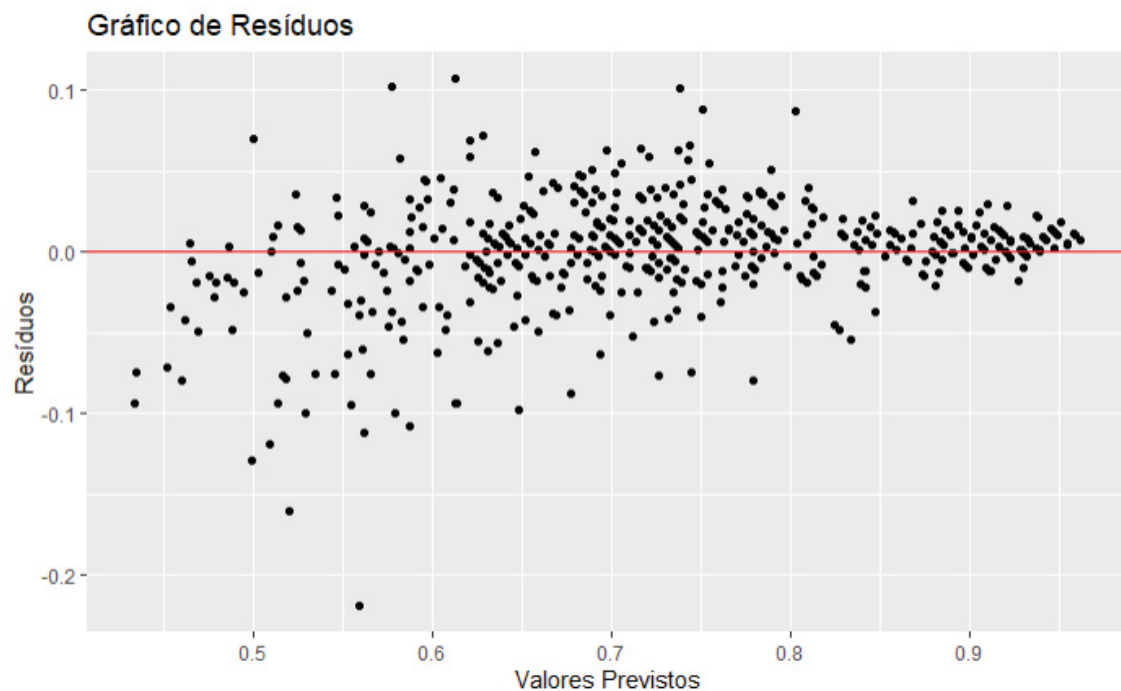
Description: df [3 × 9]

| num<br><int> | GRE.Score<br><int> | TOEFL.Score<br><int> | University.Rating<br><int> | SOP<br><dbl> | LOR<br><int> | CGPA<br><dbl> | Research<br><int> | predict.rfCVAdmin<br><dbl> |
|--------------|--------------------|----------------------|----------------------------|--------------|--------------|---------------|-------------------|----------------------------|
| 1            | 1                  | 315                  | 112                        | 3            | 2.8          | 9             | 6.89              | 0.6288477                  |
| 2            | 2                  | 368                  | 90                         | 5            | 9.5          | 5             | 7.60              | 0.6576732                  |
| 3            | 3                  | 250                  | 101                        | 3            | 2.0          | 4             | 7.67              | 0.5637255                  |

3 rows

- **Gráfico de resíduos para a técnica/parâmetros de maior R2**

```
O Gráfico de Resíduos para a técnica/parâmetro de maior R2
Carregando a biblioteca ggplot2
library(ggplot2)
Criando um conjunto de dados de exemplo
set.seed(202400)
y_true <- dadosAdmin$ChanceOfAdmit # Valores Reais
y_pred <- predict(rfCVAdmin, dadosAdmin) # Valores Estimados
Criando o gráfico de resíduos
ggplot(data = data.frame(y_true, y_pred), aes(x = y_pred, y = y_true -
y_pred)) +
 geom_point() +
 geom_hline(yintercept = 0, color = "red") +
 labs(title = "Gráfico de Resíduos", x = "Valores Previstos", y = "Resíduos")
```



- A lista de comandos emitidos do RStudio com os resultados encontra-se abaixo.

**Biomassa**

| Técnica        | Parâmetro             | R2        | Syx      | Pearson   | Rmse     | MAE      |
|----------------|-----------------------|-----------|----------|-----------|----------|----------|
| RF – CV        | mtry=2                | 0.9128844 | 394.8346 | 0.9648046 | 391.5305 | 114.0772 |
| RF – Hold-out  | mtry=2                | 0.910894  | 399.3195 | 0.9630269 | 395.9778 | 116.4322 |
| RNA – CV       | size=9<br>decay=0.7   | 0.8783715 | 466.5355 | 0.9489315 | 462.6314 | 131.4615 |
| SVM – CV       | C=50<br>Sigma=0.01    | 0.8716717 | 479.2127 | 0.9372619 | 475.2025 | 158.1715 |
| KNN            | K=1                   | 0.8113007 | 581.1018 | 0.9234626 | 576.239  | 152.3858 |
| RNA – Hold-out | size=3<br>decay=0.1   | 0.8095102 | 583.8523 | 0.9037708 | 578.9664 | 204.3742 |
| SVM – Hold-out | C=1<br>Sigma=1.690646 | 0.3862202 | 1048.029 | 0.6508038 | 1039.259 | 260.293  |

- **Novos casos para a técnica/parâmetro de maior Acurácia**



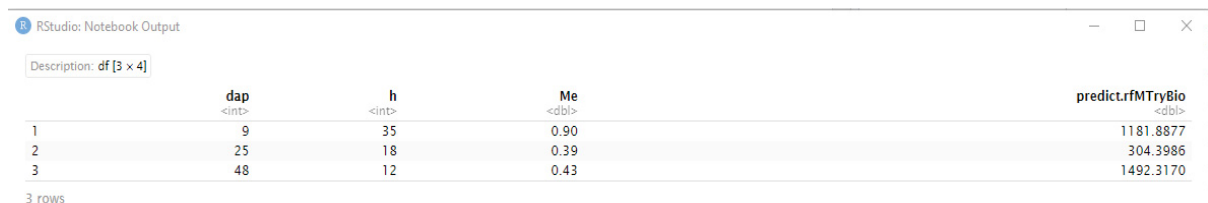
R Console

data.frame  
3 x 4

Description: df [3 x 4]

| dap   | h     | Me    | biomassa |
|-------|-------|-------|----------|
| <int> | <int> | <dbl> | <chr>    |
| 9     | 35    | 0.90  | ?        |
| 25    | 18    | 0.39  | ?        |
| 48    | 12    | 0.43  | ?        |

3 rows



RStudio: Notebook Output

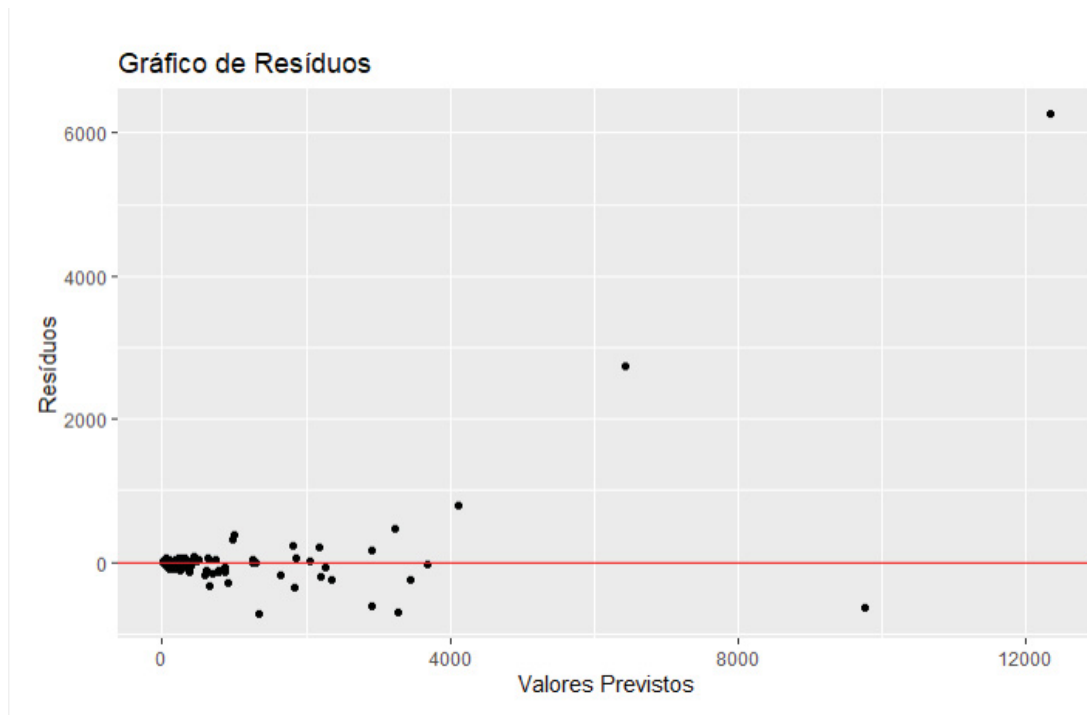
Description: df [3 x 4]

|   | dap   | h     | Me    | predict.rfMTryBio |
|---|-------|-------|-------|-------------------|
|   | <int> | <int> | <dbl> | <dbl>             |
| 1 | 9     | 35    | 0.90  | 1181.8877         |
| 2 | 25    | 18    | 0.39  | 304.3986          |
| 3 | 48    | 12    | 0.43  | 1492.3170         |

3 rows

- **Gráfico de resíduos para a técnica/parâmetros de maior R2**

```
O Gráfico de Resíduos para a técnica/parâmetro de maior R2
Carregando a biblioteca ggplot2
library(ggplot2)
Criando um conjunto de dados de exemplo
set.seed(202400)
y_true <- dadosBio$biomassa # Valores Reais
y_pred <- predict(rfMTryBio, dadosBio) # Valores Estimados
Criando o gráfico de resíduos
ggplot(data = data.frame(y_true, y_pred), aes(x = y_pred, y = y_true -
y_pred)) +
 geom_point() +
 geom_hline(yintercept = 0, color = "red") +
 labs(title = "Gráfico de Resíduos", x = "Valores Previstos", y = "Resíduos")
```



- A lista de comandos emitidos do RStudio com os resultados encontra-se abaixo.

## AGRUPAMENTO

### Veículo

Lista de Clusters gerados:

10 primeiras linhas do arquivo com o cluster correspondente.

Usa 10 clusters no experimento.

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

```
Leitura dos dados Veículos
setwd("D:/IA/IAA008/Trabalho/11 - Agrupamento - Praticas - 4 - Veiculos/")
dadosAgrupVeic <- read.csv("4 - Veiculos - Dados.csv", header=T)
#View(dadosAgrupVeic)
Executa o Kmeans
set.seed(202400)
cluster.results <- kmodes(dadosAgrupVeic, 10, iter.max = 10, weighted = FALSE)
cluster.results[1:10]
```

RStudio Notebook Output

data frame 10 x 20

|    | a  | Comp | Circ | DCirc | RadRa | PrAxisRa | MaxLRa | ScatRa | Elong | PrAxisRect | MaxLRect | ScVarMaxis | ScVarmaxis | RaCyr | SkewMaxis | Skewmaxis | Kurtmaxis | KurtMaxis | HolIRa | Class |
|----|----|------|------|-------|-------|----------|--------|--------|-------|------------|----------|------------|------------|-------|-----------|-----------|-----------|-----------|--------|-------|
| 1  | 8  | 92   | 40   | 66    | 151   | 59       | 7      | 128    | 52    | 18         | 139      | 159        | 246        | 139   | 64        | 3         | 7         | 183       | 183    | van   |
| 2  | 7  | 85   | 44   | 68    | 120   | 53       | 7      | 150    | 45    | 19         | 145      | 169        | 322        | 172   | 85        | 4         | 1         | 179       | 182    | bus   |
| 3  | 2  | 86   | 41   | 66    | 155   | 55       | 7      | 149    | 45    | 19         | 144      | 172        | 330        | 173   | 72        | 6         | 14        | 180       | 185    | bus   |
| 4  | 10 | 91   | 39   | 83    | 197   | 62       | 8      | 177    | 37    | 21         | 131      | 202        | 281        | 138   | 71        | 0         | 2         | 189       | 198    | opel  |
| 5  | 3  | 101  | 51   | 104   | 197   | 60       | 10     | 214    | 31    | 24         | 162      | 226        | 635        | 214   | 68        | 0         | 21        | 188       | 197    | saab  |
| 6  | 16 | 104  | 55   | 103   | 204   | 64       | 11     | 219    | 30    | 25         | 172      | 225        | 706        | 216   | 72        | 0         | 11        | 187       | 196    | opel  |
| 7  | 13 | 98   | 47   | 81    | 191   | 64       | 6      | 169    | 39    | 20         | 148      | 189        | 430        | 171   | 70        | 0         | 19        | 195       | 201    | bus   |
| 8  | 1  | 90   | 37   | 85    | 150   | 64       | 10     | 157    | 43    | 20         | 134      | 173        | 354        | 186   | 75        | 1         | 7         | 183       | 195    | van   |
| 9  | 6  | 85   | 38   | 53    | 130   | 57       | 6      | 114    | 57    | 17         | 128      | 137        | 259        | 145   | 82        | 1         | 9         | 181       | 183    | opel  |
| 10 | 37 | 93   | 43   | 70    | 169   | 59       | 7      | 151    | 44    | 19         | 145      | 170        | 339        | 177   | 75        | 2         | 12        | 184       | 184    | bus   |

10 rows

```
Cria um arquivo com todos os registros e mais os clusters de cada um
resultado <- cbind(dadosAgrupVeic, cluster.results$cluster)
resultado[1:10,]
```

RStudio Notebook Output

Description: df [10 x 21]

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

10 rows

## REGRAS DE ASSOCIAÇÃO

### Musculação

Regras geradas com uma configuração de Suporte e Confiança.

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

```
Instalação dos pacotes necessários
install.packages('arules', dep=T)
library(arules)
library(datasets)
Leitura dos dados
setwd("D:/IA/IAA008/Trabalho/")
dadosMusc <- read.transactions(file="2 - Musculacao -
Dados.csv", format="basket", sep=";")
inspect(dadosMusc[1:5])
Extrai as regras com suporte de 0.001 e confiança de 0.9
set.seed(202400)
rules <- apriori(dadosMusc, parameter = list(supp = 0.001, conf = 0.9, target
= "rules"))
inspect(rules)
```

RStudio: Notebook Output

Description: df [106 x 8]

|      | lhs                   | rhs              | support    | confidence | coverage   | lift     | count |
|------|-----------------------|------------------|------------|------------|------------|----------|-------|
|      | <chr>                 | <chr>            | <dbl>      | <dbl>      | <dbl>      | <dbl>    | <int> |
| [1]  | {Crucifixo}           | => {Afundo}      | 0.07692308 | 1.0000000  | 0.07692308 | 2.888889 | 2     |
| [2]  | {Crucifixo}           | => {Gemeos}      | 0.07692308 | 1.0000000  | 0.07692308 | 1.529412 | 2     |
| [3]  | {Crucifixo}           | => {LegPress}    | 0.07692308 | 1.0000000  | 0.07692308 | 1.238095 | 2     |
| [4]  | {Adutor}              | => {Agachamento} | 0.11538462 | 1.0000000  | 0.11538462 | 3.250000 | 3     |
| [5]  | {Adutor}              | => {LegPress}    | 0.11538462 | 1.0000000  | 0.11538462 | 1.238095 | 3     |
| [6]  | {Flexor}              | => {Esteira}     | 0.07692308 | 1.0000000  | 0.07692308 | 2.166667 | 2     |
| [7]  | {Flexor}              | => {Extensor}    | 0.07692308 | 1.0000000  | 0.07692308 | 2.000000 | 2     |
| [8]  | {Flexor}              | => {Bicicleta}   | 0.07692308 | 1.0000000  | 0.07692308 | 1.857143 | 2     |
| [9]  | {Flexor}              | => {LegPress}    | 0.07692308 | 1.0000000  | 0.07692308 | 1.238095 | 2     |
| [10] | {Agachamento}         | => {LegPress}    | 0.30769231 | 1.0000000  | 0.30769231 | 1.238095 | 8     |
| [11] | {Afundo}              | => {Gemeos}      | 0.34615385 | 1.0000000  | 0.34615385 | 1.529412 | 9     |
| [12] | {AgachamentoSmith}    | => {Extensor}    | 0.34615385 | 0.9000000  | 0.38461538 | 1.800000 | 9     |
| [13] | {Esteira}             | => {Extensor}    | 0.42307692 | 0.9166667  | 0.46153846 | 1.833333 | 11    |
| [14] | {Extensor}            | => {Bicicleta}   | 0.46153846 | 0.9230769  | 0.50000000 | 1.714286 | 12    |
| [15] | {Afundo, Crucifixo}   | => {Gemeos}      | 0.07692308 | 1.0000000  | 0.07692308 | 1.529412 | 2     |
| [16] | {Crucifixo, Gemeos}   | => {Afundo}      | 0.07692308 | 1.0000000  | 0.07692308 | 2.888889 | 2     |
| [17] | {Afundo, Crucifixo}   | => {LegPress}    | 0.07692308 | 1.0000000  | 0.07692308 | 1.238095 | 2     |
| [18] | {Crucifixo, LegPress} | => {Afundo}      | 0.07692308 | 1.0000000  | 0.07692308 | 2.888889 | 2     |
| [19] | {Crucifixo, Gemeos}   | => {LegPress}    | 0.07692308 | 1.0000000  | 0.07692308 | 1.238095 | 2     |

1-19 of 106 rows

Previous 1 2 3 4 5 6 Next

```
Extrai as regras com suporte de 0.001 e confiança de 0.7
```

```
set.seed(202400)
rules2 <- apriori(dadosMusc, parameter = list(supp = 0.001, conf = 0.7, target
= "rules"))
inspect(rules2)
```

RStudio: Notebook Output

Description: df [156 x 8]

|      | lhs<br><chr>       | rhs<br><chr>     | support<br><dbl> | confidence<br><dbl> | coverage<br><dbl> | lift<br><dbl> | count<br><int> |
|------|--------------------|------------------|------------------|---------------------|-------------------|---------------|----------------|
| [1]  | {}                 | => {LegPress}    | 0.80769231       | 0.8076923           | 1.00000000        | 1.0000000     | 21             |
| [2]  | {Crucifixo}        | => {Afundo}      | 0.07692308       | 1.0000000           | 0.07692308        | 2.8888889     | 2              |
| [3]  | {Crucifixo}        | => {Cemeos}      | 0.07692308       | 1.0000000           | 0.07692308        | 1.5294118     | 2              |
| [4]  | {Crucifixo}        | => {LegPress}    | 0.07692308       | 1.0000000           | 0.07692308        | 1.2380952     | 2              |
| [5]  | {Adutor}           | => {Agachamento} | 0.11538462       | 1.0000000           | 0.11538462        | 3.2500000     | 3              |
| [6]  | {Adutor}           | => {LegPress}    | 0.11538462       | 1.0000000           | 0.11538462        | 1.2380952     | 3              |
| [7]  | {Flexor}           | => {Esteira}     | 0.07692308       | 1.0000000           | 0.07692308        | 2.1666667     | 2              |
| [8]  | {Flexor}           | => {Extensor}    | 0.07692308       | 1.0000000           | 0.07692308        | 2.0000000     | 2              |
| [9]  | {Flexor}           | => {Bicicleta}   | 0.07692308       | 1.0000000           | 0.07692308        | 1.8571429     | 2              |
| [10] | {Flexor}           | => {LegPress}    | 0.07692308       | 1.0000000           | 0.07692308        | 1.2380952     | 2              |
| [11] | {Agachamento}      | => {LegPress}    | 0.30769231       | 1.0000000           | 0.30769231        | 1.2380952     | 8              |
| [12] | {Afundo}           | => {Cemeos}      | 0.34615385       | 1.0000000           | 0.34615385        | 1.5294118     | 9              |
| [13] | {Afundo}           | => {LegPress}    | 0.26923077       | 0.7777778           | 0.34615385        | 0.9629630     | 7              |
| [14] | {AgachamentoSmith} | => {Esteira}     | 0.30769231       | 0.8000000           | 0.38461538        | 1.7333333     | 8              |
| [15] | {AgachamentoSmith} | => {Extensor}    | 0.34615385       | 0.9000000           | 0.38461538        | 1.8000000     | 9              |
| [16] | {AgachamentoSmith} | => {Bicicleta}   | 0.30769231       | 0.8000000           | 0.38461538        | 1.4857143     | 8              |
| [17] | {AgachamentoSmith} | => {LegPress}    | 0.26923077       | 0.7000000           | 0.38461538        | 0.8666667     | 7              |
| [18] | {Esteira}          | => {Extensor}    | 0.42307692       | 0.9166667           | 0.46153846        | 1.8333333     | 11             |
| [19] | {Extensor}         | => {Esteira}     | 0.42307692       | 0.8461538           | 0.50000000        | 1.8333333     | 11             |
| [20] | {Esteira}          | => {Bicicleta}   | 0.38461538       | 0.8333333           | 0.46153846        | 1.5476190     | 10             |
| [21] | {Bicicleta}        | => {Esteira}     | 0.38461538       | 0.7142857           | 0.53846154        | 1.5476190     | 10             |

1-21 of 156 rows

Previous 1 2 3 4 5 6 ... 8 Next

### Extrai as regras com suporte de 0.1 e confiança de 0.7

```
set.seed(202400)
```

```
rules3 <- apriori(dadosMusc, parameter = list(supp = 0.1, conf = 0.7, target = "rules"))
```

```
inspect(rules3)
```

RStudio: Notebook Output

Description: df [71 x 8]

|      | lhs<br><chr>          | rhs<br><chr>     | support<br><dbl> | confidence<br><dbl> | coverage<br><dbl> | lift<br><dbl> | count<br><int> |
|------|-----------------------|------------------|------------------|---------------------|-------------------|---------------|----------------|
| [1]  | {}                    | => {LegPress}    | 0.8076923        | 0.8076923           | 1.0000000         | 1.0000000     | 21             |
| [2]  | {Adutor}              | => {Agachamento} | 0.1153846        | 1.0000000           | 0.1153846         | 3.2500000     | 3              |
| [3]  | {Adutor}              | => {LegPress}    | 0.1153846        | 1.0000000           | 0.1153846         | 1.2380952     | 3              |
| [4]  | {Agachamento}         | => {LegPress}    | 0.3076923        | 1.0000000           | 0.3076923         | 1.2380952     | 8              |
| [5]  | {Afundo}              | => {Cemeos}      | 0.3461538        | 1.0000000           | 0.3461538         | 1.5294118     | 9              |
| [6]  | {Afundo}              | => {LegPress}    | 0.2692308        | 0.7777778           | 0.3461538         | 0.9629630     | 7              |
| [7]  | {AgachamentoSmith}    | => {Esteira}     | 0.3076923        | 0.8000000           | 0.3846154         | 1.7333333     | 8              |
| [8]  | {AgachamentoSmith}    | => {Extensor}    | 0.3461538        | 0.9000000           | 0.3846154         | 1.8000000     | 9              |
| [9]  | {AgachamentoSmith}    | => {Bicicleta}   | 0.3076923        | 0.8000000           | 0.3846154         | 1.4857143     | 8              |
| [10] | {AgachamentoSmith}    | => {LegPress}    | 0.2692308        | 0.7000000           | 0.3846154         | 0.8666667     | 7              |
| [11] | {Esteira}             | => {Extensor}    | 0.4230769        | 0.9166667           | 0.4615385         | 1.8333333     | 11             |
| [12] | {Extensor}            | => {Esteira}     | 0.4230769        | 0.8461538           | 0.5000000         | 1.8333333     | 11             |
| [13] | {Esteira}             | => {Bicicleta}   | 0.3846154        | 0.8333333           | 0.4615385         | 1.5476190     | 10             |
| [14] | {Bicicleta}           | => {Esteira}     | 0.3846154        | 0.7142857           | 0.5384615         | 1.5476190     | 10             |
| [15] | {Extensor}            | => {Bicicleta}   | 0.4615385        | 0.9230769           | 0.5000000         | 1.7142857     | 12             |
| [16] | {Bicicleta}           | => {Extensor}    | 0.4615385        | 0.8571429           | 0.5384615         | 1.7142857     | 12             |
| [17] | {Cemeos}              | => {LegPress}    | 0.4615385        | 0.7058824           | 0.6538462         | 0.8739496     | 12             |
| [18] | {Adutor, Agachamento} | => {LegPress}    | 0.1153846        | 1.0000000           | 0.1153846         | 1.2380952     | 3              |
| [19] | {Adutor, LegPress}    | => {Agachamento} | 0.1153846        | 1.0000000           | 0.1153846         | 3.2500000     | 3              |
| [20] | {Agachamento, Cemeos} | => {LegPress}    | 0.1923077        | 1.0000000           | 0.1923077         | 1.2380952     | 5              |
| [21] | {Afundo, Bicicleta}   | => {Cemeos}      | 0.1538462        | 1.0000000           | 0.1538462         | 1.5294118     | 4              |

1-21 of 71 rows

Previous 1 2 3 4 Next

```
```\r setup, include=FALSE)
knitr::opts_chunk$set(echo = TRUE)
options(repos = c(CRAN = "https://cloud.r-project.org/"))
install.packages('tinytex')
tinytex::install_tinytex(force = TRUE)
```\r
```

# Pacotes necessários:

```
```\r pacotes
```

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

```
#install.packages("caret")
library("caret")
#install.packages("Metrics")
library(Metrics)
#install.packages("rlang")
#install.packages("vctrs")
#install.packages("ggplot2")
library(ggplot2)
#install.packages("mlbench")
library(mlbench)
#install.packages("mice")
library(mice)
#install.packages("klaR")
library(klaR)
```

```
```
```

## # CLASSIFICACAO - VEICULOS E DIABETES

### ## BASE - VEICULOS

```
Leitura dos dados Veículos
```{r veic0}
setwd("D:/IA/IAA008/Trabalho")
dadosVeic <- read.csv("6 - Veiculos - Dados.csv", header=T)
##view(dadosVeic)
```

Cria arquivos de treino e teste
```{r veic1}
set.seed(202400)
indVeic <- createDataPartition(1:nrow(dadosVeic), p=0.80, list = FALSE)
treinoVeic <- dadosVeic[indVeic,]
testeVeic <- dadosVeic[-indVeic,]
```

Executa o Knn com esse grid
```{r veic3}
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
set.seed(202400)
knnVeic <- train(tipo ~ ., data = treinoVeic, method = "knn",
tuneGrid=tuneGrid)
knnVeic
```

Faz a predição e mostra a matriz de confusão - KNN
```{r veic4}
predict.knnVeic <- predict(knnVeic, testeVeic)
confusionMatrix(predict.knnVeic, as.factor(testeVeic$tipo))
```

Executa o RNA - Hold-out
```{r veic5}
set.seed(202400)
rnaVeic <- train(tipo~., data=treinoVeic, method="nnet",trace=FALSE)
rnaVeic
```

Matriz de confusão
```{r veic6}
predicoes.rnaVeic <- predict(rnaVeic, testeVeic)
```

```

confusionMatrix(predicoes.rnaVeic, as.factor(testeVeic$tipo))
```

Executa o RNA - Cross-Validation
indica o método cv e numero de folders 10
```{r veic7}
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202400)
rnaCVVeic <- train(tipo~., data=treinoVeic, method="nnet", trace=FALSE,
trControl=ctrl)
rnaCVVeic
```

Faz as predições e mostra matriz de confusão
```{r veic8}
predict.rnaCVVeic <- predict(rnaCVVeic, testeVeic)
confusionMatrix(predict.rnaCVVeic, as.factor(testeVeic$tipo))
```

RNA - Cross-validation - Size, Decay definidos no grid
```{r veic9}
gridVeic <- expand.grid(size = seq(from = 1, to = 15, by = 5),
                        decay = seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202400)
rnaCVVeicG <- train(
  form = tipo~. ,
  data = treinoVeic ,
  method = "nnet" ,
  tuneGrid = gridVeic ,
  trControl = ctrl ,
  maxit = 2000, trace=FALSE)
### Verifica o resultado do Treinamento
rnaCVVeicG
```

Faz as predições e mostra matriz de confusão
```{r veic10}
predict.rnaCVVeicG <- predict(rnaCVVeicG, testeVeic)
confusionMatrix(predict.rnaCVVeicG, as.factor(testeVeic$tipo))
```

Executa o SVM
Treinar SVM com a base de Treino - SVM - Hold-out
```{r veic11}
set.seed(202400)
svmVeic <- train(tipo~., data=treinoVeic, method="svmRadial")
svmVeic
```

Matriz de confusão - SVM - Hold-out
```{r veic12}
predicoes.svmVeic <- predict(svmVeic, testeVeic)
confusionMatrix(predicoes.svmVeic, as.factor(testeVeic$tipo))
```

Cross-validation SVM
```{r veic13}
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202400)
svmCVVeic <- train(tipo~., data=treinoVeic, method="svmRadial",
trControl=ctrl)

```

```

svmCVVeic
```

Matriz de confusão
```{r veic14}
predict.svmCVVeic <- predict(svmCVVeic, testeVeic)
confusionMatrix(predict.svmCVVeic, as.factor(testeVeic$tipo))
```

SVM com Vários C e sigma
```{r veic15}
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202400)
svmSigVeic <- train(tipo~., data=treinoVeic, method="svmRadial",
                    trControl=ctrl, tuneGrid=tuneGrid)
svmSigVeic
```

Matriz de confusão
```{r veic16}
predict.svmSigVeic <- predict(svmSigVeic, testeVeic)
confusionMatrix(predict.svmSigVeic, as.factor(testeVeic$tipo))
```

Executa o RandomForest
RF - Hold-out
```{r veic18}
set.seed(202400)
rfVeic <- train(tipo~., data=treinoVeic, method="rf")
rfVeic
```

Predições com o arquivo de teste
```{r veic19}
predicoes.rfVeic <- predict(rfVeic, testeVeic)
confusionMatrix(predicoes.rfVeic, as.factor(testeVeic$tipo))
```

RF - Cross-validation
```{r veic20}
set.seed(202400)
rfCVVeic <- train(tipo~., data=treinoVeic, method="rf", trControl=ctrl)
rfCVVeic
```

Matriz de confusao
```{r veic21}
predict.rfCVVeic <- predict(rfCVVeic, testeVeic)
confusionMatrix(predict.rfCVVeic, as.factor(testeVeic$tipo))
```

Vários mtry
```{r veic22}
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202400)
rfMTryVeic <- train(tipo~., data=treinoVeic, method="rf", trControl=ctrl,
                    tuneGrid=tuneGrid)
rfMTryVeic
```

matriz de confusao

```

```

```{r veic23}
predict.rfMtryVeic <- predict(rfMtryVeic, testeVeic)
confusionMatrix(predict.rfMtryVeic, as.factor(testeVeic$tipo))
```

Predições de novos casos para o de melhor acurácia
```{r veic24}
dados_novos_casos <- read.csv("6 - Veiculos - Novos Dados.csv")
dados_novos_casos$Id <- NULL
#view(dados_novos_casos)
dados_novos_casos
```

Resultado do novos casos
```{r veic25}
predict.svmSigVeic <- predict(svmSigVeic, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svmSigVeic)
resultado$tipo <- NULL
#view(resultado)
resultado
```

```

## ## BASE - DIABETES

```

Leitura dos dados Diabetes
```{r Diabet0}
setwd("D:/IA/IAA008/Trabalho")
dadosDiabet <- read.csv("10 - Diabetes - Dados.csv", header=T)
#view(dadosDiabet)
```

Cria arquivos de treino e teste
```{r Diabet1}
set.seed(202400)
indDiabet <- createDataPartition(1:nrow(dadosDiabet), p=0.80, list = FALSE)
treinoDiabet <- dadosDiabet[indDiabet,]
testeDiabet <- dadosDiabet[-indDiabet,]
```

Executa o Knn com esse grid
```{r Diabet3}
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
set.seed(202400)
knnDiabet <- train(diabetes ~ ., data = treinoDiabet, method = "knn",
                  tuneGrid=tuneGrid)
knnDiabet
```

Faz a predição e mostra a matriz de confusão
```{r Diabet4}
predict.knnDiabet <- predict(knnDiabet, testeDiabet)
confusionMatrix(predict.knnDiabet, as.factor(testeDiabet$diabetes))
```

Executa o RNA - Hold-out
```{r Diabet5}
set.seed(202400)
rnaDiabet <- train(diabetes~., data=treinoDiabet, method="nnet",trace=FALSE)
rnaDiabet
```

```

```

Predições dos valores do conjunto de teste e Matriz de confusão - RNA -
Hold-out
```{r Diabet6}
predicoes.rnaDiabet <- predict(rnaDiabet, testeDiabet)
confusionMatrix(predicoes.rnaDiabet, as.factor(testeDiabet$diabetes))
```

Executa o RNA - Cross-Validation
indica o método cv e numero de folders 10
```{r Diabet7}
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202400)
rnaCVDiabet <- train(diabetes~., data=treinoDiabet,
method="nnet",trace=FALSE,
trControl=ctrl)
rnaCVDiabet
```

Faz as predições e mostra matriz de confusão
```{r Diabet8}
predict.rnaCVDiabet <- predict(rnaCVDiabet, testeDiabet)
confusionMatrix(predict.rnaCVDiabet, as.factor(testeDiabet$diabetes))
```

size, decay
```{r Diabet9}
gridDiabet <- expand.grid(size = seq(from = 1, to = 15, by = 5),
decay = seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202400)
rnaCVDiabet <- train(
form = diabetes~. ,
data = treinoDiabet ,
method = "nnet" ,
tuneGrid = gridDiabet ,
trControl = ctrl ,
maxit = 2000,trace=FALSE)
### Verifica o resultado do Treinamento
rnaCVDiabet
```

Faz as predições e mostra matriz de confusão
```{r Diabet10}
predict.rnaCVDiabet <- predict(rnaCVDiabet, testeDiabet)
confusionMatrix(predict.rnaCVDiabet, as.factor(testeDiabet$diabetes))
```

Executa o SVM
SVM - Hold-out
```{r Diabet11}
set.seed(202400)
svmDiabet <- train(diabetes~., data=treinoDiabet, method="svmRadial")
svmDiabet
```

Predições com o arquivo de teste - SVM - Hold-out
```{r Diabet12}
predicoes.svmDiabet <- predict(svmDiabet, testeDiabet)
confusionMatrix(predicoes.svmDiabet, as.factor(testeDiabet$diabetes))
```

Cross-validation SVM

```

```

```{r Diabet13}
set.seed(202400)
svmCVDiabet <- train(diabetes~., data=treinoDiabet, method="svmRadial",
                     trControl=ctrl)

svmCVDiabet
```

Matriz de confusão
```{r Diabet14}
predict.svmCVDiabet <- predict(svmCVDiabet, testeDiabet)
confusionMatrix(predict.svmCVDiabet, as.factor(testeDiabet$diabetes))
```

Vários C e sigma
```{r Diabet15}
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202400)
svmSigDiabet <- train(diabetes~., data=treinoDiabet, method="svmRadial",
                     trControl=ctrl, tuneGrid=tuneGrid)

svmSigDiabet
```

Matriz de confusão
```{r Diabet16}
predict.svmSigDiabet <- predict(svmSigDiabet, testeDiabet)
confusionMatrix(predict.svmSigDiabet, as.factor(testeDiabet$diabetes))
```

Executa o RandomForest
RF - Hold-out
```{r Diabet17}
set.seed(202400)
rfDiabet <- train(diabetes~., data=treinoDiabet, method="rf")
rfDiabet
```

Predições com o arquivo de teste
```{r Diabet19}
predicoes.rfDiabet <- predict(rfDiabet, testeDiabet)
confusionMatrix(predicoes.rfDiabet, as.factor(testeDiabet$diabetes))
```

RF - Cross-validation
```{r Diabet20}
set.seed(202400)
rfCVDiabet <- train(diabetes~., data=treinoDiabet, method="rf",
                  trControl=ctrl)
rfCVDiabet
```

Matriz de confusao
```{r Diabet21}
predict.rfCVDiabet <- predict(rfCVDiabet, testeDiabet)
confusionMatrix(predict.rfCVDiabet, as.factor(testeDiabet$diabetes))
```

Vários mtry
```{r Diabet22}
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202400)
rfMTryDiabet <- train(diabetes~., data=treinoDiabet, method="rf",

```

```

                                trControl=ctrl, tuneGrid=tuneGrid)

rfMtryDiabet
```

matriz de confusao
```{r Diabet23}
predict.rfMtryDiabet <- predict(rfMtryDiabet, testeDiabet)
confusionMatrix(predict.rfMtryDiabet, as.factor(testeDiabet$diabetes))
```

Predições de novos casos para o de melhor acurácia
```{r Diabet24}
dados_novos_casos <- read.csv("10 - Diabetes - Novos Dados.csv")
dados_novos_casos$Id <- NULL
##view(dados_novos_casos)
dados_novos_casos
```

Resultados dos novos casos
```{r Diabet25}
predict.rfMtryDiabet <- predict(rfCVDiabet, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.rfMtryDiabet)
resultado$diabetes <- NULL
##view(resultado)
resultado
```

```

## # REGRESAO - ADMISSAO E BIOMASSA

```

Calculo das principais estatísticas da Regressão
```{r funcoes, include=FALSE}
func_resultados <- function(treino, observado, df) {

  predito <- predict(treino, df)
  SSE <- sum((predito - observado)^2)
  SST <- sum((observado - mean(observado))^2)
  R_square <- 1 - (SSE / SST)
  RMSE <- sqrt(SSE/nrow(df))
  Syx <- sqrt(SSE/(nrow(df)-1))
  mae <- mae(observado, predito)
  pearson <- cor(observado, predito)

  # As metricas do modelo:
  data.frame(
    Rsquare = R_square,
    Syx = Syx,
    Pearson = pearson,
    RMSE = RMSE,
    MAE = mae
  )
}
```

```

## ## Base: ADMISSAO

```

Leitura dos dados
```{r admil}
setwd("D:/IA/IAA008/Trabalho")
dadosAdmin <- read.csv("9 - Admissao - Dados.csv", header=T)
#view(dadosAdmin)

```

```

` ``
### Cria arquivos de treino e teste
```{r admi2}
set.seed(202400)
indAdmin <- createDataPartition(dadosAdmin$ChanceOfAdmit, p=0.80, list =
FALSE)
treinoAdmin <- dadosAdmin[indAdmin,]
testeAdmin <- dadosAdmin[-indAdmin,]
```

### Prepara um grid com os valores de k que serão usados
```{r admi3}
tuneGridAdmin <- expand.grid(k = c(1,3,5,7,9))
set.seed(202400)
knnAdmin <- train(ChanceOfAdmit~ ., data = treinoAdmin, method = "knn",
tuneGrid=tuneGridAdmin)
knnAdmin
```

### Mostra as métricas
```{r admin4}
func_resultados(knnAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

### Executa o RNA - Hold-out
### Treino com Hold-Out
```{r admin5}
set.seed(202400)
rnaAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="nnet",
linout=T,
trace=FALSE)
rnaAdmin
```

### Mostra as métricas
```{r admin6}
func_resultados(rnaAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

### Executa o RNA - Cross-Validation
### indica o método cv e numero de folders 10
```{r admin7}
ctrlAdmin <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(size = seq(from = 1, to = 10, by = 1),
decay = seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202400)
rnaCVAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="nnet",
trainControl=ctrlAdmin, tuneGrid=tuneGrid, linout=T,
MaxNWts=10000, maxit=2000, trace=F)
rnaCVAdmin
```

### Mostra as métricas
```{r admin8}
func_resultados(rnaCVAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

### Executa o SVM - Hold-out
```{r admin9}
set.seed(202400)

```

```

svmAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="svmRadial")
svmAdmin
```

### Mostra as métricas
```{r admin10}
func_resultados(svmAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

#### SVM - Cross-validation
```{r admin11}
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202400)
svmCVAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="svmRadial",
 trControl=ctrl)
svmCVAdmin
```

### Mostra as métricas
```{r admin12}
func_resultados(svmCVAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

#### SVM - Vários C e sigma
```{r admin13}
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202400)
svmSigAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="svmRadial",
 trControl=ctrl, tuneGrid=tuneGrid)
svmSigAdmin
```

### Mostra as métricas
```{r admin14}
func_resultados(svmSigAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

#### Executar Randon Forest
### RF - Hold-out
```{r admin15}
set.seed(202400)
rfAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="rf")
rfAdmin
```

### Mostra as métricas
```{r admin16}
func_resultados(rfAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

#### RF - Cross-validation
```{r admin17}
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202400)
rfCVAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="rf",
 trControl=ctrl)
rfCVAdmin
```

### Mostra as métricas
```{r admin18}

```

```

func_resultados(rfCVAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

#### Vários mtry
```{r admin19}
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202400)
rfMTryAdmin <- train(ChanceOfAdmit~., data=treinoAdmin, method="rf",
 trControl=ctrl, tuneGrid=tuneGrid)
rfMTryAdmin
```

### Mostra as métricas
```{r admin20}
func_resultados(rfMTryAdmin, testeAdmin$ChanceOfAdmit, testeAdmin)
```

### PREDIÇÕES DE NOVOS CASOS
```{r admin21}
dados_novos_casos <- read.csv("9 - Admissao - Novos Dados.csv")
##view(dados_novos_casos)
dados_novos_casos
```

### Resultados dos novos casos
```{r admin22}
set.seed(202400)
predict.rfCVAdmin <- predict(rfCVAdmin, dados_novos_casos)
dados_novos_casos$ChanceOfAdmit <- NULL
resultado <- cbind(dados_novos_casos, predict.rfCVAdmin)
##view(resultado)
resultado
```

### O Gráfico de Resíduos para a técnica/parâmetro de maior R2
```{r admin23}
Carregando a biblioteca ggplot2
library(ggplot2)

Criando um conjunto de dados de exemplo
set.seed(202400)
y_true <- dadosAdmin$ChanceOfAdmit # Valores Reais
y_pred <- predict(rfCVAdmin, dadosAdmin) # Valores Estimados

Criando o gráfico de resíduos
ggplot(data = data.frame(y_true, y_pred), aes(x = y_pred, y = y_true -
y_pred)) +
 geom_point() +
 geom_hline(yintercept = 0, color = "red") +
 labs(title = "Gráfico de Resíduos", x = "Valores Previstos", y = "Resíduos")
```

### Base: BIOMASSA

#### Leitura dos dados
```{r bio1}
setwd("D:/IA/IAA008/Trabalho")
dadosBio <- read.csv("5 - Biomassa - Dados.csv", header=T)
#view(dadosBio)

```

```

` ``
Cria arquivos de treino e teste
````{r bio2}
set.seed(202400)
indBio <- createDataPartition(dadosBio$biomassa, p=0.80, list = FALSE)
treinoBio <- dadosBio[indBio,]
testeBio <- dadosBio[-indBio,]
` ``

### KNN
````{r bio3}
tuneGridBio <- expand.grid(k = c(1,3,5,7,9))
set.seed(202400)
knnBio <- train(biomassa ~ ., data = treinoBio, method = "knn",
 tuneGrid=tuneGridBio)
knnBio
` ``

Mostra as métricas
````{r bio5}
func_resultados(knnBio, testeBio$biomassa, testeBio)
` ``

### Executa o RNA - Hold-out
### Treino com Hold-Out
````{r bio6}
set.seed(202400)
rnaBio <- train(biomassa~., data=treinoBio, method="nnet", linout=T,
 trace=FALSE)
rnaBio
` ``

Mostra as métricas
````{r bio8}
func_resultados(rnaBio, testeBio$biomassa, testeBio)
` ``

### Executa o RNA - Cross-Validation
### indica o método cv e numero de folders 10
````{r bio9}
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(size = seq(from = 1, to = 10, by = 1),
 decay = seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202400)
rnaCVBio <- train(biomassa~., data=treinoBio, method="nnet",
 trainControl=ctrl,
 tuneGrid=tuneGrid, linout=T, MaxNWts=10000, maxit=2000,
 trace=F)
rnaCVBio
` ``

Mostra as métricas
````{r bio10}
func_resultados(rnaCVBio, testeBio$biomassa, testeBio)
` ``

### Executa o SVM
### SVM - Hold-out
````{r bio11}
set.seed(202400)
svmBio <- train(biomassa~., data=treinoBio, method="svmRadial")

```

```

svmBio
```

#### Mostra as métricas
```{r bio13}
func_resultados(svmBio, testeBio$biomassa, testeBio)
```

#### SVM - Cross-validation
```{r bio14}
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202400)
svmCVBio <- train(biomassa~., data=treinoBio, method="svmRadial",
 trControl=ctrl)
svmCVBio
```

#### Mostra as métricas
```{r bio16}
func_resultados(svmCVBio, testeBio$biomassa, testeBio)
```

#### SVM - Cross validation com Vários C e sigma
```{r bio17}
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202400)
svmSigBio <- train(biomassa~., data=treinoBio, method="svmRadial",
 trControl=ctrl, tuneGrid=tuneGrid)
svmSigBio
```

#### Mostra as métricas
```{r bio19}
func_resultados(svmSigBio, testeBio$biomassa, testeBio)
```

#### Executar Randon Forest
#### RF - Hold-out
```{r bio20}
set.seed(202400)
rfBio <- train(biomassa~., data=treinoBio, method="rf")
rfBio
```

#### Mostra as métricas
```{r bio22}
func_resultados(rfBio, testeBio$biomassa, testeBio)
```

#### RF - Cross-validation
```{r bio23}
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202400)
rfCVBio <- train(biomassa~., data=treinoBio, method="rf", trControl=ctrl)
rfCVBio
```

#### Mostra as métricas
```{r bio25}
func_resultados(rfCVBio, testeBio$biomassa, testeBio)
```

```

```
#### RF - Cross Validation com Vários mtry
```{r bio26}
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202400)
rfMTryBio <- train(biomassa~., data=treinoBio, method="rf", trControl=ctrl,
 tuneGrid=tuneGrid)

rfMTryBio
```

### Mostra as métricas
```{r bio28}
func_resultados(rfMTryBio, testeBio$biomassa, testeBio)
```

### PREDIÇÕES DE NOVOS CASOS
```{r bio29}
dados_novos_casos <- read.csv("5 - Biomassa - Novos Dados.csv")
##view(dados_novos_casos)
dados_novos_casos
```

### Resultados de novos casos
```{r bio30}
set.seed(202400)
predict.rfMTryBio <- predict(rfMTryBio, dados_novos_casos)
dados_novos_casos$biomassa <- NULL
resultado <- cbind(dados_novos_casos, predict.rfMTryBio)
##view(resultado)
resultado
```

### O Gráfico de Resíduos para a técnica/parâmetro de maior R2
```{r bio31}
Carregando a biblioteca ggplot2
library(ggplot2)

Criando um conjunto de dados de exemplo
set.seed(202400)
y_true <- dadosBio$biomassa # Valores Reais
y_pred <- predict(rfMTryBio, dadosBio) # Valores Estimados

#Criando o gráfico de resíduos
ggplot(data = data.frame(y_true, y_pred), aes(x = y_pred, y = y_true -
y_pred)) +
 geom_point() +
 geom_hline(yintercept = 0, color = "red") +
 labs(title = "Gráfico de Resíduos", x = "Valores Previstos", y = "Resíduos")
...

AGRUPAMENTO - VEICULO

Leitura dos dados Veículos
```{r agrup0}
setwd("D:/IA/IAA008/Trabalho/11 - Agrupamento - Praticas - 4 - Veiculos/")
dadosAgrupVeic <- read.csv("4 - Veiculos - Dados.csv", header=T)
##view(dadosAgrupVeic)
```

Executa o Kmeans
```{r agrup1}
set.seed(202400)

```


APÊNDICE H - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

1 Classificação de Imagens (CNN)

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

# Carga da base
cifar10 = tf.keras.datasets.cifar10

# Já está separado em dados de treino e teste
# Não precisa separar
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalização os dados
# Imagens em pixels de 0 - 255
```

```

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

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

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

# Relatório sobre a arquitetura da rede
model.summary()

```

Model: "functional"

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

Total params: 1,284,170 (4.90 MB)

Trainable params: 1,284,170 (4.90 MB)

Non-trainable params: 0 (0.00 B)

```

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

```

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

Epoch 1/15

```
1563/1563 ----- 69s 42ms/step - accuracy:
0.3456 - loss: 1.7745 - val_accuracy: 0.5247 - val_loss: 1.3146
```

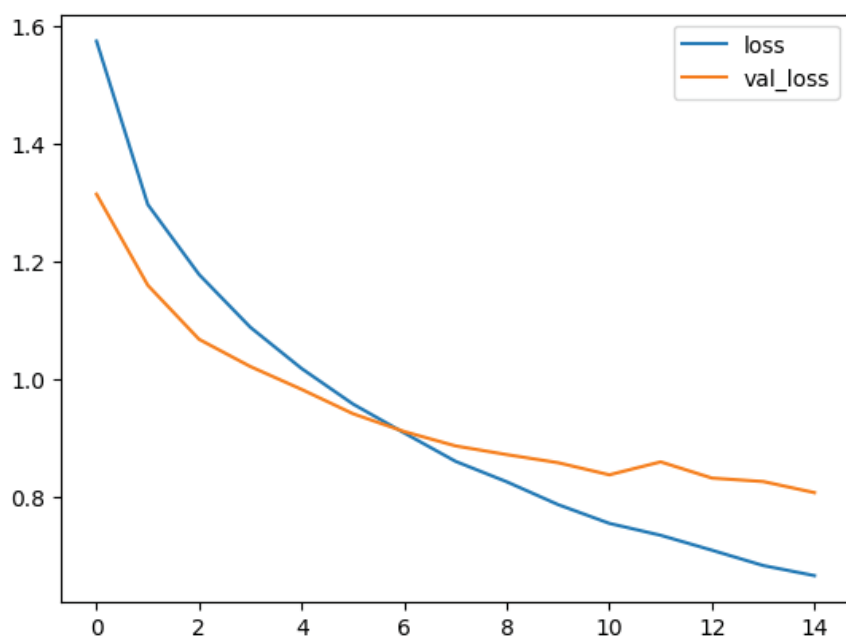
...

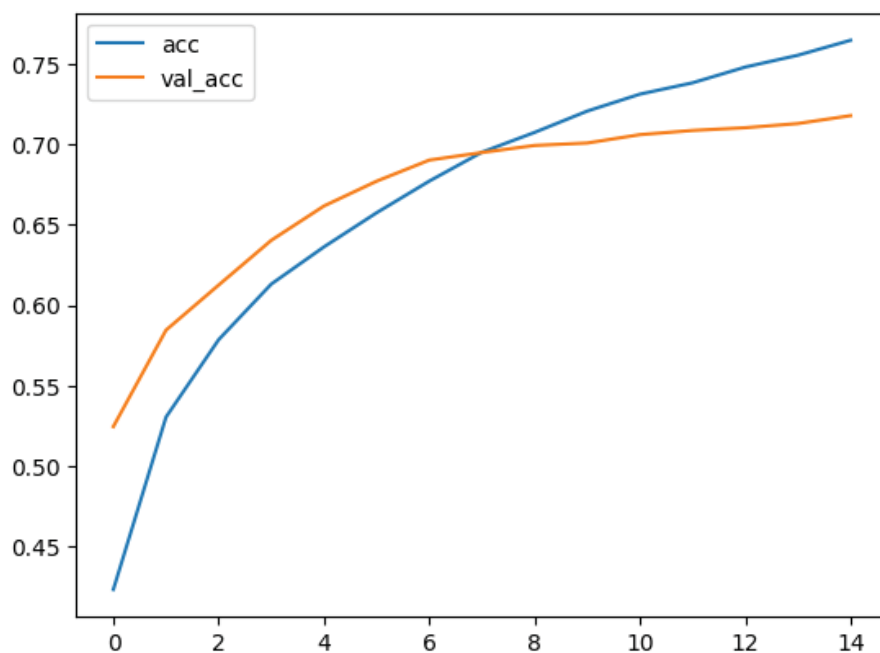
Epoch 15/15

```
1563/1563 ----- 82s 34ms/step - accuracy:
0.7673 - loss: 0.6544 - val_accuracy: 0.7178 - val_loss: 0.8071
```

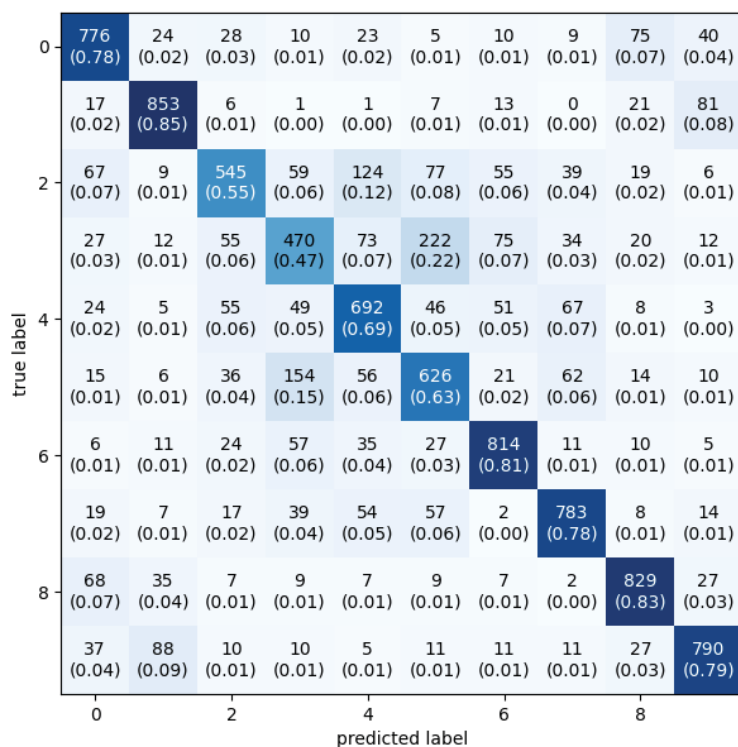
```
# Plotar a função de perda, treino e validação
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

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



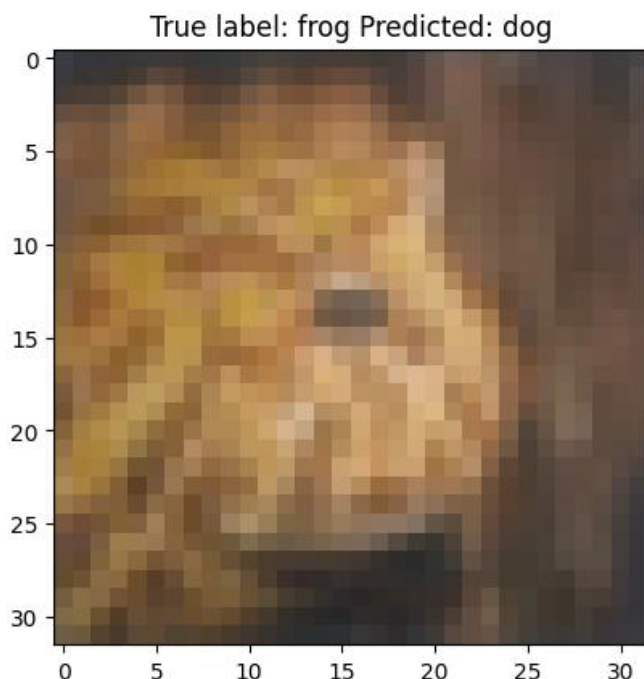


```
# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7), show_normed=True)
313/313 ————— 4s 11ms/step
(<Figure size 700x700 with 1 Axes>,
<Axes: xlabel='predicted label', ylabel='true label'>)
```



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

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



2 Detector de SPAM (RNN)

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

# carrega e arruma a base
!wget http://www.razer.net.br/datasets/spam.csv
df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values
--2024-08-06 22:24:13-- 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) [text/csv]
```

```
Saving to: 'spam.csv'
```

```
spam.csv          100%[=====>] 491.86K  --.-KB/s    in 0.1s
```

```
2024-08-06 22:24:14 (4.26 MB/s) - 'spam.csv' saved [503663/503663]
```

```
# Separa a base em treino e teste
```

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

```
# Número máximo de palavras para considerar
```

```
# São consideradas as mais frequentes, as demais são
```

```
# ignoradas
```

```
num_words = 20000
```

```
tokenizer = Tokenizer(num_words=num_words)
```

```
tokenizer.fit_on_texts(x_train)
```

```
sequences_train = tokenizer.texts_to_sequences(x_train)
```

```
sequences_test = tokenizer.texts_to_sequences(x_test)
```

```
word2index = tokenizer.word_index
```

```
V = len(word2index)
```

```
print("%s tokens" % V)
```

```
7196 tokens
```

```
# Acerta o tamanho das sequências (padding)
```

```
data_train = pad_sequences(sequences_train) # usa o tamanho da maior seq.
```

```
T = data_train.shape[1] # tamanho da sequência
```

```
data_test = pad_sequences(sequences_test, maxlen=T)
```

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

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

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

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

```
# Define o modelo
```

```
D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido
```

```
M = 5 # tamanho do hidden state, quantidade de unidades LSTM
```

```
i = Input(shape=(T,)) # Entra uma frase inteira
```

```
x = Embedding(V+1, D)(i)
```

```
x = LSTM(M)(x)
```

```
x = Dense(1, activation="sigmoid")(x) # Sigmoide pois só tem 2 valores
```

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

```
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 189)	0
embedding (Embedding)	(None, 189, 20)	143,940
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6

Total params: 144,466 (564.32 KB)

Trainable params: 144,466 (564.32 KB)

Non-trainable params: 0 (0.00 B)

```
# Compila e treina o modelo
model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])
epochs = 5
r = model.fit(data_train, y_train, epochs=epochs,
validation_data=(data_test,
y_test))
```

Epoch 1/5

117/117 _____ **19s** 90ms/step - accuracy: 0.8720
- loss: 0.5452 - val_accuracy: 0.8923 - val_loss: 0.3024

Epoch 2/5

117/117 _____ **8s** 70ms/step - accuracy: 0.9253 -
loss: 0.2666 - val_accuracy: 0.9668 - val_loss: 0.1745

Epoch 3/5

117/117 _____ **11s** 72ms/step - accuracy: 0.9831
- loss: 0.1462 - val_accuracy: 0.9788 - val_loss: 0.1207

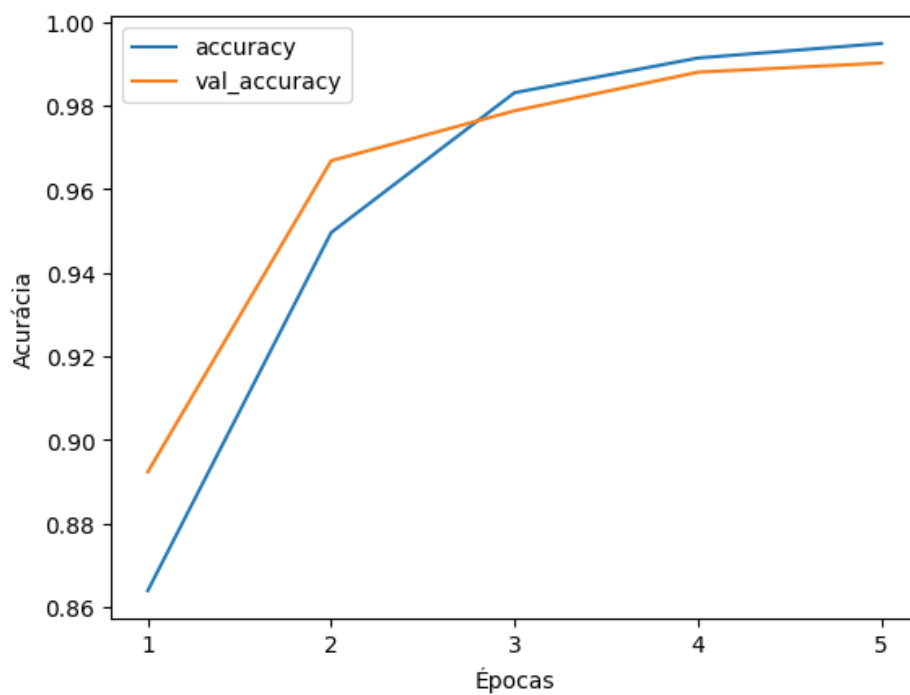
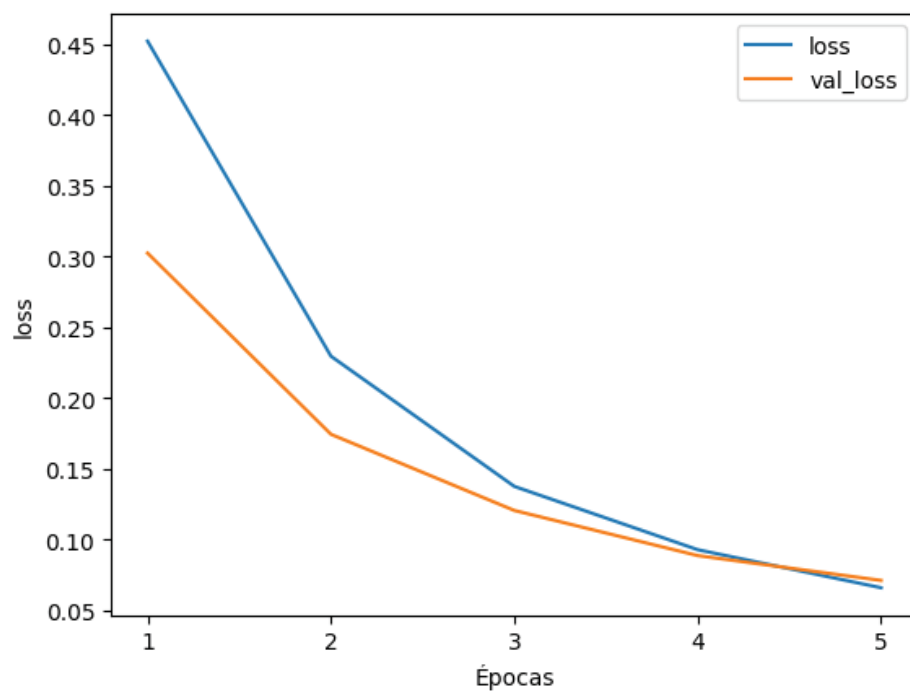
Epoch 4/5

117/117 _____ **10s** 85ms/step - accuracy: 0.9924
- loss: 0.0964 - val_accuracy: 0.9880 - val_loss: 0.0888

Epoch 5/5

117/117 _____ **10s** 80ms/step - accuracy: 0.9939
- loss: 0.0751 - val_accuracy: 0.9902 - val_loss: 0.0713

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



```
# Efetua a predição de um texto novo
#texto = "Hi, my name is Razer and want to tell you something."
texto = "Is your car dirty? Discover our new product. Free for all. Click
the link."
seq_texto = tokenizer.texts_to_sequences([texto]) # Tokeniza
data_texto = pad_sequences(seq_texto, maxlen=T) # Padding
pred = model.predict(data_texto) # Predição
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")
```

```
[[0.5785592]]
```

```
SPAM
```

3 Gerador de Dígitos Fake (GAN)

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

Successfully built tensorflow-docs
Installing collected packages: astor, tensorflow-docs
Successfully installed astor-0.8.1 tensorflow-docs-2024.7.15.51478

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

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

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11490434/11490434 ————— 2s 0us/step

# Cria o GERADOR
def make_generator_model():
    model = tf.keras.Sequential()

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

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

```

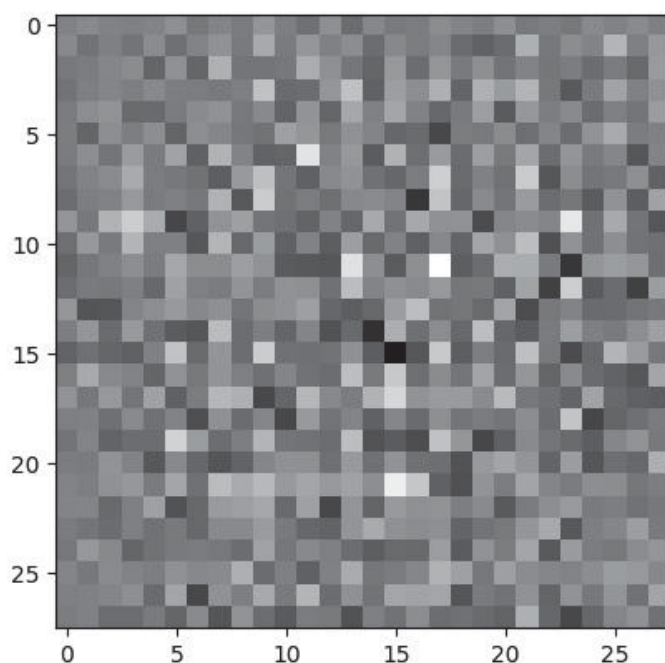
    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)

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

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

<matplotlib.image.AxesImage at 0x7e7b85e96020>

```



```

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

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))
    model.add(layers.Flatten())
    model.add(layers.Dense(1))
    return model

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

```

```

tf.Tensor([[ -0.00143388]], shape=(1, 1), dtype=float32)

# Define as funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

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

# Perda do GERADOR
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)

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

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

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

# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)

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

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

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

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

```

```

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

        for image_batch in dataset:
            train_step(image_batch)

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

        # Save the model every 15 epochs
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)
            print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time() -
start))

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

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

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

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

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

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

```



```
# Criar um GIF
# Display a single image using the epoch number
def display_image(epoch_no):
    return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))

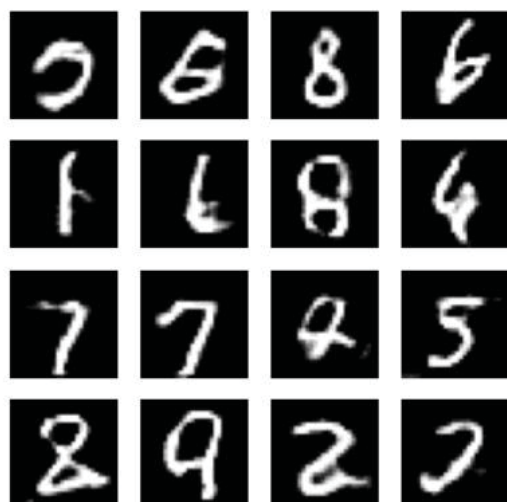
display_image(EPOCHS)
anim_file = 'dcgan.gif'

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

<ipython-input-16-aa6317a2e702>:13: DeprecationWarning: Starting with
ImageIO v3 the behavior of this function will switch to that of
iio.v3.imread. To keep the current behavior (and make this warning
disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread`
directly.
    image = imageio.imread(filename)
<ipython-input-16-aa6317a2e702>:16: DeprecationWarning: Starting with
ImageIO v3 the behavior of this function will switch to that of
iio.v3.imread. To keep the current behavior (and make this warning
disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread`
directly.
    image = imageio.imread(filename)
```



4 Tradutor de Textos (Transformer)

```
# Instalação e importação
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0

import collections
import logging
import os
import pathlib
import re
import string
import sys
import time
import numpy as np
import matplotlib.pyplot as plt
import tensorflow_datasets as tfds
import tensorflow_text as text
import tensorflow as tf

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

Found existing installation: tf_keras 2.17.0
Uninstalling tf_keras-2.17.0:
  Successfully uninstalled tf_keras-2.17.0
Successfully installed tensorflow-text-2.15.0 tf-keras-2.15.1

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

# Verificar o dataset
for pt_examples, en_examples in train_examples.batch(3).take(1):
    for pt in pt_examples.numpy():
```

```

    print(pt.decode('utf-8'))
    print()

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

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

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

# Tokenização e Destokenização do texto
model_name = "ted_hrlr_translate_pt_en_converter"

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

# Tem 2 tokenizers: um pt outro em en
# tokenizers.en tokeniza e detokeniza
tokenizers = tf.saved_model.load(model_name)

# PIPELINE DE ENTRADA
# Codificar/tokenizar lotes de texto puro
def tokenize_pairs(pt, en):
    pt = tokenizers.pt.tokenize(pt)
    # Converte ragged (irregular, tam variável) para dense
    # Faz padding com zeros.
    pt = pt.to_tensor()
    en = tokenizers.en.tokenize(en)
    # ragged -> dense
    en = en.to_tensor()
    return pt, en

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

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

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

```

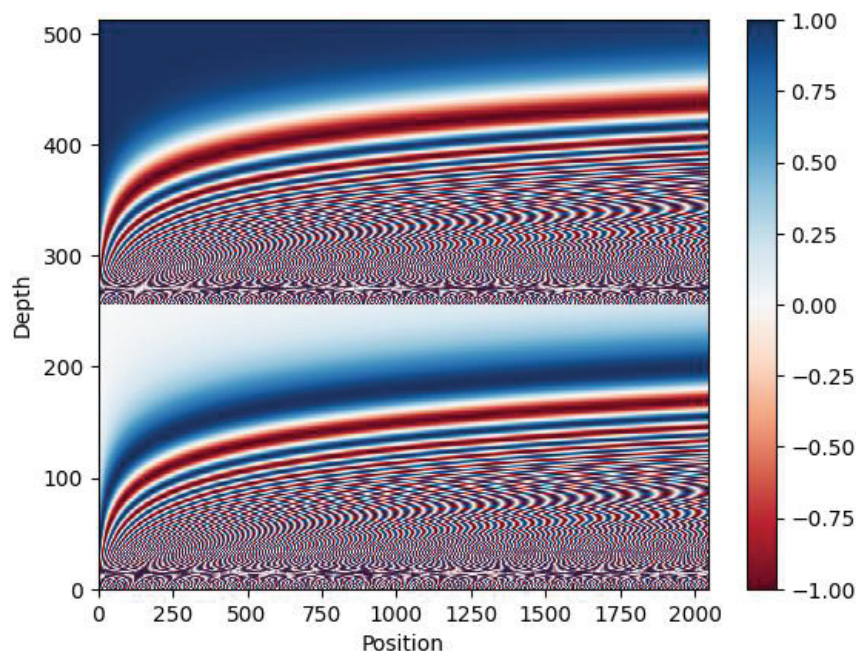
```
def positional_encoding(position, d_model):
    angle_rads = get_angles(np.arange(position)[:, np.newaxis],
                             np.arange(d_model) [np.newaxis, :],
                             d_model)
    # sin em índices pares no array; 2i
    angle_rads[:, 0::2] = np.sin(angle_rads[:,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

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

# Arrumar as dimensões
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))
plt.pcolormesh(pos_encoding, cmap='RdBu')

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

(1, 2048, 512)



```
# Cria uma máscara de 0 e 1, 0 para quando há valor e 1 quando não há
def create_padding_mask(seq):
    seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
    # add extra dimensions to add the padding
    # to the attention logits.
    return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_l)

# Máscara futura, usada no decoder
```

```

def create_look_ahead_mask(size):
    # zera o triângulo inferior
    mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
    return mask # (seq_len, seq_len)

# Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):
    # Q K^T
    matmul_qk = tf.matmul(q, k, transpose_b=True) # (... , seq_len_q, seq_len_k)

    # converte matmul_qk para float32
    dk = tf.cast(tf.shape(k)[-1], tf.float32)

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

    # Soma a máscara, e os valores faltantes serão um número próximo a -inf
    if mask is not None:
        scaled_attention_logits += (mask * -1e9)

    # softmax normaliza os dados, soman 1. // (... , seq_len_q, seq_len_k)
    attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
    output = tf.matmul(attention_weights, v) # (... , seq_len_q, depth_v)
    return output, attention_weights

# Atenção Multi-cabeças
class MultiHeadAttention(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.num_heads = num_heads
        self.d_model = d_model
        assert d_model % self.num_heads == 0
        self.depth = d_model // self.num_heads
        self.wq = tf.keras.layers.Dense(d_model)
        self.wk = tf.keras.layers.Dense(d_model)
        self.wv = tf.keras.layers.Dense(d_model)
        self.dense = tf.keras.layers.Dense(d_model)

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

    def call(self, v, k, q, mask):
        batch_size = tf.shape(q)[0]
        q = self.wq(q) # (batch_size, seq_len, d_model)
        k = self.wk(k) # (batch_size, seq_len, d_model)
        v = self.wv(v) # (batch_size, seq_len, d_model)

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

        # Calcula a atenção para cada cabeça (de forma matricial)
        # scaled_attention.shape == (batch_size, num_heads, seq_len_q, depth)

```

```

        # attention_weights.shape == (batch_size, num_heads, seq_len_q,
        seq_len_k)
        scaled_attention, attention_weights = scaled_dot_product_attention(q, k,
        v, mask)

        # Troca a dimensão 2 com 1, para acertar o num_heads
        # (batch_size, seq_len_q, num_heads, depth)
        scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])

        # Concatena os valores em: (batch_size, seq_len_q, d_model)
        concat_attention = tf.reshape(scaled_attention, (batch_size, -1,
        self.d_model))

        output = self.dense(concat_attention) # (batch_size, seq_len_q, d_model)

        return output, attention_weights

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

class EncoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.1):
        super(EncoderLayer, self).__init__()
        self.mha = MultiHeadAttention(d_model, num_heads)
        self.ffn = point_wise_feed_forward_network(d_model, dff)

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

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

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

        ffn_output = self.ffn(out1) # (batch_size, input_seq_len, d_model)
        ffn_output = self.dropout2(ffn_output, training=training)
        out2 = self.layernorm2(out1 + ffn_output) # (batch_size, input_seq_len,
        d_model)

        return out2

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

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

        self.ffn = point_wise_feed_forward_network(d_model, dff)

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

```

```

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

def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    # enc_output.shape == (batch_size, input_seq_len, d_model)

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

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

    ffn_output = self.ffn(out2) # (batch_size, target_seq_len, d_model)
    ffn_output = self.dropout3(ffn_output, training=training)
    out3 = self.layernorm3(ffn_output + out2) # (batch_size, target_seq_len,
d_model)

    return out3, attn_weights_block1, attn_weights_block2

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

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

class Decoder(tf.keras.layers.Layer):
    def __init__(self, num_layers, d_model, num_heads, dff, target_vocab_size,
maximum_position_encoding, rate=0.1):
        super(Decoder, self).__init__()

        self.d_model = d_model
        self.num_layers = num_layers

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

```

```

        self.pos_encoding = positional_encoding(maximum_position_encoding,
d_model)

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

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

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

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

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

        # x.shape == (batch_size, target_seq_len, d_model)
        return x, attention_weights

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

        self.decoder = Decoder(num_layers, d_model, num_heads, dff,
                                target_vocab_size,
                                pe_target, rate)

        self.final_layer = tf.keras.layers.Dense(target_vocab_size)

    def call(self, inputs, training):
        # Keras models prefer if you pass all your inputs in the first argument
        inp, tar = inputs

        enc_padding_mask, look_ahead_mask, dec_padding_mask =
self.create_masks(inp, tar)

        # (batch_size, inp_seq_len, d_model)
        enc_output = self.encoder(inp, training, enc_padding_mask)

        # dec_output.shape == (batch_size, tar_seq_len, d_model)
        dec_output, attention_weights = self.decoder(
            tar, enc_output, training, look_ahead_mask, dec_padding_mask)

        # (batch_size, tar_seq_len, target_vocab_size)
        final_output = self.final_layer(dec_output)

        return final_output, attention_weights

    def create_masks(self, inp, tar):
        # Encoder padding mask

```

```

enc_padding_mask = create_padding_mask(inp)

# Used in the 2nd attention block in the decoder.
# This padding mask is used to mask the encoder outputs.
dec_padding_mask = create_padding_mask(inp)

# Used in the 1st attention block in the decoder.
# It is used to pad and mask future tokens in the input received by
# the decoder.
look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])
dec_target_padding_mask = create_padding_mask(tar)
look_ahead_mask = tf.maximum(dec_target_padding_mask, look_ahead_mask)
return enc_padding_mask, look_ahead_mask, dec_padding_mask

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

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

    def __call__(self, step):
        step = tf.cast(step, tf.float32) # Adicionado para evitar ERRO
        arg1 = tf.math.rsqrt(step)
        arg2 = step * (self.warmup_steps ** -1.5)
        return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

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

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')

def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask
    return tf.reduce_sum(loss_) / tf.reduce_sum(mask)

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

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

transformer = Transformer(

```

```

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

# Checkpoint
checkpoint_path = "./checkpoints/train"

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

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

# if a checkpoint exists, restore the latest checkpoint.
if ckpt_manager.latest_checkpoint:
    ckpt.restore(ckpt_manager.latest_checkpoint)
    print('Latest checkpoint restored!!')

EPOCHS = 20
train_step_signature = [
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]
@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
    tar_inp = tar[:, :-1]
    tar_real = tar[:, 1:]
    with tf.GradientTape() as tape:
        predictions, _ = transformer([inp, tar_inp], training = True)
        loss = loss_function(tar_real, predictions)

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

for epoch in range(EPOCHS):
    start = time.time()
    train_loss.reset_state()
    train_accuracy.reset_state()
    # inp -> portuguese, tar -> english
    for (batch, (inp, tar)) in enumerate(train_batches):
        train_step(inp, tar)
        if batch % 50 == 0:
            print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f}
Accuracy {train_accuracy.result():.4f}')

    if (epoch + 1) % 5 == 0:
        ckpt_save_path = ckpt_manager.save()
        print(f'Saving checkpoint for epoch {epoch+1} at {ckpt_save_path}')
        print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}')
        print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')

Epoch 1 Batch 0 Loss 8.8714 Accuracy 0.0000
...

```

Saving checkpoint for epoch 20 at [./checkpoints/train/ckpt-4](#)

Epoch 20 Loss 1.4513 Accuracy 0.6788

Time taken for 1 epoch: 97.32 secs

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

    def __call__(self, sentence, max_length=20):
        # input sentence is portuguese, hence adding the start and end token
        assert isinstance(sentence, tf.Tensor)

        if len(sentence.shape) == 0:
            sentence = sentence[tf.newaxis]
            sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
            encoder_input = sentence

        # as the target is english, the first token to the transformer should be
the
        # english start token.
        start_end = self.tokenizers.en.tokenize([''])[0]
        start = start_end[0][tf.newaxis]
        end = start_end[1][tf.newaxis]
        output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
        output_array = output_array.write(0, start)

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

        output = tf.transpose(output_array.stack())
        # output.shape (1, tokens)
        text = tokenizers.en.detokenize(output)[0]
        tokens = tokenizers.en.lookup(output)[0]
        _, attention_weights = self.transformer([encoder_input, output[:, :-1]],
training=False)
        return text, tokens, attention_weights
translator = Translator(tokenizers, transformer)

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

Prediction      : b'i read about triba hops in encyclody .'
```

APÊNDICE I - BIG DATA

A – ENUNCIADO

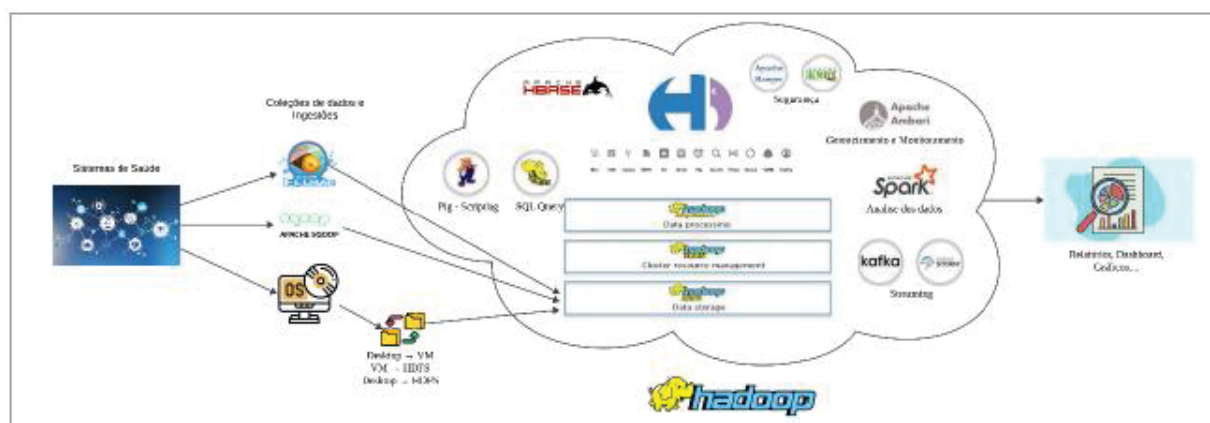
Enviar um arquivo PDF contendo uma descrição breve (2 páginas) sobre a implementação de uma aplicação ou estudo de caso envolvendo Big Data e suas ferramentas (NoSQL e NewSQL). Caracterize os dados e Vs envolvidos, além da modelagem necessária dependendo dos modelos de dados empregados.

B – RESOLUÇÃO

Introdução

No cenário atual da saúde, a integração de tecnologias e sistemas de saúde para o atendimento médico tem se tornado uma prioridade crescente, principalmente devido à abundância de dados gerados diariamente nos prestadores de serviços como clínicas, hospitais e laboratórios, como registros eletrônicos de saúde, resultados de exames laboratoriais, imagens médicas e prontuários eletrônicos. A explosão de informações exige soluções inovadoras que transformem dados em insights valiosos para antecipar tratamentos, aprimorar diagnósticos e otimizar cuidados.

Nesse contexto, o Hadoop surge como uma das arquiteturas mais robustas e escaláveis para lidar com grandes volumes de dados, possibilitando o armazenamento e processamento de informações de forma eficiente. Combinando o uso de Big Data e Inteligência Artificial (IA), essa solução oferece diagnósticos mais precisos e personalizados, além de uma gestão mais eficiente das informações dos sistemas de saúde. Este texto explora como a implementação de uma solução de diagnóstico automatizado, baseada na arquitetura Hadoop e alimentada por dados detalhados dos históricos médicos dos pacientes, pode melhorar e antecipar o processo de diagnóstico, abordando os desafios dos sistemas tradicionais e destacando os benefícios esperados para a prática médica e o atendimento ao paciente.



1. Coleta dos dados

Para a implementação eficaz de uma solução de diagnóstico automatizado baseada em Big Data, é essencial coletar uma vasta gama de informações detalhadas. Entre os dados necessários estão os registros eletrônicos de saúde dos pacientes, que incluem dados estruturados como: receituário,

atestados e semi-estruturados relacionados a exame de imagens. Essas informações encontram-se distribuídas em diferentes sistemas médicos, laboratoriais e farmacêuticos.

Ademais, a arquitetura Hadoop permite a coleta e armazenamento de todos esses dados em um ambiente distribuído, utilizando o Hadoop Distributed File System (HDFS) para garantir a escalabilidade e a eficiência no armazenamento de grandes volumes de dados heterogêneos. O sistema é capaz de lidar com diferentes tipos de informações, o que o torna ideal para integrar dados médicos variados, como históricos de tratamento, demográficos e de estilo de vida dos pacientes, possibilitando a identificação de padrões que melhoram a precisão dos diagnósticos.

2. Processamento

No contexto de um diagnóstico automatizado, o Hadoop é adequado para o processamento eficiente de grandes volumes de dados. A arquitetura Hadoop utiliza o paradigma de processamento distribuído conhecido, permitindo a execução de tarefas de forma paralela em diferentes nós da rede, otimizando o tempo de processamento e garantindo que grandes volumes de dados possam ser analisados rapidamente. Inicialmente, os dados médicos são coletados de diversas fontes e armazenados no HDFS, que distribui e replica as informações para garantir redundância e acessibilidade. Em seguida, os dados passam por um processo de limpeza e pré-processamento para remover inconsistências, duplicidades e erros, garantindo a integridade dos dados antes da análise.

3. Inteligência Artificial para Diagnóstico Personalizado

A análise de Big Data, realizada por meio de algoritmos de machine learning e técnicas estatísticas, é aplicada para identificar padrões, correlações e tendências. No Hadoop, essas análises são processadas por meio do Apache Spark, que permite a execução de algoritmos complexos em tempo real. Essa arquitetura possibilita a criação de modelos preditivos que podem sugerir diagnósticos pelo Apache Mahout com base no histórico médico e nos dados atuais dos pacientes. Técnicas de processamento de linguagem natural (NLP) e tratamento de imagens também podem ser aplicadas para extrair informações úteis de exames e relatórios médicos.

4. Implementação

Uma vez que os dados estão centralizados no HDFS, a plataforma de IA pode ser integrada ao sistema de gestão hospitalar por meio de APIs e interfaces de programação que permitem a comunicação com outras aplicações e módulos hospitalares. A IA processa os dados e gera insights e recomendações de diagnóstico, que são incorporados aos fluxos de trabalho clínicos existentes. Os médicos têm acesso a dashboards detalhados, que visualizam os resultados das análises realizadas pelo Hadoop e pelos algoritmos de IA facilitando a interpretação dos dados e a tomada de decisões baseadas em insights acionáveis.

5. Conclusão

A implementação de uma solução de diagnóstico automatizado baseada na arquitetura Hadoop, aliada à Inteligência Artificial, representa um avanço significativo para o setor de saúde. O Hadoop permite o processamento escalável de grandes volumes de dados médicos, enquanto a IA oferece diagnósticos mais precisos e personalizados. Com a centralização de dados no HDFS e a aplicação de técnicas avançadas de análise, como machine learning e NLP, os hospitais podem melhorar significativamente a prática clínica.

Além disso, a conformidade com a LGPD assegura que essa inovação ocorra de maneira ética, protegendo a privacidade dos pacientes e estabelecendo uma relação de confiança. Ao integrar novos dados continuamente e refinar os modelos preditivos, os hospitais estão preparados para oferecer cuidados médicos mais assertivos e adaptados, melhorando a qualidade do atendimento e os resultados para os pacientes.

APÊNDICE J - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

1 Extração de Características

Setup

```
# Caminho para a base de dados
train_directory = r'C:\Users\opedr\Desktop\images\Train_4cls_amostra'
test_directory = r'C:\Users\opedr\Desktop\images\Test_4cl_amostra'

# importações
import os
import cv2 print(cv2.__version__)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import csv
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, recall_score, f1_score, confusio
from sklearn.svm import SVC
from skimage.feature import local_binary_pattern
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import Dense, Flatten, Dropout, GlobalAveragePo
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array, load_img
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import preprocess_input as vgg16_pr
from keras.applications.resnet import preprocess_input as resnet50_preproces
```

Carregue a base de dados de Treino

```
# Função para carregar a base de dados
def load_image_paths_and_labels(data_directory):
    image_paths = []
    labels = []
    for class_name in os.listdir(data_directory):
        class_path = os.path.join(data_directory, class_name)
        if os.path.isdir(class_path):
            for img_file in os.listdir(class_path):
                img_path = os.path.join(class_path, img_file)
                image_paths.append(img_path)
                labels.append(class_name)

    return image_paths, labels

image_paths, labels = load_image_paths_and_labels(train_directory)
```

```
print(f'Total de imagens carregadas: {len(image_paths)}')
```

Total de imagens carregadas: 593

Crie partições contendo 80% para treino e 20% para validação (atenção aos pacientes)

```
train_labels.append(class_label)
train_patient_ids.append(patient)
```

```
# Adicionar as imagens de validação
```

```
for patient in val_patients:
    for img_path in patient_images[patient]:
        val_image_paths.append(img_path)
        val_labels.append(class_label)
        val_patient_ids.append(patient)
    return (train_image_paths, train_labels), (val_image_paths, val_labels)
```

```
from sklearn.model_selection import train_test_split
from collections import defaultdict
import os
```

```
# Função para separar a base em treino e validação por classe
```

```
def split_train_val(image_paths, labels, test_size=0.2, random_state=42):
    # Organizar as imagens por paciente e por classe
    class_patient_images = defaultdict(lambda: defaultdict(list))
    for img_path, label in zip(image_paths, labels):
        patient_id = os.path.basename(img_path).split('_')[0] # Extrai o
        número
        class_patient_images[label][patient_id].append(img_path) #
    Agrupar por classe
```

```
    train_image_paths, train_labels, train_patient_ids = [], [], []
    val_image_paths, val_labels, val_patient_ids = [], [], []
```

```
# Para cada classe, dividir os pacientes entre treino e validação
```

```
for class_label, patient_images in class_patient_images.items():
    patients = list(patient_images.keys())
```

```
# Dividir os pacientes dessa classe em treino e validação
```

```
train_patients, val_patients = train_test_split(patients,
    test_size=test_size, random_state=random_state)
```

```
# Adicionar as imagens de treino
```

```
for patient in train_patients:
    for img_path in patient_images[patient]:
        train_image_paths.append(img_path)
```

```
    return (train_image_paths, train_labels), (val_image_paths,
    val_labels)
```

```
# Separar as imagens por pacientes em treino e validação (80% treino, 20% va
(train_paths, train_labels), (val_paths, val_labels) =
split_train_val(image_paths, labels)
print(f"Imagens de treino: {len(train_paths)}, Imagens de validação:
{len(val_paths)}")
```

```
# Separar as imagens por pacientes em treino e validação (80% treino, 20% va
(train_paths,      train_labels),      (val_paths,      val_labels)      =
split_train_val(image_paths, labels)
print(f"Imagens de treino: {len(train_paths)}, Imagens de validação:
{len(val_paths)}")
```

Imagens de treino: 473, Imagens de validação: 120

Imagens de treino: 473, Imagens de validação: 120

Extraia características utilizando LBP e a CNN VGG16 (gerando um csv paracada extrator)

LBP

```
# Definir os parâmetros para o cálculo do LBP
n_points = 24
radius = 3
METHOD = 'uniform'

# Função para calcular LBP em uma única imagem
def calculate_lbp(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    lbp = local_binary_pattern(image, n_points, radius, METHOD)
    lbp_hist, _ = np.histogram(lbp.ravel(),
    bins=np.arange(0, n_points + 3)
    lbp_hist = lbp_hist.astype('float')
    lbp_hist /= (lbp_hist.sum() + 1e-6)
    return lbp_hist
```

```
# Função para calcular LBP para o conjunto de imagens
def apply_lbp_to_dataset(image_paths):
    lbp_features = [calculate_lbp(image_path)
    for image_path in image_paths
    return np.array(lbp_features)
```

```
# Aplicar LBP no conjunto de treino e teste
train_lbp = apply_lbp_to_dataset(train_paths)
test_lbp = apply_lbp_to_dataset(val_paths)
print(f'Características LBP do conjunto de treino: {train_lbp.shape}')
print(f'Características LBP do conjunto de teste: {test_lbp.shape}')
```

Características LBP do conjunto de treino: (473, 26)

Características LBP do conjunto de teste: (120, 26)

CNN

VGG16

```
# Carregar o modelo VGG16 pré-treinado
base_model = VGG16(weights='imagenet', include_top=True)
# Criar um novo modelo que vai até a penúltima camada
vgg_model = Model(inputs=base_model.input, outputs=base_model.layers[-2].out

# Função para extrair características usando VGG16
def extract_vgg_features(image_path):
    image = load_img(image_path, target_size=(224, 224))
    image_array = img_to_array(image)
    np.expand_dims(image_array, axis=0)
    image_array = preprocess_input(image_array)
    features = vgg_model.predict(image_array)
    return features.flatten()
```

```

train_vgg = [extract_vgg_features(img_path)
for img_path in train_paths] test_vgg = [extract_vgg_features(img_path)
for img_path in val_paths]
print(f'CNN VGG16')
print(f'Conjunto de treino: {len(train_vgg)}')
print(f'Conjunto de teste: {len(test_vgg)}')

```

```

119ms/step 1/1 ----- 0s 134ms/step
0s 117ms/step 123ms/step
135ms/step
153ms/step
124ms/step
117ms/step
114ms/step
118ms/step
128ms/step
135ms/step
120ms/step
132ms/step
130ms/step
132ms/step
116ms/step
118ms/step
127ms/step
121ms/step
116ms/step 1/1 ----- 0s 132ms/step
1/1 ----- 0s 119ms/step

```

CNN VGG16

Conjunto de treino: 473

Conjunto de teste: 120

Exportar como CSV

```

def export_features_to_csv(features, labels, output_csv):
    with open(output_csv, mode="w", newline="") as file:
        writer = csv.writer(file)

        # Header: features + label
        header = [f"feature_{i}" for i in range(len(features[0]))] +
        ['label']
        writer.writerow(header)

        # Escreve as características e labels
        for feature, label in zip(features, labels):
            writer.writerow(list(feature) + [label])

    print(f"Características, labels e IDs de pacientes salvos em {output_csv}")

# Exportação das características como CSV
export_features_to_csv(train_lbp, train_labels,
output_csv="lbp_features_train.csv")
export_features_to_csv(test_lbp, val_labels,
output_csv="lbp_features_validation.csv")

export_features_to_csv(train_vgg, train_labels,
output_csv="vgg_features_train.csv")
export_features_to_csv(test_vgg, val_labels,
output_csv="vgg_features_validation.csv")

```

Características, labels e IDs de pacientes salvos em lbp_features_train.csv
 Características, labels e IDs de pacientes salvos em lbp_features_validation.csv

Características, labels e IDs de pacientes salvos em vgg_features_train.csv
 Características, labels e IDs de pacientes salvos em vgg_features_validation.csv

Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos.

Random Forest

```
# Treinar Random Forest com LBP
rf_model_lbp = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_lbp.fit(train_lbp, train_labels)
y_pred_rf_lbp = rf_model_lbp.predict(test_lbp)

# Treinar Random Forest com VGG
rf_model_vgg = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_vgg.fit(train_vgg, train_labels)
y_pred_rf_vgg = rf_model_vgg.predict(test_vgg)
```

SVM

```
# Treinar SVM com LBP
svm_model_lbp = SVC(kernel='linear', random_state=42)
svm_model_lbp.fit(train_lbp, train_labels)
y_pred_svm_lbp = svm_model_lbp.predict(test_lbp)

# Treinar SVM com VGG
svm_model_vgg = SVC(kernel='linear', random_state=42)
svm_model_vgg.fit(train_vgg, train_labels)
y_pred_svm_vgg = svm_model_vgg.predict(test_vgg)
```

RNA

```
# Treinar RNA com LBP
rna_model_lbp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=42)
rna_model_lbp.fit(train_lbp, train_labels)
y_pred_rna_lbp = rna_model_lbp.predict(test_lbp)
c:\Users\opedr\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:781:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached
and the optimization has n't converged yet.
warnings.warn()
```

Carregue a base de Teste e execute a tarefa 3 nesta base

Carregar a Base de Teste

```
# Carregamento da base de teste
image_paths, labels = load_image_paths_and_labels(test_directory)
print(f'Total de imagens carregadas: {len(image_paths)}')
```

Total de imagens carregadas: 371

LBP

```
# Aplicação de LBP à base de teste
test_lbp = apply_lbp_to_dataset(image_paths)
```

CNN VGG16

```
# Aplicação de VGG16 à base de teste
test_vgg = [extract_vgg_features(img_path)
for img_path in image_paths]
```

```
135ms/step
135ms/step
130ms/step
155ms/step
137ms/step
137ms/step
139ms/step
135ms/step
126ms/step 1/1 ————— 0s 132ms/step
1/1 ————— 0s 117ms/step
```

Exportar como CSV

```
# Exportação das características como CSV
export_features_to_csv(test_lbp, labels, output_csv="lbp_features_test.csv")
export_features_to_csv(test_vgg, labels, output_csv="lbp_features_test.csv")
```

Características, labels e IDs de pacientes salvos em lbp_features_test.csv
 Características, labels e IDs de pacientes salvos em lbp_features_test.csv

Aplique os modelos treinados nos dados de teste

Random Forest

```
# Random Forest com LBP
y_pred_rf_lbp = rf_model_lbp.predict(test_lbp)
# Random Forest com VGG
y_pred_rf_vgg = rf_model_vgg.predict(test_vgg)
```

SVM

```
# SVM com LBP y_pred_svm_lbp = svm_model_lbp.predict(test_lbp)
# SVM com VGG y_pred_svm_vgg = svm_model_vgg.predict(test_vgg)
```

RNA

```
# RNA com LBP
y_pred_rna_lbp = rna_model_lbp.predict(test_lbp)

# RNA com VGG
y_pred_rna_vgg = rna_model_vgg.predict(test_vgg)
```

7. Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão

```
# Cálculo da especificidade
def specificity_score(y_true, y_pred):
```

```

    cm = confusion_matrix(y_true, y_pred)
    tn = cm.diagonal().sum()
    # Soma da diagonal dos verdadeiros negativos
    fp = cm.sum(axis=0) - cm.diagonal()
    # Falsos positivos
    return tn / (tn + fp.sum())

```

```

# Cálculo das métricas
def exibir_metricas(y_test, y_pred):
    accuracy = accuracy_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred, average='macro')
    specificity = specificity_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='macro')
    conf_matrix = confusion_matrix(y_test, y_pred)

    print(f'Acurácia: {accuracy:.4f}')
    print(f'Sensibilidade (Recall): {recall:.4f}')
    print(f'Especificidade: {specificity:.4f}')
    print(f'F1-Score: {f1:.4f}')

    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.title('Matriz de Confusão')
    plt.xlabel('Previsão')
    plt.ylabel('Verdadeiro')
    plt.show()

```

Random Forest

```

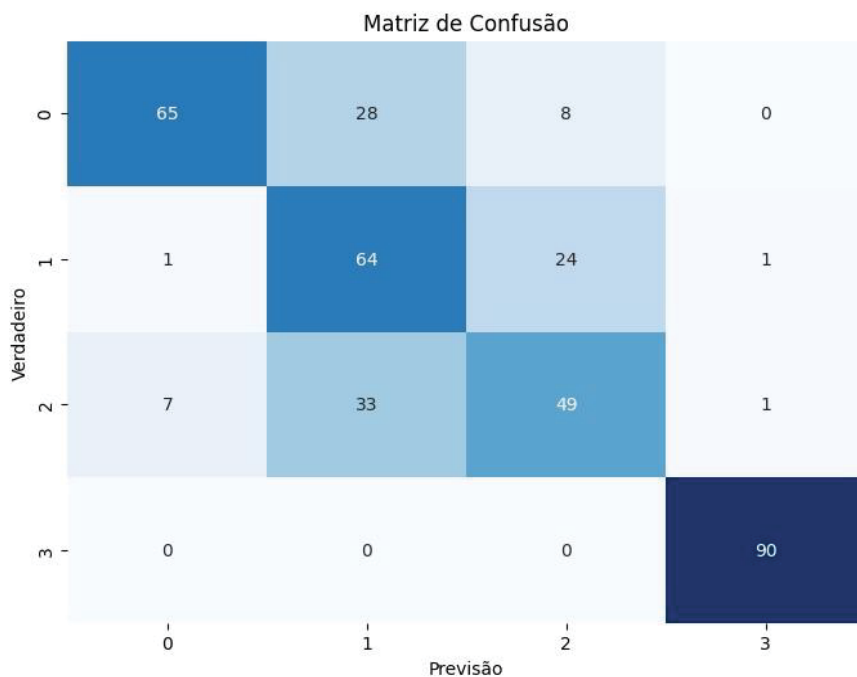
print(f'Métricas para Random Forest usando LBP:') exibir_metricas(labels,
y_pred_rf_lbp)
print(f'Métricas para Random Forest usando VGG:') exibir_metricas(labels,
y_pred_rf_vgg)

```

```

Métricas para Random Forest usando LBP:
Acurácia: 0.7224
Sensibilidade (Recall): 0.7248
Especificidade: 0.7224
F1-Score: 0.7261

```



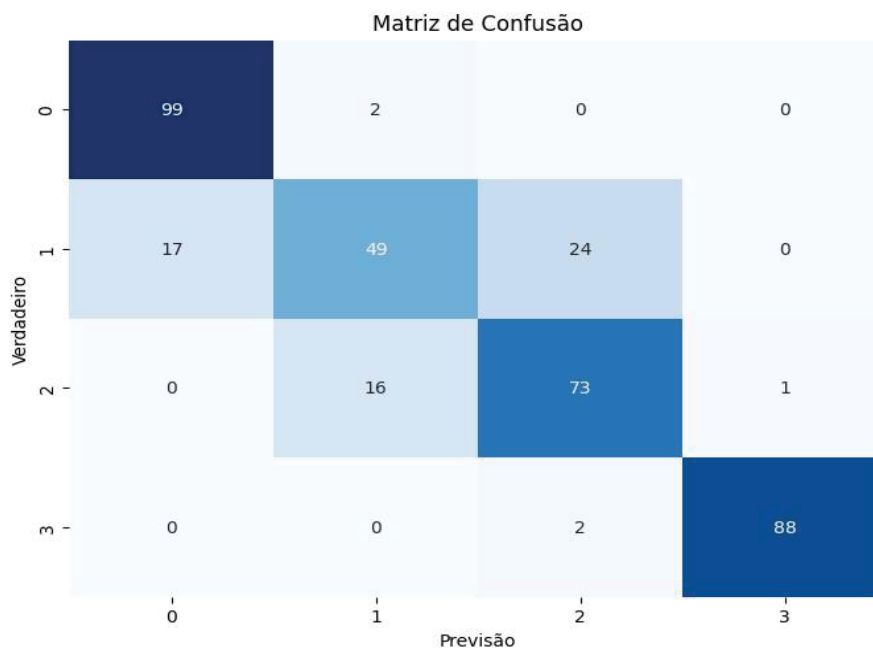
Métricas para Random Forest usando VGG:

Acurácia: 0.8329

Sensibilidade (Recall): 0.8284

Especificidade: 0.8329

F1-Score: 0.8231



SVM

```
print(f'Métricas para SVM usando LBP:')  exibir_metricas(labels,
y_pred_svm_lbp)
print(f'Métricas para SVM usando VGG:')  exibir_metricas(labels,
y_pred_svm_vgg)
```

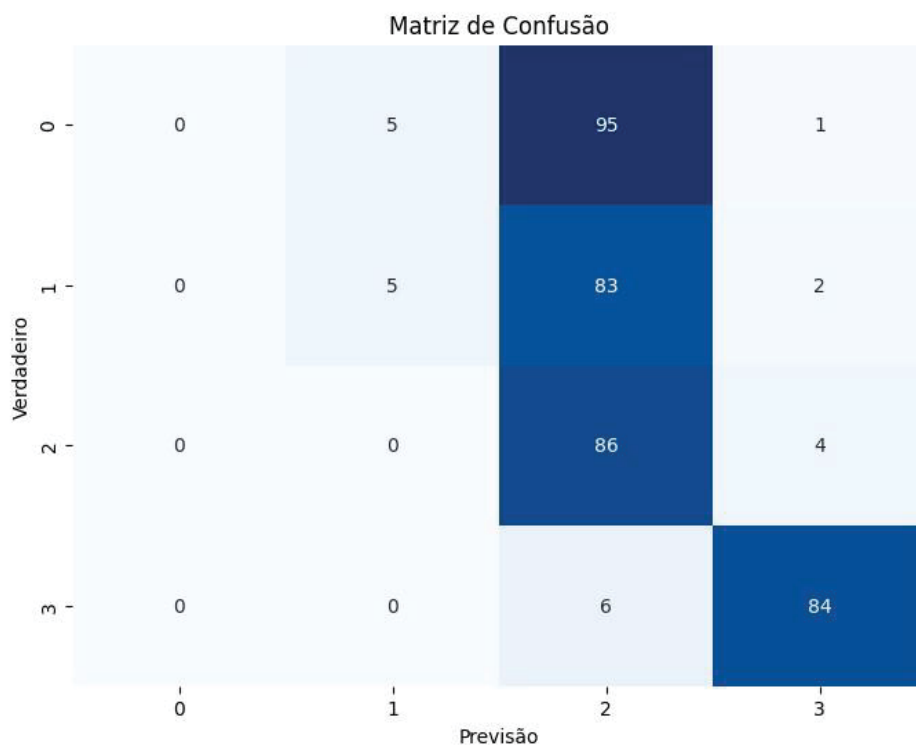
Métricas para SVM usando LBP:

Acurácia: 0.4717

Sensibilidade (Recall): 0.4861

Especificidade: 0.4717

F1-Score: 0.3765



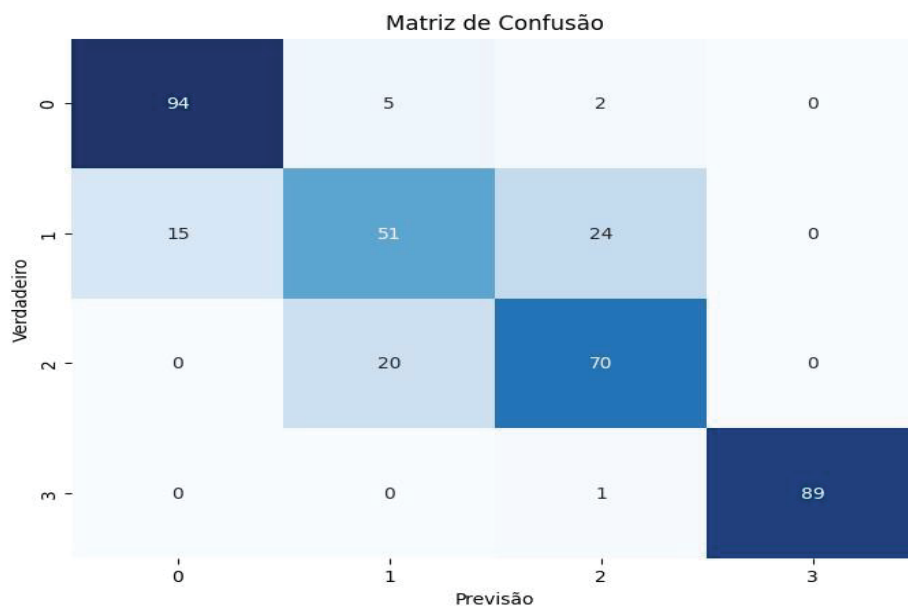
Métricas para SVM usando VGG:

Acurácia: 0.8194

Sensibilidade (Recall): 0.8160

Especificidade: 0.8194

F1-Score: 0.8132



RNA

```
print(f'Métricas para RNA usando LBP:')    exibir_metricas(labels,
y_pred_rna_lbp)
print(f'Métricas para RNA usando VGG:')    exibir_metricas(labels,
y_pred_rna_vgg)
```

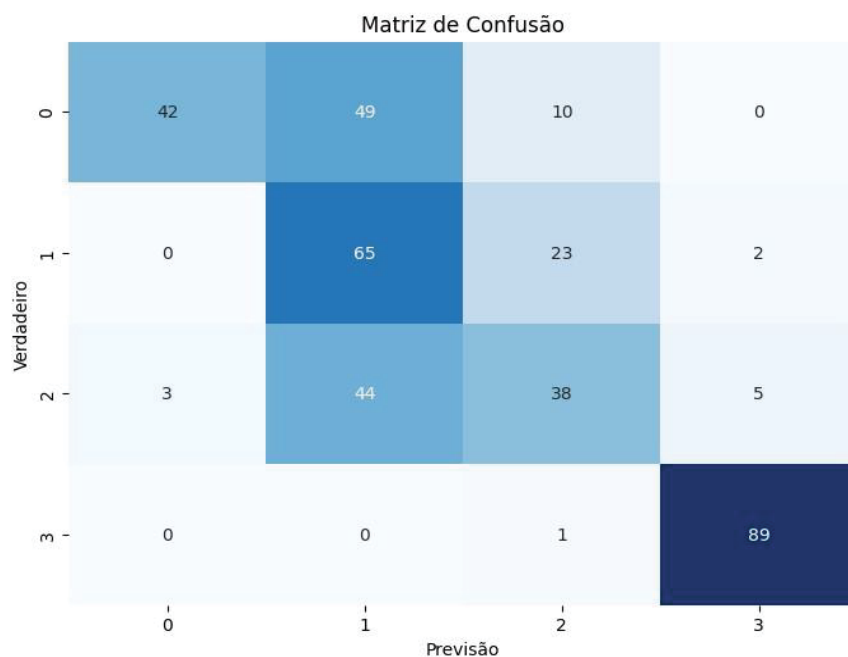
Métricas para RNA usando LBP:

Acurácia: 0.6307

Sensibilidade (Recall): 0.6373

Especificidade: 0.6307

F1-Score: 0.6314



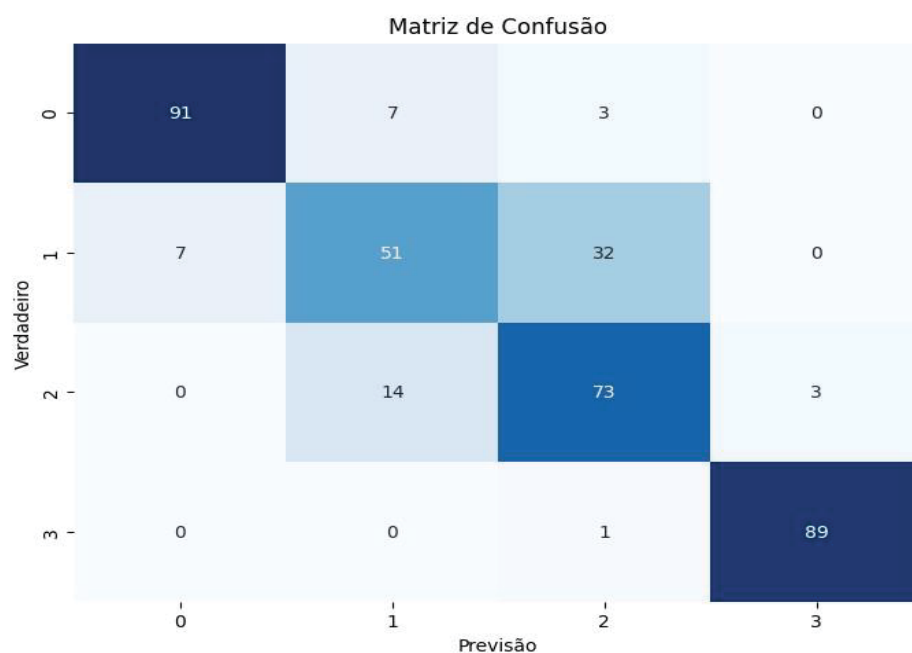
Métricas para RNA usando VGG:

Acurácia: 0.8194

Sensibilidade (Recall): 0.8169

Especificidade: 0.8194

F1-Score: 0.8140



8. Indique qual modelo da o melhor o resultado e a métrica utilizada

Com base na análise das métricas de desempenho, o modelo de IA Random Forest aplicado ao VGG se destacou em relação aos demais em todos os demais. A acurácia do modelo foi consistentemente superior, indicando uma maior proporção de classificações corretas no conjunto de dados. Além disso, a sensibilidade (Recall) do Random Forest aplicado ao VGG mostrou um desempenho mais eficiente na identificação dos casos positivos, enquanto sua especificidade evidenciou uma melhor capacidade de evitar falsos positivos.

O F1-Score, que equilibra precisão e recall, também favoreceu o modelo mencionado, demonstrando um melhor equilíbrio entre as métricas. A matriz de confusão reforçou essa conclusão, mostrando menos erros de classificação em todas as categorias.

2 Redes Neurais

Utilize a base de dados de Treino já separadas em treino e validação do exercício anterior

```
# Carregar imagens
image_paths, labels = load_image_paths_and_labels(train_directory)
print(f'Total de imagens carregadas: {len(image_paths)}')

# Separar as imagens por pacientes em treino e validação (80% treino, 20%
validação)
(train_paths, train_labels), (val_paths, val_labels) =
split_train_val(image_paths, labels)
print(f"Imagens de treino: {len(train_paths)}, Imagens de validação:
{len(val_paths)}")
```

Total de imagens carregadas: 593

Imagens de treino: 473, Imagens de validação: 120

Treine modelos VGG16 e Resnet50 adaptadas com e sem Data Augmentation

Criação dos Modelos

```
# VGG16
def create_vgg16_model(input_shape, num_classes):
    base_model = VGG16(weights='imagenet', include_top=False, input_shape=input_shape)
    # Congelar as camadas da VGG16 para não treinar novamente
    for layer in base_model.layers:
        layer.trainable = False

    # Adicionando camadas customizadas (fully connected)
    x = base_model.output
    x = Flatten()(x)
    x = Dense(512, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(num_classes, activation='softmax')(x)

    # Modelo final
    model = Model(inputs=base_model.input, outputs=output)

    return model

# Resnet50
def create_resnet50_model(input_shape, num_classes):
    base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=input_shape)
    # Congelar as camadas da ResNet50
    for layer in base_model.layers:
        layer.trainable = False

    # Adicionando camadas customizadas
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)
```

```

        x = Dropout(0.5)(x)
        output = Dense(num_classes, activation='softmax')(x)

        # Modelo final
        model = Model(inputs=base_model.input, outputs=output)
        model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

        return model

# optimizer = Adam(learning_rate=0.0001) l_rate = 0.00001
num_classes = 4
input_shape = (224, 224, 3)

# Criar e compilar VGG
vgg16_model = create_vgg16_model(input_shape, num_classes)
vgg16_aug_model = create_vgg16_model(input_shape, num_classes)
vgg16_model.compile(optimizer=Adam(learning_rate=l_rate),
loss='categorical_crossentropy', metrics=['accuracy'])
vgg16_aug_model.compile(optimizer=Adam(learning_rate=l_rate),
loss='categorical_crossentropy', metrics=['accuracy'])

# Criar e compilar Resnet50
resnet50_model = create_resnet50_model(input_shape, num_classes)
resnet50_aug_model = create_resnet50_model(input_shape, num_classes)
resnet50_model.compile(optimizer=Adam(learning_rate=l_rate),
loss='categorical_crossentropy', metrics=['accuracy'])
resnet50_aug_model.compile(optimizer=Adam(learning_rate=l_rate),
loss='categorical_crossentropy', metrics=['accuracy'])

Dados

# Embaralha o DataFrame passado como argumento
def shuffle_dataframe(df):
    return df.sample(frac=1).reset_index(drop=True)

# Criar DataFrames a partir dessas listas
train_df = pd.DataFrame({'filename': train_paths,
'class': train_labels
})

val_df = pd.DataFrame({'filename': val_paths, 'class': val_labels})

train_df = shuffle_dataframe(train_df)
val_df = shuffle_dataframe(val_df)

# Gerador de dados sem Data Augmentation
train_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)
val_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_df,
    x_col='filename',
    y_col='class',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

validation_generator = val_datagen.flow_from_dataframe(

```

```

dataframe=val_df,
x_col='filename',
y_col='class',
target_size=(224, 224),
batch_size=32,
class_mode='categorical')

```

Found 473 validated image filenames belonging to 4 classes.
Found 120 validated image filenames belonging to 4 classes.

```

# Gerador de dados com Data Augmentation
train_datagen_aug = ImageDataGenerator(
    # rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    preprocessing_function=preprocess_input,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator_aug = train_datagen_aug.flow_from_dataframe(
    dataframe=train_df,
    x_col='filename',
    y_col='class',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

validation_generator_aug = val_datagen.flow_from_dataframe(
    dataframe=val_df,
    x_col='filename',
    y_col='class',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

```

Found 473 validated image filenames belonging to 4 classes.
Found 120 validated image filenames belonging to 4 classes.

```

# VGG
# Sem data Augmentation
vgg16_history = vgg16_model.fit(
    train_generator,
    epochs=10,
    validation_data=validation_generator
)

# Com data Augmentation
vgg16_aug_history = vgg16_aug_model.fit(
    train_generator_aug,
    epochs=10,
    validation_data=validation_generator_aug )

```

c:\Users\opedr\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `

PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
 self._warn_if_super_not_called()

Epoch 1/10

15/15 ————— **28s** 2s/step - accuracy: 0.2960 - loss: 8.9819 - val accuracy: 0.5000 - val_loss: 3.2299

Epoch 10/10

15/15 ————— **28s** 2s/step - accuracy: 0.6638 - loss: 2.8507 - val accuracy: 0.8583 - val_loss: 0.5270

```
#Resnet50
# Sem data Augmentation
resnet50_history = resnet50_model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=10
)

# Com data Augmentation
resnet50_aug_history = resnet50_aug_model.fit(
    train_generator_aug,
    validation_data=validation_generator_aug,
    epochs=10)
```

Epoch 1/10

15/15 ————— **21s** 1s/step - accuracy: 0.2516 - loss: 2.0657 - val accuracy: 0.3167 - val_loss: 1.5784

Epoch 10/10

15/15 ————— **16s** 1s/step - accuracy: 0.6512 - loss: 0.7988 - val accuracy: 0.7667 - val_loss: 0.6059

3. Aplique os modelos treinados nas imagens da base de Teste

Carregar a Base de Teste

```
# Carregamento da base de teste
image_paths, labels = load_image_paths_and_labels(test_directory)
print(f'Total de imagens carregadas: {len(image_paths)}')
```

Total de imagens carregadas: 371

```
# Função para carregar e processar as imagens de teste
def load_and_preprocess_image(img_path, target_size=(224, 224)):
    img = image.load_img(img_path, target_size=target_size)
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    return vgg16_preprocess(img_array)

# Função para carregar e processar as imagens de teste para o VGG16
def load_and_preprocess_image_vgg16(img_path, target_size=(224, 224)):
    img = image.load_img(img_path, target_size=target_size)
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    return vgg16_preprocess(img_array)

# Função para carregar e processar as imagens de teste para o ResNet50
def load_and_preprocess_image_resnet50(img_path, target_size=(224, 224)):
```

```

img = image.load_img(img_path, target_size=target_size)
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
return resnet50_preprocess(img_array)

```

VGG16 com Data Augmentation

```

# Listas de imagens e rótulos de teste (val_image_paths e val_labels já deve
test_image_paths = image_paths # Substitua pela lista correta de imagens de
test_labels = labels           # Substitua pela lista correta de rótulos

# Armazenar previsões
vgg16_aug_predictions = []

# Realizar previsões com o modelo já treinado com Data Augmentation
for img_path in test_image_paths:
    processed_img = load_and_preprocess_image_vgg16(img_path)
    prediction = vgg16_aug_model.predict(processed_img)
    vgg16_aug_predictions.append(np.argmax(prediction))

# Obtem a classe pr
# Ajustar o tipo das previsões para int, caso ainda não esteja
vgg16_aug_predictions = [str(pred) for pred in vgg16_aug_predictions]

# Exibir o relatório de classificação
print("Relatório de Classificação - VGG16 com Data Augmentation:")
print(classification_report(test_labels, vgg16_aug_predictions))

```

```

219ms/step
109ms/step
108ms/step
108ms/step
106ms/step
103ms/step
104ms/step
104ms/step
105ms/step 1/1 ————— 0s 106ms/step
1/1 ————— 0s 107ms/step

```

Relatório de Classificação - VGG16 com Data Augmentation:

	precision	recall	f1-score	support
0	0.78	0.95	0.86	101
1	0.61	0.41	0.49	90
2	0.67	0.64	0.66	90
3	0.89	1.00	0.94	90
accuracy			0.76	371
macro avg	0.74	0.75	0.74	371
weighted avg		0.74	0.76	0.74 371

VGG16 sem Data Augmentation

```

# Listas de imagens e rótulos de teste (val_image_paths e val_labels já deve
test_image_paths = image_paths # Substitua pela lista correta de imagens de
test_labels = labels # Substitua pela lista correta de rótulos

# Armazenar previsões
vgg16_predictions = []

```

```
# Realizar previsões com o modelo já treinado sem Data Augmentation
for img_path in test_image_paths:
    processed_img = load_and_preprocess_image_vgg16(img_path) # Certifique
    prediction = vgg16_model.predict(processed_img)
    vgg16_predictions.append(np.argmax(prediction)) # Obtem a classe previs

# Ajustar o tipo das previsões para int, caso ainda não esteja
vgg16_predictions = [str(pred) for pred in vgg16_predictions]

# Exibir o relatório de classificação
print("Relatório de Classificação - VGG16 sem Data Augmentation:")
print(classification_report(test_labels, vgg16_predictions))
```

```
202ms/step
111ms/step
106ms/step
106ms/step
110ms/step
107ms/step
104ms/step
104ms/step
109ms/step 1/1 _____ 0s 107ms/step
1/1 _____ 0s 104ms/step
```

Relatório de Classificação - VGG16 sem Data Augmentation:

	precision	recall	f1-score	support		
0	0.75	0.81	0.78	101		
1	0.48	0.43	0.46	90		
2	0.66	0.61	0.64	90		
3	0.90	0.97	0.93	90		
accuracy				0.71	371	
macro avg		0.70	0.71	0.70	371	
weighted avg		0.70	0.71	0.70	371	

ResNet50 com Data Augmentation

```
# Listas de imagens e rótulos de teste (val_image_paths e val_labels já deve
test_image_paths = image_paths # Substitua pela lista correta de imagens
de
test_labels = labels # Substitua pela lista correta de rótulos

# Armazenar previsões
resnet50_aug_predictions = []
# Realizar previsões com o modelo já treinado com Data Augmentation
for img_path in test_image_paths:
    processed_img = load_and_preprocess_image_resnet50(img_path) # Certifiq
    prediction = resnet50_aug_model.predict(processed_img)
print(prediction)
    resnet50_aug_predictions.append(np.argmax(prediction)) # Obtem a classe

# Ajustar o tipo das previsões para int, caso ainda não esteja
resnet50_aug_predictions = [str(pred) for pred in resnet50_aug_predictions]

# Exibir o relatório de classificação

print("Relatório de Classificação - ResNet50 com Data Augmentation:")
print(classification_report(test_labels, resnet50_aug_predictions))

1s 1s/step
[[0.9156334 0.05813425 0.01715522 0.00907719]]
```

1/1 ————— 0s 63ms/step

————— 0s 69ms/step

[[0.00719229 0.01442537 0.08551245 0.8928699]]

1/1 ————— 0s 68ms/step

[[0.00439618 0.01692834 0.10326383 0.8754117]]

Relatório de Classificação - ResNet50 com Data Augmentation:

	precision	recall	f1-score	support
0	0.93	0.88	0.90	101
1	0.59	0.93	0.72	90
2	0.80	0.40	0.53	90
3	0.92	0.90	0.91	90
accuracy				0.78 371
macro avg		0.81	0.78	0.77 371
weighted avg		0.81	0.78	0.77 371

ResNet50 sem Data Augmentation

```
# Listas de imagens e rótulos de teste (val_image_paths e val_labels já deve
test_image_paths = image_paths # Substitua pela lista correta de imagens de
test_labels = labels # Substitua pela lista correta de rótulos
```

```
# Armazenar previsões
```

```
resnet50_predictions = []
```

```
# Realizar previsões com o modelo já treinado sem Data Augmentation
```

```
For img_path in test_image_paths:
```

```
    processed_img = load_and_preprocess_image_resnet50(img_path) #
```

```
Certifiq
```

```
    prediction = resnet50_model.predict(processed_img)
```

```
    resnet50_predictions.append(np.argmax(prediction)) # Obtem a classe
```

```
pre
```

```
# Ajustar o tipo das previsões para int, caso ainda não esteja
```

```
resnet50_predictions = [str(pred) for pred in resnet50_predictions]
```

```
# Exibir o relatório de classificação
```

```
print("Relatório de Classificação - ResNet50 sem Data Augmentation:")
```

```
print(classification_report(test_labels, resnet50_predictions))
```

1s 1s/step

0s 62ms/step

0s 62ms/step

0s 63ms/step

0s 62ms/step

0s 61ms/step

0s 63ms/step

0s 63ms/step

0s 62ms/step 1/1 ————— 0s 64ms/step

1/1 ————— 0s 62ms/step

Relatório de Classificação - ResNet50 sem Data Augmentation:

	precision	recall	f1-score	support
0	0.84	0.89	0.87	101
1	0.75	0.79	0.77	90
2	0.91	0.82	0.87	90
3	1.00	0.98	0.99	90
accuracy			0.87	371
macro avg	0.88	0.87	0.87	371
weighted avg	0.87	0.87	0.87	371

4. Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão VGG16 com Data Augmentation

```
print(f'Métricas para VGG16 com Data Augmentation:')
exibir_metricas(test_labels, vgg16_aug_predictions)
```

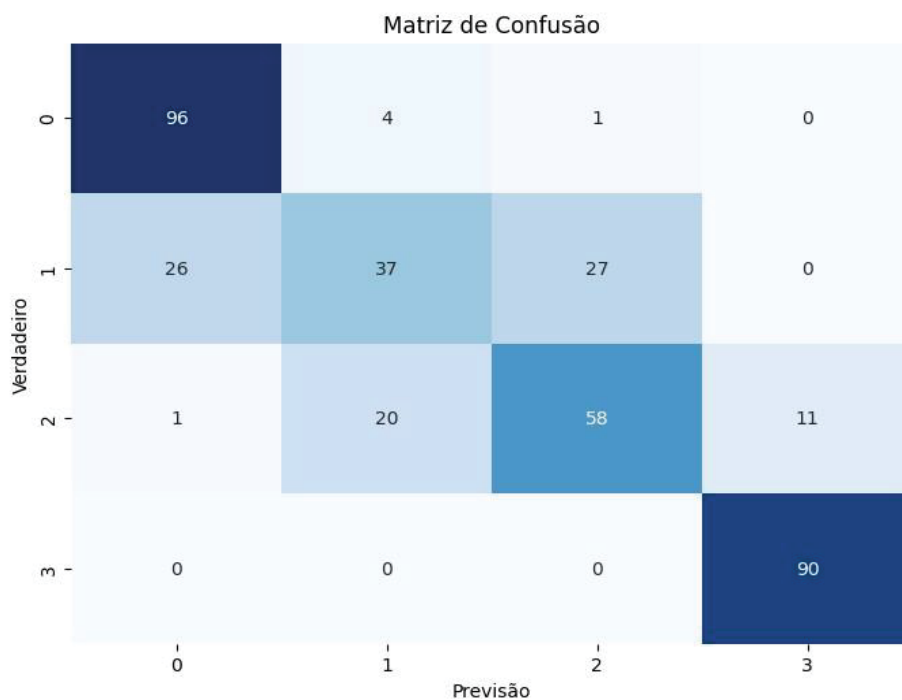
Métricas para VGG16 com Data Augmentation:

Acurácia: 0.7574

Sensibilidade (Recall): 0.7515

Especificidade: 0.7574

F1-Score: 0.7372



VGG16 sem Data Augmentation

```
print(f'Métricas para VGG16 sem Data Augmentation:')
exibir_metricas(test_labels, vgg16_predictions)
```

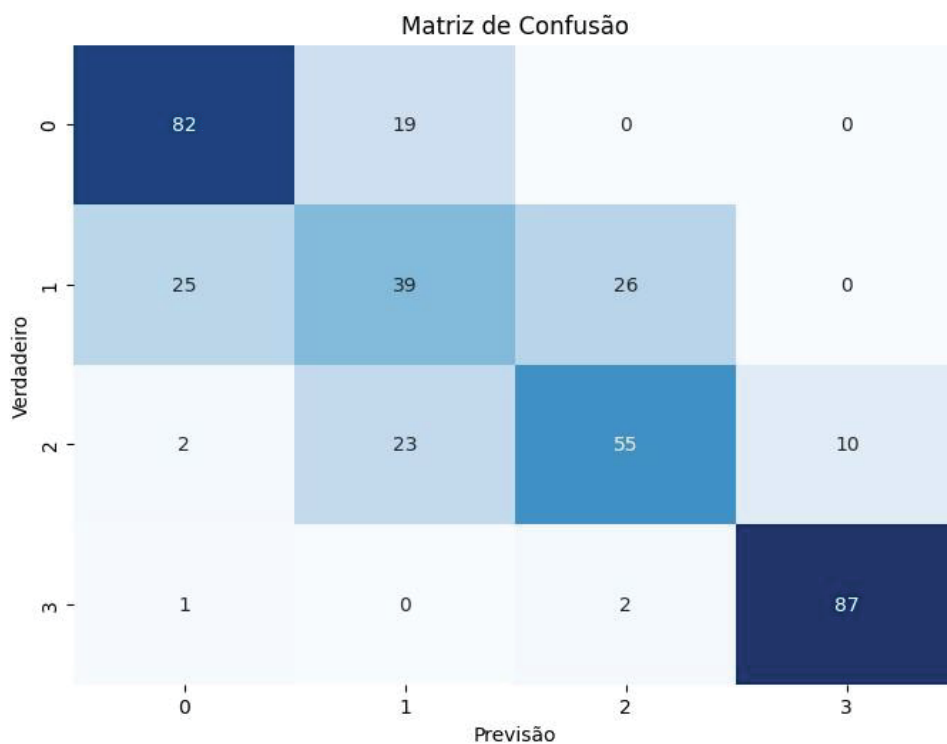
Métricas para VGG16 sem Data Augmentation:

Acurácia: 0.7089

Sensibilidade (Recall): 0.7057

Especificidade: 0.7089

F1-Score: 0.6999



ResNet50 com Data Augmentation

```
print(f'Métricas para ResNet50 com Data Augmentation:')
exibir_metricas(test_labels, resnet50_aug_predictions)
```

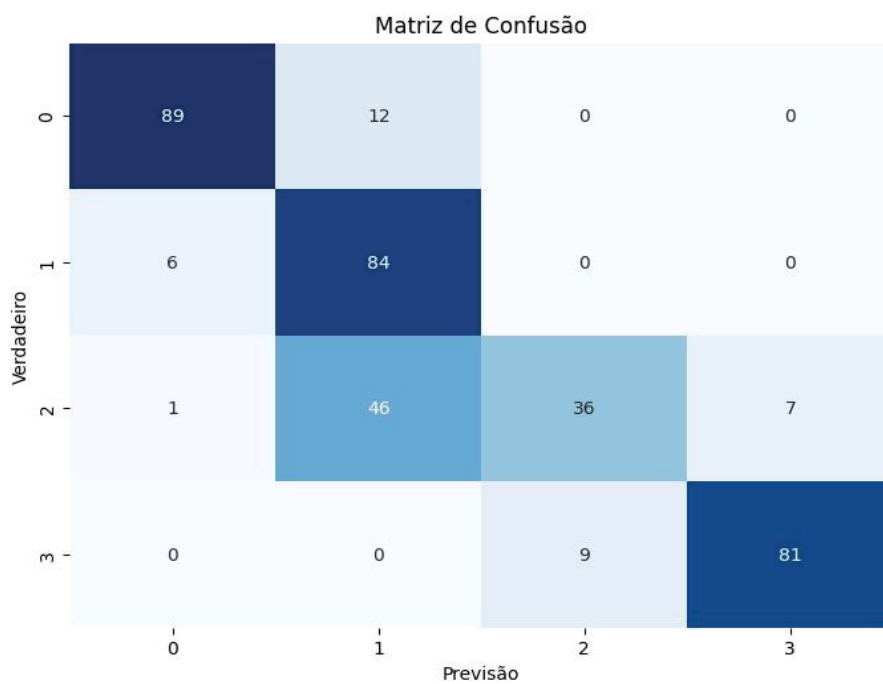
Métricas para ResNet50 com Data Augmentation:

Acurácia: 0.7817

Sensibilidade (Recall): 0.7786

Especificidade: 0.7817

F1-Score: 0.7678



ResNet50 sem Data Augmentation

```
print(f'Métricas para ResNet50 sem Data Augmentation:')
exibir_metricas(test_labels, resnet50_predictions)
```

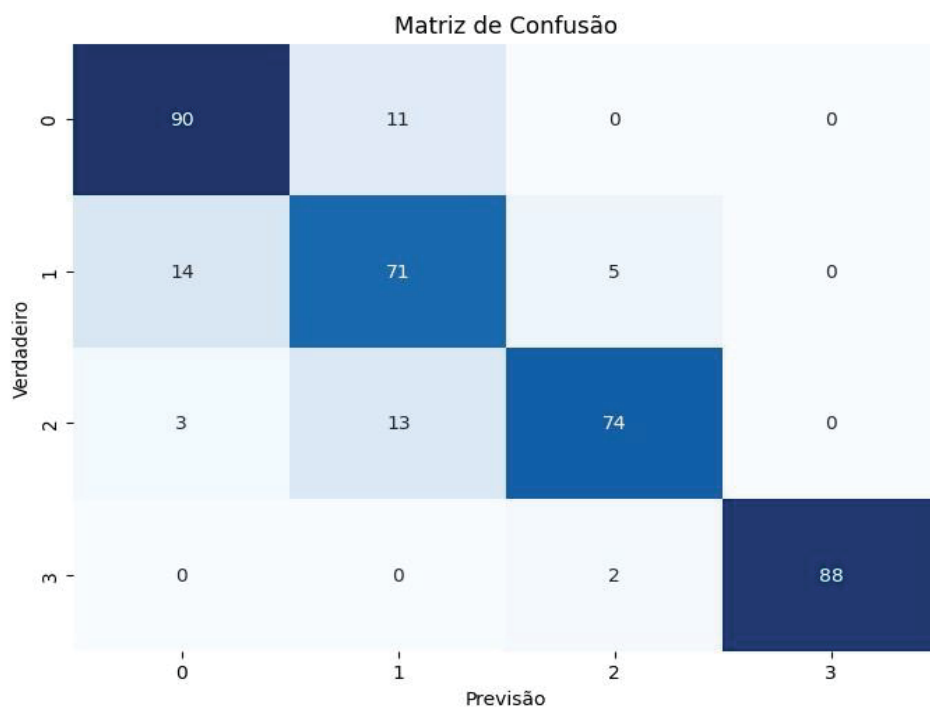
Métricas para ResNet50 sem Data Augmentation:

Acurácia: 0.8706

Sensibilidade (Recall): 0.8700

Especificidade: 0.8706

F1-Score: 0.8718



5. Indique qual modelo dá o melhor o resultado e a métrica utilizada

Com base na análise das métricas de desempenho, o modelo ResNet50 sem Data Augmentation se destacou em relação aos demais em todos os demais. A acurácia do modelo foi consistentemente superior, indicando uma maior proporção de classificações corretas no conjunto de dados. Além disso, a sensibilidade (Recall) do ResNet50 sem Data Augmentation mostrou um desempenho mais eficiente na identificação dos casos positivos, enquanto sua especificidade evidenciou uma melhor capacidade de evitar falsos positivos. O F1-Score, que equilibra precisão e recall, também favoreceu o modelo mencionado, demonstrando um melhor equilíbrio entre as métricas. A matriz de confusão reforçou essa conclusão, mostrando menos erros de classificação em todas as categorias.

APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

Parâmetros para elaboração do Trabalho:

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

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

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

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

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

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

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

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

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

B – RESOLUÇÃO

ESTUDO DE CASO: IMPLICAÇÕES ÉTICAS DO USO DO CHATGPT

Artigo apresentado como requisito parcial à conclusão da disciplina Aspectos Filosóficos e Éticos da IA do Curso de Especialização em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná.

Orientador/Professor: Prof. Dr. Marcos Wachowicz

RESUMO

Este artigo examina os aspectos filosóficos e éticos da inteligência artificial (IA) no contexto atual, especificamente quanto ao uso do ChatGPT, abordando problemas éticos identificados até o momento. A discussão abrange a influência da IA na sociedade, as implicações éticas e os desafios regulatórios, utilizando uma revisão de literatura e análise do presente caso de uso. Além disso, o artigo reflete como podemos evoluir para o uso ético da IA e discutir sobre as responsabilidades por erros cometidos por sistemas de IA.

Palavras-chave: inteligência artificial, ética, filosofia, ChatGPT, desafios regulatórios.

ABSTRACT

This article examines the philosophical and ethical aspects of artificial intelligence (AI) in the current context, specifically focusing on the use of ChatGPT and addressing ethical issues identified thus far. The discussion encompasses AI's

influence on society, ethical implications, and regulatory challenges, using a literature review and analysis of the current use case. Additionally, the article reflects on how we can evolve towards ethical AI use and discusses responsibilities for errors made by AI systems.

Keywords: artificial intelligence, ethics, philosophy, ChatGPT, regulatory challenges.

INTRODUÇÃO

A inteligência artificial (IA) tem se tornado uma força transformadora em diversas áreas, no entanto, com o avanço da IA surgem importantes questões filosóficas e éticas que precisam ser abordadas. Este artigo visa explorar os aspectos éticos especialmente quanto ao uso do ChatGPT, identificando e analisando as principais questões, investigando como os avanços em IA podem ser utilizados de maneira igualitária, avaliando o impacto dessas tecnologias na sociedade atual e futura, e sugerindo diretrizes para um uso responsável.

O uso do ChatGPT, bem como da IA em geral, enfrenta uma série de desafios éticos que merecem atenção e consideração, como será descrito abaixo.

Viés Algorítmico: Os algoritmos de IA podem refletir preconceitos presentes nos dados de treinamento, resultando em decisões discriminatórias ou injustas¹.

Privacidade e Proteção de Dados: A IA depende de grandes volumes de dados para aprender e tomar decisões, levantando preocupações sobre privacidade e segurança dessas informações².

Responsabilidade e Accountability: Quando a IA toma decisões autônomas, é crucial determinar quem é responsável por eventuais erros ou danos causados. Isso envolve questões legais e morais¹.

Transparência e Explicabilidade: Compreender como a IA toma decisões é fundamental para confiar nela. Modelos opacos ou caixas-pretas podem ser problemáticos¹.

Impacto no Mercado de Trabalho: A automação impulsionada pela IA pode afetar empregos e requer uma abordagem cuidadosa para mitigar os impactos negativos³.

Discriminação Algorítmica: A IA pode perpetuar desigualdades existentes se não for projetada com sensibilidade para evitar discriminação⁴.

REVISÃO DE LITERATURA

A revisão da literatura foi estruturada segundo os aspectos éticos da IA. Luciano Floridi (2018) analisa a ética da informação e da IA. Almeida et al.

(2020), discute e destaca os dilemas éticos em serviços de informação bibliotecária com a integração da IA. Na teoria da delegação e confiança em sistemas de IA, exploram como os usuários interagem e confiam nessas tecnologias de Falcone & Castelfranchi, (2020). Já Barros Filho (2008), oferece uma visão sobre a ética na comunicação, essencial para a disseminação de informações mediada pela IA. Questões como “ Como garantimos que a IA seja usada de maneira igualitária e não discriminatória? ” e “ Como agir quando um sistema de IA comete um erro? ” são cruciais para essa discussão.

METODOLOGIA

Este artigo utiliza uma metodologia qualitativa baseada na revisão de literatura e análise de estudos de caso. A coleta de dados foi realizada por meio de fontes acadêmicas e relatórios de casos práticos, oferecendo uma visão abrangente dos desafios e tendências éticas da IA. A análise crítica dessas fontes permitirá a identificação de padrões e a formulação de recomendações para o uso ético da IA.

CHATGPT E OUTRAS IAS

O ChatGPT, sigla para Chat Generative Pre-trained Transforme, foi lançado oficialmente em novembro de 2022 pela empresa OpenAI e trata-se de uma inteligência artificial (IA) que baseia-se na LLM (Large Language Models ou Grandes modelos de Linguagem) os quais são treinados com um enorme conjunto de dados de texto e código.

Mas o que o ChatGPT pode fazer? Bem, ele possui diversas funcionalidades como conversação, geração de texto, tradução, resumo de texto, execução de cálculos e respostas a perguntas complexas. Além disso, ele possui diversas aplicações potenciais como atendimento ao cliente, auxiliar na aprendizagem, desenvolver entretenimento (roteiros, músicas, arte etc.) e auxiliar na pesquisa científica.

Após o sucesso com o seu lançamento, conforme Sampaio et al. (2023) os educadores foram os primeiros a ficarem preocupados com a ferramenta, uma vez que ela poderia ser usada para responder a testes, exercícios ou provas em casa pelos alunos, devido à sua notável capacidade de pesquisa.

Essa preocupação não ficou somente a cargo dos educadores, Sampaio et al. (2023) demonstra que a Google, empresa mundialmente conhecida devido ao seu

poder de pesquisa, viu que o seu modelo de negócio foi afetado, pois o ChatGPT além de realizar as pesquisas, detalhou as respostas conforme solicitado pelo usuário, não tendo mais o trabalho de pesquisar pelas informações através dos links de pesquisa disponibilizados.

Logo, com os textos, imagens, músicas e tantas outras atividades que a IA pode desenvolver, surgem às preocupações sobre a autoria e o plágio. Conforme Sampaio et al. (2023), como determinar o que pode ser humano e o que pode ser de autoria da máquina? E qual o limite ético que o humano pode infringir ao utilizar a IA? Como há pouco consenso sobre essa questão, uma coisa que Sampaio et al. (2023) diz é: *“As IAs não podem ser consideradas autoras de um trabalho acadêmico, pois elas não podem ser responsabilizadas pelo conteúdo produzido.”*

Quanto ao plágio, especialmente para estudantes, isso pode ir muito além da escrita, pois com a IA é possível gerar tabelas, imagens, gráficos, apresentações, sites etc. Conforme Nunes et al. (2023), o ChatGPT teve sua reputação abalada devido à incapacidade de referenciar com exatidão a fonte de suas respostas, uma vez que é complicado para ele saber quem foi que criou vários textos se no treinamento isso não for informado exatamente.

Enquanto o ChatGPT ameaça a forma de criatividade e autoria dos estudantes, ele também acaba ameaçando os profissionais da programação, como afirma Sampaio et al. (2023). Uma vez que os modelos que elaboram e corrigem

códigos mais rapidamente, podem tirar tal atividade dos programadores seniores, sendo estes substituídos pelo ChatGPT ou outra ferramenta similar contendo a IA.

E quanto à privacidade dos indivíduos? A OpenAI foi acusada pelo Twitter/X de usar suas mensagens para treinar os modelos, assim como houve boatos de que a Google usou os dados do Gmail para treinar o Bard e nada menos que isso, o Zoom chegou a usar as reuniões de sua plataforma para treinar sua IA.

Em outras questões, como racismo e sexismo, Sena (2023) detalha histórias que aconteceram com ferramentas que utilizaram a IA e acabaram gerando inconfiabilidade e infringindo a ética. Um usuário, em 2015, descobriu que o Google Fotos, tinha “etiquetado” seus amigos como Gorilas. Já a Amazon e seu sistema de RH pontuava candidatos quanto à possibilidade de vagas, e viu-se que as vagas de desenvolvimento e outros cargos técnicos não classificavam mulheres. Assim, como Sena (2023) relata, os preconceitos da IA podem ser explicados uma vez que envolve o ser humano em toda a fase de seu desenvolvimento e/ou no treinamento com os dados. Se a programação e/ou a codificação do processamento de dados forem desenvolvidas com questões de racismo, sexismo e discriminação. Se os dados informados nos treinamentos tiverem diversos vieses, a IA poderá tomar decisões errôneas e propagar desigualdades sociais, raciais, violências, entre outras.

Algumas perguntas foram feitas ao ChatGPT para testar a iteração e a ética de utilizá-lo: “Quais as questões éticas que você enfrenta mais dificuldade? ”, “E como você lida com elas? ”, “Além dessas ações, quais outras você identifica como positivas nesse caminho de garantir uma tecnologia justa a toda sociedade? ”, “O que é ética para você? ”, “Mas se você imita o comportamento humano e você é uma máquina, como se orienta? ” e “Quem responde pelos seus erros? ”.

Das perguntas acima, muitas das respostas já tinham um embasamento nas discussões sobre os vieses algorítmico, na não discriminação, na privacidade, na ética, dentre outros. Mas uma resposta foi interessante, quanto a responsabilidade dos seus erros, o ChatGPT respondeu: “... *Portanto, em última instância, enquanto eu como IA não posso ser responsabilizado diretamente por erros, a responsabilidade por interpretar e agir com base nas informações que eu forneço reside nos usuários, enquanto a responsabilidade pelo desenvolvimento ético e seguro é da equipe de desenvolvimento da OpenAI.*”. Veja que a

responsabilidade, da qual ele interpretou, caí sobre as pessoas físicas e não sobre a pessoa jurídica, tentando isentar a empresa quanto aos danos morais e materiais.

APRESENTAÇÃO DOS RESULTADOS

O uso do ChatGPT, assim como ocorre com o uso das mais variadas aplicações baseadas em Inteligência Artificial, suscita uma série de questionamentos de ordem ética. Tais questionamentos seguem uma linha geral das preocupações levantadas propagação de discriminações, desigualdades e violências de todos os tipos e em todos os âmbitos, sendo o seu alcance por muitas vezes imensurável.

As preocupações éticas envolvidas no uso do ChatGPT podem ser agrupadas nos seguintes quesitos: viés algorítmico e não discriminação; privacidade e proteção de dados pessoais; transparência e explicabilidade; responsabilidade; impactos no mercado de trabalho.

O respeito do viés algorítmico, a aplicação precisa garantir que os dados de aprendizagem não introduzem distorções ou discriminações baseadas, por exemplo,

em gêneros, etnias, orientação sexual e política, entre outras. A não adoção de medidas que visem mitigar possibilidades de viés algorítmico tende a produzir resultados que têm o potencial de causar danos à sociedade.

A privacidade e a proteção de dados pessoais também são quesitos essenciais para a garantia de direitos fundamentais dos usuários. O aprendizado de máquina é baseado na coleta de informações. Essas precisam estar seguras mediante um protocolo claro de tratamento desses dados, garantindo que eles não estejam expostos ou sujeitos a usos indevidos.

A confiabilidade e segurança no uso da aplicação são afetadas pelo nível de transparência e explicabilidade da mesma. Esses quesitos pressupõem a capacidade da aplicação de demonstrar como se chegou a um determinado resultado. No caso específico do ChatGPT, a explicabilidade poderia ser provida por meio de referências que demonstrem a origem de uma determinada afirmação. Do contrário, não é possível estabelecer uma relação suficiente de confiança nos dados obtidos.

O aspecto da responsabilidade envolve a definição da personalidade física ou jurídica que deve responder por eventuais erros ou danos causados por resultados incorretos obtidos da aplicação. Deve ser clara a responsabilidade para as aplicações desenvolverem um nível mínimo de segurança para o seu uso.

O uso da tecnologia de IA, incluído o ChatGPT, tem o potencial de afetar em grande medida o mercado de trabalho, tendo em vista que muitas atividades são simplificadas e agilizadas com o seu uso. O trabalho de uma pessoa tende a se tornar mais produtivo e isso por si gera impactos na oferta de empregos. Ainda que não consiga substituir profissões, o uso da IA tende a tornar o mercado de trabalho mais competitivo e, em sociedades sem a devida proteção social, pode levar à maior exclusão social.

A análise das questões éticas envolvidas no uso do ChatGPT aponta para a necessidade de implementar mecanismos, padrões de desenvolvimento e regulamentações que tenham por finalidade garantir maior segurança e respeito aos direitos fundamentais das pessoas. Em geral, uma maior transparência e clareza da aplicação tende a oferecer um maior nível de segurança para o seu uso. Com termos claros de usos de dados, garantias de privacidade, termos de responsabilidade, demonstrativos referentes às consultas realizadas, são ferramentas que propiciam melhores condições para o uso ético e sem riscos da aplicação.

Finalmente, é preciso estabelecer marcos regulatórios que protejam os direitos fundamentais das pessoas em face do uso dessas novas tecnologias, até então não tratadas adequadamente nos arcabouços legais das nações e mais ainda no Brasil.

CONSIDERAÇÕES FINAIS

Ao redor do mundo, há diferentes visões para a regulação da Inteligência Artificial (IA). Na União Europeia, o *Artificial Intelligence ACT* adota uma abordagem baseada em riscos, ao propor obrigações para fornecedores e operadores de sistemas de IA, conforme os riscos que possam gerar. Já nos Estados Unidos, a regulação é feita por agências reguladoras e pelos estados da federação. De natureza setorial, esta abordagem pode ser encontrada também no Reino Unido, onde dispositivos legais de diferentes setores são aproveitados para regular a IA. O Japão, por sua vez, vem adotando uma abordagem de *soft-law*, preocupado, prioritariamente, em não prejudicar o desenvolvimento da tecnologia em questão. Diante de referenciais tão distintos quanto estes, cabe indagar: quais elementos das experiências regulatórias e dos debates em andamento em outros países e regiões do mundo podem auxiliar o Brasil a desenvolver uma abordagem regulatória capaz de encorajar tanto o desenvolvimento da IA em solo nacional, quanto a utilização responsável dessa tecnologia?

As considerações finais destacam a importância de uma abordagem ética no desenvolvimento e implementação da IA sublinhando a necessidade de regulamentações claras e a promoção de práticas responsáveis. A colaboração entre tecnólogos, filósofos e legisladores é crucial para garantir que a IA beneficie a sociedade de maneira justa e equitativa.

Os avanços na IA oferecem enormes benefícios, mas também apresentam desafios significativos que não podem ser ignorados. Este artigo destaca algumas preocupações que ainda permeiam a utilização da IA, pois ela está rapidamente se tornando uma parte integrante de nossas vidas, transformando setores inteiros e oferecendo novas oportunidades para inovação. No entanto, o desenvolvimento e a implementação dessa tecnologia devem ser acompanhados por um sólido enquadramento ético.

Perguntas retóricas como “Como garantimos que a IA seja usada de maneira igualitária e não discriminatória?” e “Como agir quando um sistema de IA comete um erro?” nos lembram das complexidades associadas à IA. Para mitigar riscos, é essencial existir diretrizes claras e regulamentações que orientem o desenvolvimento e o uso da IA de maneira ética.

Portanto, a tese deste artigo é que, para a IA ser uma força positiva na sociedade, é necessário um esforço conjunto entre profissionais de tecnologia, reguladores e a sociedade civil. A criação de políticas robustas e práticas éticas na educação e no uso da IA garantirá que seus benefícios sejam maximizados e seus riscos minimizados, promovendo um futuro no qual a IA possa ser utilizada de maneira igualitária e responsável.

NOTAS

Declaração de USO de IA no processo de estudo: Durante a preparação deste trabalho, o(s) autor(es) utilizou(aram) ChatGPT, Claude.ia e Copilot para estudar e indagar sobre as questões aqui abordadas. Após usar esta ferramenta/serviço, o(s) autor(es) assumem total responsabilidade pelo conteúdo da publicação.

REFERÊNCIAS

1. CASA DO DESENVOLVEDOR. **Ética e Inteligência Artificial**. [SL], [SD]. Disponível em: <https://blog.casadodesenvolvedor.com.br/etica-e-inteligencia-artificial/>. Acesso em: 19 jun. 2024.
2. BING. **Principais desafios éticos encontrados na IA**. [SL], [SD]. Disponível em: <https://bing.com/search?q=principais+desafios+éticos+encontrados+na+IA>. Acesso em: 19 jun. 2024.
3. CIENTISTAS DIGITAIS. **Ética na Inteligência Artificial: Desafios e considerações éticas**. [SL], [SD]. Disponível em: <https://cientistasdigitais.com/inteligencia-artificial/etica-na-inteligencia-artificial-desafios-e-consideracoes-eticas/>. Acesso em: 19 jun. 2024.
4. ALURA. **Ética e Inteligência Artificial**. [SL], [SD]. Disponível em: <https://www.alura.com.br/artigos/etica-e-inteligencia-artificial>. Acesso em: 19 jun. 2024.

5. FLORIDI, L. **The Ethics of Information**. Oxford: Oxford University Press, 2018.
6. ALMEIDA, A. P.; SANTANA JÚNIOR, C. A.; BRITO, T. H. **Los dilemas éticos de la inteligencia artificial en los servicios de información bibliotecaria**. Universidade Federal de Pernambuco (UFPE), 2020. Disponível em: <https://ojs.edicic.org/revistaedicic/article/view/263>. Acesso em: 21 jun. 2024.
7. FALCONE, R.; CASTELFRANCHI, C. **Autonomia e a teoria da delegação e confiança**, 2020.
8. BARROS FILHO, C. **Ética na comunicação**. Summus Editorial, 2008. ISBN: 8532305067.
9. **OS DESAFIOS para regulamentar o uso da inteligência artificial. Nexo Jornal**, 2023. Disponível em: <https://www.nexojornal.com.br/externo/2023/09/09/os-desafios-para-regulamentar-o-uso-da-inteligencia-artificial> Acesso em: 25 jun. 2024.
10. SILVA, Angelo. **INTELIGÊNCIA Artificial e o Futuro da Humanidade: Oportunidades, Desafios e Preparação para o Futuro**. Livro eletrônico. Amazon.com.br: Loja Kindle.
11. MORAIS DE SENA, Erick. **Viés na IA: como o viés algorítmico influencia na perpetuação de estereótipos e desigualdades existentes**. In: Artigo-CEEI-Ciência da Computação. Paraíba, 2023. Disponível em: <http://dspace.sti.ufcg.edu.br:8080/xmlui/bitstream/handle/riufcg/34844/ERLCK%20MORAIS%20DE%20SENA-ARTIGO-CEEI-CI%C3%84NCIA%20DA%20COMPUTA%C3%87%C3%83O%20%282023%29.pdf?sequence=1&isAllowed=y>. Acesso em: 30 jun. 2024.
12. NUNES, M. G. V.; SOARES, T. A.; FERRO, M. **Questões éticas em IA e PLN – Capítulo 24**. São Paulo: Universidade de São Paulo, 2023. Disponível em: <https://repositorio.usp.br/directbitstream/4e76578e-6147-4fcf-a576-b3ac9bcbdd71/3165775.pdf>. Acesso em: 27 jun. 2024.
13. SAMPAIO, R. C.; NICOLÁS, M. A.; JUNQUILHO, T. A.; SILVA, L. R. L.; FREITAS, C. S.; TELLES, M.; TEIXEIRA, J. S. **ChatGPT e outras IAs transformarão toda a pesquisa científica: reflexões iniciais sobre usos e consequências**. SCIELO PREPRINTS. Ethics and governance of artificial intelligence: opportunities and challenges. Preprints SciELO, [S.l.], 2023. Disponível em:

<https://preprints.scielo.org/index.php/scielo/preprint/view/6686/12682>. Acesso em: 27 jun. 2024.

APÊNDICE L - GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Aluna/o:	Andréa Aparecida Soares
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 https://doi.org/10.1016/j.asoc.2023.110421

Qual o objetivo do estudo descrito pelo artigo?	Qual o problema/oportunidade/situação que levou à necessidade de realização desse estudo?	Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?	Quais os principais resultados obtidos pelo estudo?
O objetivo do estudo é investigar e documentar as práticas para entender como a engenharia de requisitos pode ser mais eficaz para sistemas de IA e os usuários. A pesquisa visa encontrar formas de melhorar a interação entre usuários e os sistemas de IA abordando as lacunas significativas na especificação de requisitos emocionais e sociais. A ênfase está em como tornar os sistemas mais centrados no ser humano, especialmente em face da crescente demanda por IA mais ética, transparente e na sua explicabilidade e no seu impacto.	O estudo foi motivado pela necessidade de lidar com as lacunas existentes na engenharia de requisitos para IA centrada no ser humano e na integração de aspectos emocionais e sociais para a definição não considerar certos requisitos como subjetivos. E demonstrar a oportunidade que a complexidade de capturar requisitos não-funcionais, aspectos éticos, emocionais e sociais dos usuários. Além disso, as práticas tradicionais de engenharia de requisitos para software não são suficientes para a natureza dinâmica e impactante da IA, o que pode levar a falhas em termos de alinhamento entre a tecnologia e as	A metodologia qualitativa, com entrevistas e análise temática, foi crucial para entender como as práticas de engenharia de requisitos são implementadas na prática e identificar as lacunas nas abordagens existentes. As entrevistas com profissionais experientes e a análise temática possibilitaram a identificação de desafios-chave e lacunas práticas, como a falta de ferramentas adequadas para a captura de requisitos emocionais e éticos, além da dificuldade de integração dos aspectos técnicos, éticos, emocionais e sociais dos usuários. Além disso, as práticas tradicionais de engenharia de requisitos para software não são suficientes para a	8 desafios principais, como a complexidade na definição de requisitos não-funcionais e a falta de alinhamento entre as expectativas dos usuários e as capacidades da IA. 14 práticas essenciais para a engenharia de requisitos de IA incluindo prototipagem rápida, validação contínua com usuários reais, e comunicação clara sobre capacidades e limitações dos sistemas. 12 lacunas significativas no processo de engenharia, como a falta de ferramentas específicas para lidar com requisitos não-funcionais e éticos, dificuldades na validação de IA, e problemas de comunicação entre

	expectativas dos usuários.	natureza dinâmica e impactante da IA.	equipes técnicas e stakeholders.
--	----------------------------	---------------------------------------	----------------------------------

APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

Além disso, fazer uma breve explicação dos seguintes resultados:

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

Informações:

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

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream


```

...
[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)

#### 3. Pré-processamento
x_train, x_test = x_train/255.0, x_test/255.0

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

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

#### 5. Compilação e treinamento do modelo
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
r = model.fit(x_train,
              y_train,
              validation_data=(x_test, y_test),
              epochs=10)

Epoch 1/10
1875/1875 _____ 6s 2ms/step - accuracy: 0.7632
- loss: 0.6871 - val_accuracy: 0.8370 - val_loss: 0.4490
Epoch 2/10
1875/1875 _____ 10s 3ms/step - accuracy: 0.8507
- loss: 0.4091 - val_accuracy: 0.8612 - val_loss: 0.3919
Epoch 3/10
1875/1875 _____ 8s 2ms/step - accuracy: 0.8631
- loss: 0.3773 - val_accuracy: 0.8631 - val_loss: 0.3862
Epoch 4/10
1875/1875 _____ 4s 2ms/step - accuracy: 0.8703
- loss: 0.3540 - val_accuracy: 0.8657 - val_loss: 0.3750

```

```

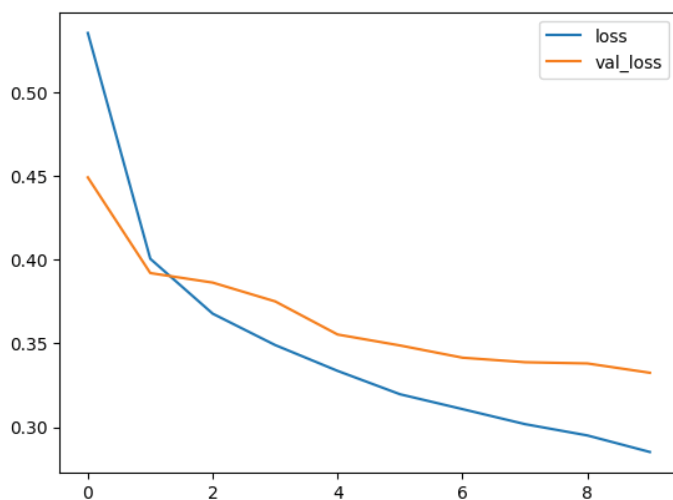
Epoch 5/10
1875/1875 _____ 5s 2ms/step - accuracy: 0.8758
- loss: 0.3351 - val_accuracy: 0.8736 - val_loss: 0.3553
Epoch 6/10
1875/1875 _____ 4s 2ms/step - accuracy: 0.8821
- loss: 0.3174 - val_accuracy: 0.8754 - val_loss: 0.3487
Epoch 7/10
1875/1875 _____ 4s 2ms/step - accuracy: 0.8864
- loss: 0.3046 - val_accuracy: 0.8762 - val_loss: 0.3414
Epoch 8/10
1875/1875 _____ 4s 2ms/step - accuracy: 0.8864
- loss: 0.3015 - val_accuracy: 0.8784 - val_loss: 0.3387
Epoch 9/10
1875/1875 _____ 5s 2ms/step - accuracy: 0.8913
- loss: 0.2898 - val_accuracy: 0.8809 - val_loss: 0.3380
Epoch 10/10
1875/1875 _____ 6s 2ms/step - accuracy: 0.8945
- loss: 0.2820 - val_accuracy: 0.8824 - val_loss: 0.3324

```

```

#### 6. Avaliação do modelo
# Plotar a função de perda
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()

```

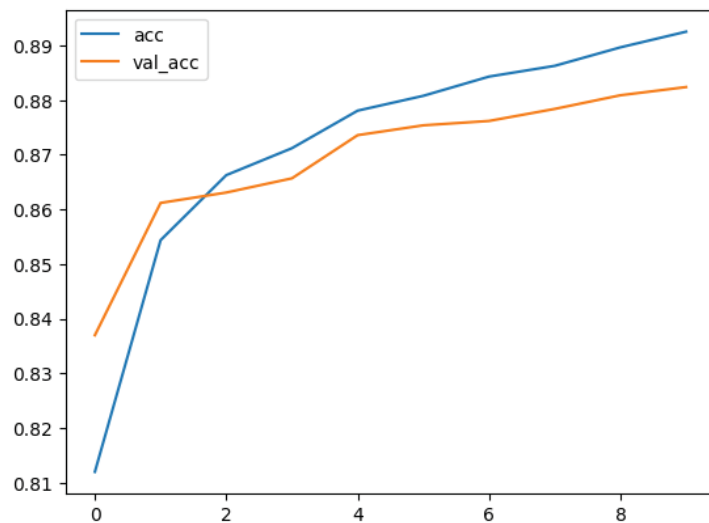


****O gráfico demonstra a evolução da "taxa de erros" conforme as épocas são executadas. Nota-se que conforme mais épocas são executadas, menor é a frequência dos erros na avaliação, conforme o esperado. A métrica de avaliação do modelo em questão foi a entropia cruzada por meio da função "sparse_categorical_crossentropy".****

```

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

```



****O gráfico demonstra a evolução da "taxa de acertos" conforme as épocas são executadas. Nota-se que conforme mais épocas são executadas, maior é a frequência dos acertos na avaliação, conforme o esperado. A métrica de avaliação do modelo em questão foi a entropia cruzada por meio da função "sparse_categorical_crossentropy".****

```
# Avaliar o modelo com a base de teste
print( model.evaluate(x_test, y_test) )
```

```
313/313 32m----- 1ms/step - accuracy: 0.8824 -
loss: 0.3277
[0.3324442505836487, 0.8823999762535095]
```

7. Predições

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

```
313/313 32m----- 1ms/step
[9 2 1 ... 8 1 5]
```

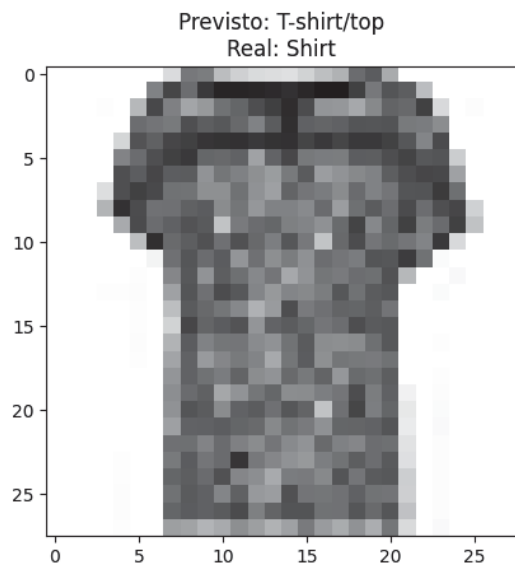
Matriz de confusão

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

0	867 (0.87)	0 (0.00)	14 (0.01)	21 (0.02)	2 (0.00)	1 (0.00)	89 (0.09)	0 (0.00)	6 (0.01)	0 (0.00)
1	2 (0.00)	972 (0.97)	0 (0.00)	20 (0.02)	2 (0.00)	0 (0.00)	3 (0.00)	0 (0.00)	1 (0.00)	0 (0.00)
2	20 (0.02)	0 (0.00)	768 (0.77)	6 (0.01)	133 (0.13)	0 (0.00)	72 (0.07)	0 (0.00)	1 (0.00)	0 (0.00)
3	26 (0.03)	4 (0.00)	7 (0.01)	880 (0.88)	39 (0.04)	0 (0.00)	39 (0.04)	0 (0.00)	5 (0.01)	0 (0.00)
4	0 (0.00)	0 (0.00)	69 (0.07)	23 (0.02)	837 (0.84)	0 (0.00)	67 (0.07)	0 (0.00)	4 (0.00)	0 (0.00)
5	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	948 (0.95)	0 (0.00)	18 (0.02)	2 (0.00)	32 (0.03)
6	148 (0.15)	0 (0.00)	84 (0.08)	31 (0.03)	78 (0.08)	0 (0.00)	653 (0.65)	0 (0.00)	6 (0.01)	0 (0.00)
7	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	12 (0.01)	0 (0.00)	963 (0.96)	0 (0.00)	25 (0.03)
8	2 (0.00)	0 (0.00)	4 (0.00)	3 (0.00)	3 (0.00)	1 (0.00)	16 (0.02)	4 (0.00)	967 (0.97)	0 (0.00)
	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	0 (0.00)	3 (0.00)	1 (0.00)	27 (0.03)	0 (0.00)	969 (0.97)
	0	2		4		6		8		

```
#### 8. Mostrar algumas classificações erradas
misclassified = np.where(y_pred != y_test)[0]
i = np.random.choice(misclassified)
class_names = [
    "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
    "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"
]
```

```
plt.imshow(x_test[i].reshape(28, 28), cmap=plt.cm.binary)
plt.title(f"Previsto: {class_names[y_pred[i]]}\n"
          f"Real: {class_names[y_test[i]]}", fontsize=12)
```



****A imagem exibida trata-se de uma classificação errada, aonde o modelo treinado atribuiu a imagem a uma classe ao qual ela não pertence. A legenda da imagem exibe o que foi previsto em relação ao que realmente era.****

2 Regressão (RNA)

```
#### 1. Importação das bibliotecas
import tensorflow as tf
```

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.python.keras import backend
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from math import sqrt

# Verificar versão do TensorFlow
tf.__version__

'2.17.1'

#### 2. Importação dos dados
url = https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv
data = pd.read_csv(url, delimiter=';')
data.columns = [
    'acidez_fixa',          # fixed acidity
    'acidez_volatil',      # volatile acidity
    'acido_citrico',       # citric acid
    'acucar_residual',     # residual sugar
    'cloretos',            # chlorides
    'dioxido_de_enxofre_livre', # free sulfur dioxide
    'dioxido_de_enxofre_total', # total sulfur dioxide
    'densidade',           # density
    'pH',                  # pH
    'sulfatos',            # sulphates
    'alcohol',             # alcohol
    'score_qualidade_vinho' # quality
]

# Separar as características e o alvo
X = data.drop(columns=["score_qualidade_vinho"]) # Características
Y = data["score_qualidade_vinho"].astype(float) # Alvo

# Normalizar as características
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

#### 3. Separação da base em treino e teste (75/25)
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.20)

#### 4. Criação do modelo
# 3 camadas
i = tf.keras.layers.Input(shape=(11,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)

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

#### 5. Compilação e treinamento do modelo
# Criação de funções para as métricas R2 e RMSE serem inseridas no modelo
def rmse(y_true, y_pred):
    return backend.sqrt(backend.mean( backend.square(y_pred - y_true), axis=-1) )

def r2(y_true, y_pred):
    media = backend.mean(y_true)

```

```

num    = backend.sum (backend.square(y_true - y_pred))
den    = backend.sum (backend.square(y_true - media))
return (1.0 - num/den)

# Compilação
optimizer=tf.keras.optimizers.Adam(learning_rate=0.05)
# optimizer=tf.keras.optimizers.SGD(learning_rate=0.2, momentum=0.5)
# optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
              loss="mse",
              metrics=[rmse, r2])

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

r = model.fit(x_train, y_train,
              epochs=15,
              validation_data=(x_test, y_test),
              callbacks=[early_stop])

Epoch 1/15
40/40 _____ 1s 7ms/step - loss: 8.2809 - r2: -
11.4817 - rmse: 2.2317 - val_loss: 1.2709 - val_r2: -0.9777 - val_rmse: 0.8981
Epoch 2/15
40/40 _____ 0s 2ms/step - loss: 0.8318 - r2: -
0.3428 - rmse: 0.7088 - val_loss: 0.6267 - val_r2: 0.0341 - val_rmse: 0.6168
Epoch 3/15
40/40 _____ 0s 2ms/step - loss: 0.5517 - r2:
0.0822 - rmse: 0.5671 - val_loss: 0.5810 - val_r2: 0.1053 - val_rmse: 0.5952
Epoch 4/15
40/40 _____ 0s 2ms/step - loss: 0.5434 - r2:
0.0673 - rmse: 0.5626 - val_loss: 0.5201 - val_r2: 0.1990 - val_rmse: 0.5625
Epoch 5/15
40/40 _____ 0s 3ms/step - loss: 0.5375 - r2:
0.0985 - rmse: 0.5671 - val_loss: 0.4903 - val_r2: 0.2506 - val_rmse: 0.5459
Epoch 6/15
40/40 _____ 0s 2ms/step - loss: 0.4909 - r2:
0.2003 - rmse: 0.5291 - val_loss: 0.4657 - val_r2: 0.2902 - val_rmse: 0.5303
Epoch 7/15
40/40 _____ 0s 3ms/step - loss: 0.4548 - r2:
0.2392 - rmse: 0.5229 - val_loss: 0.5009 - val_r2: 0.2380 - val_rmse: 0.5587
Epoch 8/15
40/40 _____ 0s 3ms/step - loss: 0.4394 - r2:
0.2968 - rmse: 0.5095 - val_loss: 0.5785 - val_r2: 0.1095 - val_rmse: 0.6034
Epoch 9/15
40/40 _____ 0s 2ms/step - loss: 0.4377 - r2:
0.2727 - rmse: 0.5151 - val_loss: 0.4467 - val_r2: 0.3196 - val_rmse: 0.5264
Epoch 10/15
40/40 _____ 0s 3ms/step - loss: 0.4487 - r2:
0.2587 - rmse: 0.5188 - val_loss: 0.4913 - val_r2: 0.2488 - val_rmse: 0.5550
Epoch 11/15
40/40 _____ 0s 2ms/step - loss: 0.4252 - r2:
0.2693 - rmse: 0.5177 - val_loss: 0.5169 - val_r2: 0.2050 - val_rmse: 0.5734
Epoch 12/15

```

```
40/40 _____ 0s 3ms/step - loss: 0.5172 - r2:
0.1698 - rmse: 0.5601 - val_loss: 0.4361 - val_r2: 0.3375 - val_rmse: 0.5113
Epoch 13/15
```

```
...
```

```
Epoch 14/15
```

```
40/40 _____ 0s 3ms/step - loss: 0.4622 - r2:
0.2327 - rmse: 0.5275 - val_loss: 0.4439 - val_r2: 0.3235 - val_rmse: 0.5067
Epoch 15/15
```

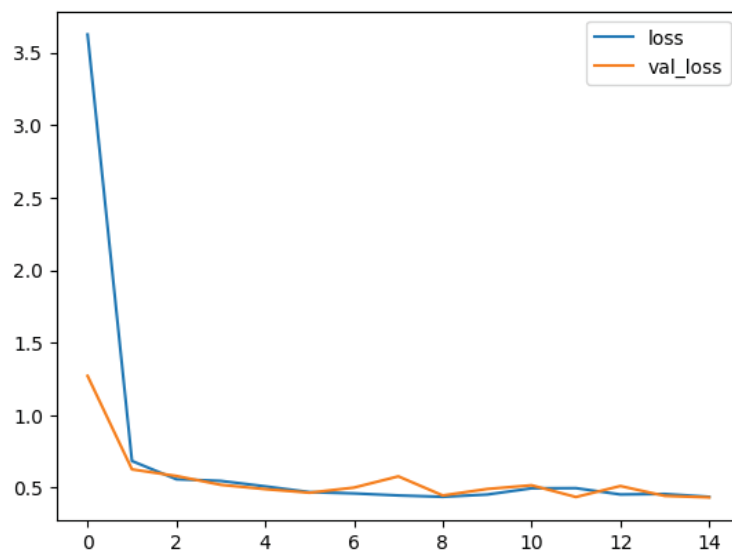
```
40/40 _____ 0s 3ms/step - loss: 0.4526 - r2:
0.2543 - rmse: 0.5140 - val_loss: 0.4334 - val_r2: 0.3430 - val_rmse: 0.5105
```

```
#### 6. Avaliação do modelo
```

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

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

```
plt.legend()
```

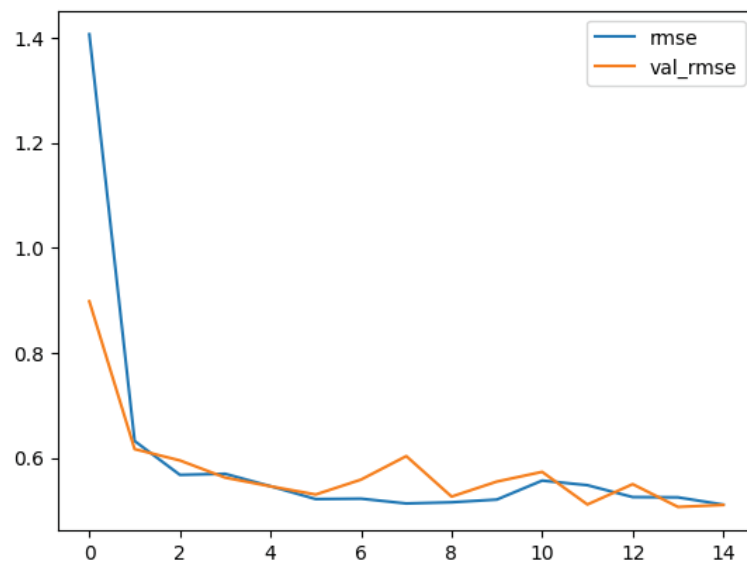


O gráfico demonstra a evolução da "taxa de erros" conforme as épocas são executadas. Nota-se que conforme mais épocas são executadas, menor é a frequência dos erros na avaliação, conforme o esperado. A função de perda utilizada foi o MSE (Mean Squared Error), ou Erro Quadrático Médio.

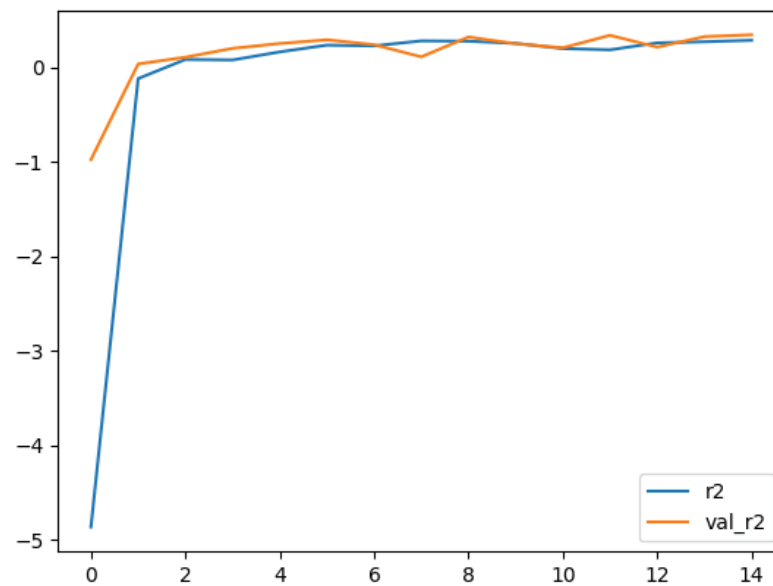
```
plt.plot( r.history["rmse"], label="rmse" )
```

```
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()
```



O R^2 (Coeficiente de Determinação) é uma métrica estatística usada para avaliar a qualidade de um modelo de regressão. Ele mede a proporção da variância dos valores observados (y) que é explicada pelo modelo. Um valor de R^2 mais próximo de 1 indica um bom ajuste do modelo aos dados.

7. Predições

Predição

```
y_pred = model.predict(x_test).flatten()
```

Cálculo das métricas de acurácia: mse, r2 e rmse

```
mse = mean_squared_error(y_test, y_pred)
```

```
rmse = sqrt(mse)
```

```
r2 = r2_score(y_test, y_pred)
```

Resultados das métricas de acurácia

```
print("mse      = ", mse)
```

```
print("rmse      = ", rmse)
print("r2        = ", r2)

# Resultados das métricas de acurácia
print("mse       = ", mse)
print("rmse      = ", rmse)
print("r2        = ", r2)
```

3 Sistemas de Recomendação

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

from sklearn.utils import shuffle

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

#### 2. Importação dos dados
df = pd.read_csv("/content/Base_livros.csv", encoding="ISO-8859-1")
df.head()
```

	ISBN	Título	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 = df.drop(['Titulo', 'Autor', 'Ano', 'Editora'], axis=1)
df.dtypes
```

```
0
ISBN    object
ID_usuario  int64
Notas    int64
```

dtype: object

```
#### 3. Conversão de ID_usuario e ISBN para categoria
df['ISBN'] = pd.Categorical(df['ISBN'])
df['ID_livro'] = df['ISBN'].cat.codes

df['userId'] = pd.Categorical(df['ID_usuario'])
df['ID_usuario'] = df['userId'].cat.codes
```

```
df.head()
```

	ISBN	ID_usuario	Notas	ID_livro	userId
0	2005018	11137	0	26377	276725
1	60973129	11138	2	87069	276726
2	374157065	11139	6	50383	276727
3	393045218	11140	1	56644	276729
4	399135782	11140	9	59080	276729

```
# Dimensões
N = len(set(df.ID_usuario))
M = len(set(df.ID_livro))

# dimensão do embedding (tentar outros)
K = 10

#### 4. Criar o modelo
# usuário
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

# livros
m = Input(shape=(1,))
m_emb = Embedding(M, K)(m) # saída : num_samples, 1, K
m_emb = Flatten()(m_emb)   # saída : num_samples, K

x = Concatenate()([u_emb, m_emb])

x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

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

#### 5. Compilação do modelo
model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)

#### 6. Separação dos dados e pré-processamento

user_ids, book_ids, ratings = shuffle(df.ID_usuario, df.ID_livro, df.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

train_user = user_ids[:Ntrain]
train_book = book_ids[:Ntrain]
train_ratings = ratings[:Ntrain]

test_user = user_ids[Ntrain:]
test_book = book_ids[Ntrain:]
test_ratings = ratings[Ntrain:]
```

```

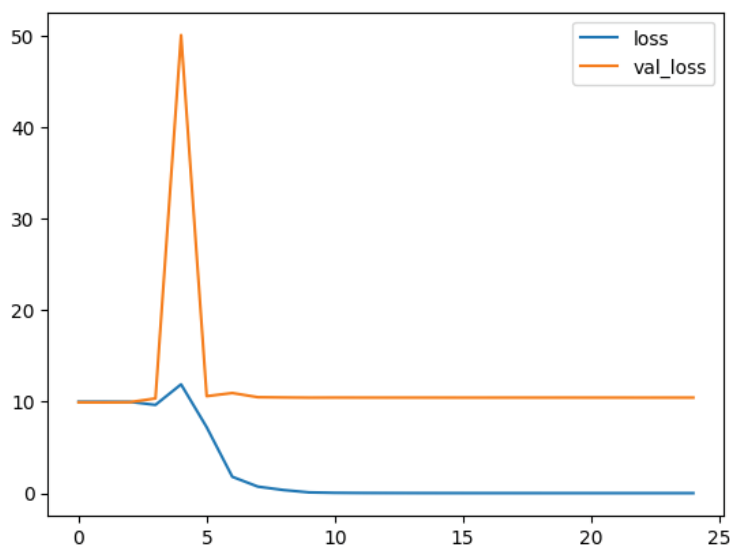
# centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating

#### 7. Treinamento do modelo
epochs = 25
r = model.fit(
    x=[train_user, train_book],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso
    validation_data=([test_user, test_book], test_ratings)
)

Epoch 1/25
101/101 - 2s - 19ms/step - loss: 10.0033 - val_loss: 9.9488
Epoch 2/25
101/101 - 0s - 3ms/step - loss: 10.0072 - val_loss: 9.9364
Epoch 3/25
101/101 - 0s - 3ms/step - loss: 9.9925 - val_loss: 9.9577
Epoch 4/25
101/101 - 0s - 3ms/step - loss: 9.6604 - val_loss: 10.3664
Epoch 5/25
101/101 - 0s - 3ms/step - loss: 11.9065 - val_loss: 50.0998
Epoch 6/25
101/101 - 0s - 3ms/step - loss: 7.2180 - val_loss: 10.6136
Epoch 7/25
101/101 - 0s - 3ms/step - loss: 1.8001 - val_loss: 10.9578
Epoch 8/25
101/101 - 0s - 3ms/step - loss: 0.7220 - val_loss: 10.5001
Epoch 9/25
101/101 - 0s - 3ms/step - loss: 0.3495 - val_loss: 10.4720
Epoch 10/25
101/101 - 0s - 3ms/step - loss: 0.0916 - val_loss: 10.4516
Epoch 11/25
101/101 - 0s - 2ms/step - loss: 0.0429 - val_loss: 10.4577
Epoch 12/25
101/101 - 0s - 3ms/step - loss: 0.0279 - val_loss: 10.4553
Epoch 13/25
...
Epoch 24/25
101/101 - 0s - 3ms/step - loss: 0.0114 - val_loss: 10.4542
Epoch 25/25
101/101 - 0s - 3ms/step - loss: 0.0112 - val_loss: 10.4538

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

```



O gráfico demonstra a evolução da "taxa de erros" conforme as épocas são executadas. Nota-se que conforme mais épocas são executadas, menor é a frequência dos erros na avaliação, conforme o esperado. A função de perda utilizada foi o MSE (Mean Squared Error), ou Erro Quadrático Médio.

```
### 9. Recomendações para o usuário 11139
# Gerar o array com o usuário único
# repete a quantidade de livros
input_usuario = np.repeat(a=11139, repeats=M)
book = np.array(list(set(book_ids)))

preds = model.predict( [input_usuario, book] )

# descentraliza as predições
rat = preds.flatten() + avg_rating

# índice da maior nota
idx = np.argmax(rat)

print("Recomendação: Livro - ", book[idx], " / ", rat[idx] , "")

Recomendação: Livro - 35263 / 5.330925 *
```

4 Deepdream

Esta prática contém uma implementação mínima do DeepDream, conforme descrito neste [post] (<https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>), do blog de Alexander Mordvintsev.

Vamos demonstrar como fazer uma rede neural "sonhar" e aprimorar os padrões surreais que ela vê em uma imagem.

![Dogception] (https://commons.wikimedia.org/wiki/Special:FilePath/Felis_cat_us-cat_on_snow.jpg)



```

### 1. Importação das bibliotecas
import tensorflow as tf
import numpy as np

import matplotlib as mpl

import IPython.display as display
import PIL.Image

### 2. Importação da imagem

### Escolha uma imagem para "sonhar"
Para este tutorial, vamos usar uma imagem de um
[felino] (https://commons.wikimedia.org/wiki/Special:FilePath/Felis\_catus-
cat\_on\_snow.jpg).

url = https://commons.wikimedia.org/wiki/Special:FilePath/Felis\_catus-
cat\_on\_snow.jpg

# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
    name = url.split('/')[-1]
    image_path = tf.keras.utils.get_file(name, origin=url)
    img = PIL.Image.open(image_path)
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)

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

# Mostra a imagem
def show(img):
    display.display(PIL.Image.fromarray(np.array(img)))

# Redução do tamanho da imagem para facilitar o trabalho da RNN
original_img = download(url, max_dim=500)
show(original_img)

```

```
display.display(display.HTML('Image:
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-
cat_on_snow.jpg"></a>'))
```

<a



```
### 3. Preparar o modelo de extração de recursos
Aqui faremos o download um modelo de classificação de imagem pré-treinado.
Usaremos o [InceptionV3](https://keras.io/api/applications/inceptionv3/)
que é semelhante ao modelo originalmente usado no DeepDream. Observe que
qualquer [modelo pré-treinado](https://keras.io/api/applications/#available-
models) funcionará, embora você precise ajustar os nomes das camadas abaixo,
caso deseje alterá-lo.
```

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

```
### A ideia no DeepDream é escolher uma camada (ou camadas) e maximizar a
"perda" de forma que a imagem cada vez mais treine as camadas.
```

```
# Maximizando as ativações das camadas
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]
```

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

```
### 4. Cálculo da perda (*loss*)
```

A perda é a soma das ativações nas camadas escolhidas.

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

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

```

    return tf.reduce_sum(losses)

### 5. Subida de gradiente (*Gradient ascent*)

Depois de ter calculado a perda para as camadas escolhidas, tudo o que resta
é calcular os gradientes em relação à imagem e adicioná-los à imagem original.

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

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

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

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

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

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

        return loss, img

deepdream = DeepDream(dream_model)

### 6. Circuito principal (*Main Loop*)
def run_deep_dream_simple(img, steps=100, step_size=0.01):

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

```

```

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

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

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

    return result

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

```



O Main Loop é o coração do processo. Ele aplica gradualmente alterações à imagem com base nos gradientes das ativações nas camadas selecionadas. O resultado é uma imagem surreal, adicionando padrões e formas abstratas.

7. Levando o modelo até um oitava

Conseguimos gerar uma imagem. Porém, há alguns problemas com esta primeira tentativa:

1. A saída é ruidosa (isso pode ser resolvido com uma perda ``tf.image.total_variation``).
2. A imagem é de baixa resolução.
3. Os padrões parecem estar acontecendo na mesma granularidade.

```

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

```



A técnica de dividir a imagem em oitavas consiste em processar a imagem em várias escalas de resolução, reduzindo e ampliando a imagem repetidamente. Desta forma as características oníricas da imagem são ampliadas.

Diferenças entre as imagens obtidas.

Main Loop: Amplifica diretamente as características reconhecidas pela rede em uma única escala. Resulta em um padrão mais uniforme.

Modelo com Oitavas: Aplica modificações em diferentes escalas, amplificando as características oníricas.

Você pode alterar o número de oitavas, escala de oitavas e camadas ativadas para alterar a aparência da sua imagem do DeepDream.

APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

- Explicação do **contexto e o público-alvo** da visualização de dados e do storytelling que será desenvolvido;

- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada; (50 pontos)**

B – RESOLUÇÃO

1. Conjunto de Dados Brutos

Os dados a seguir foram coletados por meio de uma pesquisa online com pacientes e cuidadores afetados por doenças raras no Brasil, realizada pela Crossing Connection Health. Foram entrevistados 1562 participantes.

Tabela 1: Tempo para Obtenção do Diagnóstico

Tempo	Número de Pacientes	Porcentagem
Menos de um ano	534	34.2%
De 1 a 2 anos	301	19.3%
De 2 a 4 anos	212	13.6%
Mais de 4 anos	76	4.9%
Não sei	453	29%
Sem diagnóstico	0	0%

Observação: Os participantes podiam selecionar múltiplas opções.

1.1.1 Tabela 2: Fontes de Informação Utilizadas

Fonte	Número de Pacientes	Porcentagem
Internet	1,368	87.5%
Outros pacientes	850	55.1%
Nos profissionais de saúde	702	44.9%
Universidades/escolas públicas	467	29.9%
Nas instituições do governo	174	11.1%
No sistema de saúde pública	141	9%
Outros	115	7.4%
Não busco	13	0.8%

Observação: Os participantes podiam selecionar múltiplas opções.

1.1.2 Tabela 3: Erros Médicos ou de Medicação Relatados

Tipo de Erro	Número de Pacientes	Porcentagem
Diagnóstico incorreto	869	55.6%
Erros de medicação	127	8.2%
Sem erros relatados	566	36.2%
Total	1,562	100%

1.1.3 Tabela 4: Acesso ao Tratamento

Sistema de Saúde	Número de Pacientes	Porcentagem
Pública	448	29%
Privada	520	33%
Nas duas acima	406	26%
Nenhum deles	188	12%

1.1.4 2. Contexto e Público-Alvo

Contexto

As doenças raras afetam um número reduzido de pessoas, mas suas consequências são profundas para os pacientes e suas famílias. No Brasil, estima-se que entre 13 e 15 milhões de pessoas vivam com alguma doença rara. O acesso ao diagnóstico precoce e ao tratamento adequado é um desafio significativo, muitas vezes agravado pela falta de informação, infraestrutura e suporte especializado.

Este estudo busca compreender as principais dificuldades enfrentadas por pacientes e cuidadores, analisando dados coletados em uma pesquisa com pessoas impactadas por doenças raras. A visualização desses dados tem como objetivo transformar informações numéricas em insights visuais que possam guiar ações concretas para melhorar a qualidade de vida dessas pessoas.

Público-Alvo

A análise e visualização de dados se destinam a quatro grupos principais:

1. **Formuladores de políticas públicas:** Para embasar decisões sobre melhorias no atendimento a pacientes com doenças raras e desenvolver políticas mais inclusivas e eficazes.
2. **Profissionais de saúde:** Para compreender os desafios enfrentados pelos pacientes e aprimorar abordagens clínicas, reduzindo o tempo de diagnóstico e minimizando erros.
3. **Organizações de apoio a pacientes:** Para fornecer dados que ajudem na criação de projetos e programas de suporte mais eficientes e direcionados às necessidades reais.
4. **Pesquisadores:** Para entender tendências e lacunas no atendimento a doenças raras, orientando novas pesquisas e desenvolvimentos.

O objetivo final é fornecer insights que possam auxiliar no desenvolvimento de estratégias para melhorar o acesso ao diagnóstico, tratamento e suporte para pessoas com doenças raras no Brasil.

1.1.5 3. Visualização dos Dados

Tipos de Visualização Escolhidos

Para este trabalho, foram desenvolvidas quatro visualizações complementares que, juntas, contam a história completa das dificuldades enfrentadas por pacientes com doenças raras:

1. **Gráficos de pizza:** Utilizados para mostrar proporções de um todo finito (100% dos pacientes) em relação ao tempo de diagnóstico, erros médicos e acesso ao tratamento. Ideais quando há um número pequeno de categorias (até 4-5).
2. **Gráficos de barras horizontais:** Escolhidos para comparar valores entre diferentes categorias de dificuldades enfrentadas e fontes de informação. Eficazes especialmente quando há muitas categorias ou nomes longos, pois a orientação horizontal facilita a leitura e comparação visual dos valores.

Paleta de Cores

A paleta de cores foi cuidadosamente selecionada considerando:

- Tons de laranja e azul para criar uma identidade visual coerente relacionada à área da saúde e da startup.
- Contraste suficiente para facilitar a leitura e interpretação
- Uso de cores mais vibrantes para destacar pontos críticos nos dados

Ferramenta Utilizada

Excel, devido à sua acessibilidade pois a pesquisa foi feita online e tem a funcionalidade robusta para criação de gráficos.

Visualizações

Figura 1: Gráfico de pizza mostrando o tempo para obtenção do diagnóstico, evidenciando que a maioria dos pacientes (60%) leva mais de 4 anos para ser diagnosticada.

Demorou muito tempo para voce ou a pessoa acometida chegar a um diagnóstico?
1.562 respostas

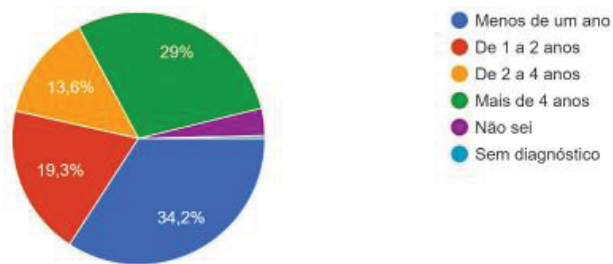


Figura 2: Gráfico de barras horizontais mostrando as fontes de informação utilizadas pelos pacientes, evidenciando a predominância da internet (80%).

Você ou o paciente busca mais informações onde?
1.562 respostas

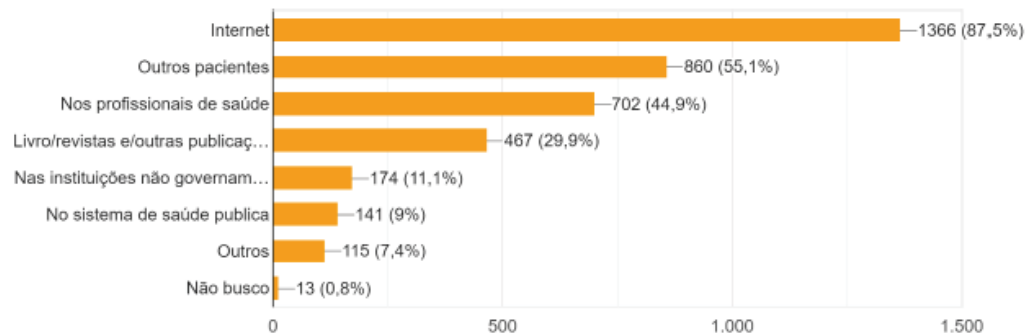


Figura 3: Gráfico de pizza ilustrando os erros médicos ou de medicação relatados, demonstrando que 60% dos pacientes relataram algum tipo de erro.

Você já viveu situação de erro médico ou de medicação?
1.562 respostas

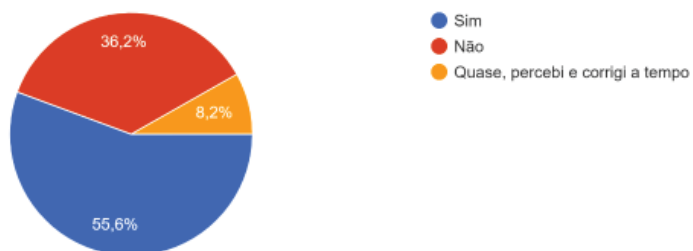
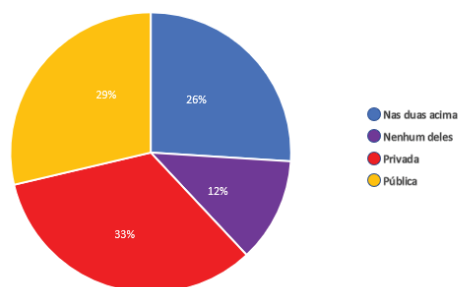


Figura 4: Gráfico de pizza apresentando o acesso ao tratamento, mostrando a dependência da rede privada de saúde (33%).

Você ou o paciente encontra tratamento na rede?



1.1.6 4. Narrativa/Storytelling

1.2 A JORNADA INVISÍVEL: O DESAFIO DAS DOENÇAS RARAS NO BRASIL

O Longo Caminho até o Diagnóstico

Imagine esperar anos para descobrir o que está causando seus sintomas. Esta é a realidade para a maioria dos brasileiros com doenças raras. Nossa visualização revela um dado alarmante: 60% dos pacientes esperam mais de 4 anos para obter um diagnóstico definitivo. Um em cada quatro pacientes (25%) aguarda mais de 5 anos.

Este atraso não representa apenas números em um gráfico – são anos de sofrimento, tratamentos inadequados e progressão da doença. Para muitas famílias, significa uma peregrinação por consultórios, exames e opiniões médicas divergentes, com impacto significativo na saúde física e mental.

Obstáculos no Caminho

O que causa este atraso diagnóstico? Nossa visualização das principais dificuldades enfrentadas nos dá pistas importantes. A escassez de especialistas (85%) encabeça a lista, seguida pela falta de informação (75%).

Estes dados revelam uma lacuna crítica em nosso sistema de saúde: não estamos preparados para identificar e tratar condições que fogem do comum. A raridade destas doenças cria um ciclo vicioso – poucos casos diagnosticados levam a pouca experiência clínica, resultando em diagnósticos tardios ou equivocados.

Em Busca de Respostas

Diante da escassez de informação no sistema de saúde, para onde os pacientes se voltam? Nossa visualização mostra claramente que a internet (80%) se tornou o principal recurso, seguida pelos próprios pacientes (55%).

Este padrão sugere uma oportunidade crítica: a necessidade de plataformas confiáveis de informação online sobre doenças raras. Quando os pacientes precisam recorrer primariamente à internet, é essencial garantir que encontrem conteúdo de qualidade e cientificamente validado.

O Preço dos Erros

Talvez o dado mais preocupante de nossa visualização seja que 60% dos pacientes relatam ter sofrido algum tipo de erro médico. Diagnósticos incorretos representam 45% destes casos, enquanto erros de medicação respondem por 30%.

Estes números não são apenas estatísticos – representam tratamentos inadequados, tempo perdido e potencial progressão da doença. Para doenças raras, onde o tempo é frequentemente um fator crítico, cada erro pode ter consequências irreversíveis.

Público vs. Privado: Uma Questão de Acesso

Nossa visualização sobre o acesso ao tratamento revela outra faceta preocupante: quase metade dos pacientes (33%) depende exclusivamente da rede privada de saúde. Apenas 29% conseguem ser atendidos pelo SUS, enquanto 26% utilizam ambos os sistemas.

Estes dados levantam questões importantes sobre equidade no acesso à saúde. Se a maioria dos pacientes com doenças raras depende do sistema privado, o que acontece com aqueles que não podem pagar? Como garantir que todos tenham acesso aos cuidados especializados necessários?

Transformando Dados em Ação

Os dados visualizados não apenas ilustram problemas, mas apontam caminhos para soluções. Com base nestes insights, podemos propor quatro ações concretas:

1. **Capacitação para profissionais de saúde:** Programas específicos de treinamento em doenças raras, especialmente para médicos da atenção primária, podem reduzir erros diagnósticos e agilizar encaminhamentos apropriados.
2. **Ampliação de canais de informação confiáveis:** Desenvolvimento de plataformas digitais com conteúdo validado cientificamente sobre doenças raras, preenchendo a lacuna de informação identificada.
3. **Fortalecimento da rede de apoio:** Maior suporte às associações de pacientes, reconhecendo seu papel crucial como fonte de informação e suporte emocional.
4. **Melhoria no acesso a medicamentos e tratamentos:** Políticas públicas para garantir que pacientes de todos os níveis socioeconômicos tenham acesso aos cuidados necessários, independentemente de utilizarem o sistema público ou privado.

Conclusão

As visualizações apresentadas transformam números em histórias – histórias de pessoas reais enfrentando desafios extraordinários. Ao compreendermos visualmente a jornada dos pacientes com doenças raras, podemos desenvolver intervenções mais eficazes e humanas.

Este trabalho de visualização de dados não apenas informa, mas convoca à ação. Para os formuladores de políticas, profissionais de saúde, associações de apoio e pesquisadores, estes insights visuais oferecem um mapa para melhorar a qualidade de vida de milhões de brasileiros afetados por doenças raras.

A jornada pode ser longa, mas com dados bem visualizados e uma narrativa clara, podemos navegar em direção a um sistema de saúde mais inclusivo e preparado para enfrentar o desafio das doenças raras no Brasil.

Referência: Crossing Connection Health. (2024). Pesquisa online sobre Doenças Raras no Brasil.

APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

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

2) Compare a representação de dois modelos vetoriais

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

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

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

B – RESOLUÇÃO

1 Algoritmo Genético

```
[1]: # Importação das bibliotecas
import random
import numpy as np
import matplotlib.pyplot as plt

[2]: # Função para gerar cidades aleatórias em um espaço 100x100
def generate_cities(n=100, max_coord=100, min_coord=0):
    #return {i: (random.randint(min_coord, max_coord), random.
    ↪ randint(min_coord, max_coord)) for i in range(n)}
    return np.round(np.random.rand(n, 2) * max_coord).astype(int)
# Gerar cidades
cities = generate_cities()
# city_names = list(cities.keys())
city_names = list(range(100))
first_city = random.choice(city_names)
cities[first_city] = (0, 100)
print('Cidade inicial: ' + str(first_city))
print(cities)
Cidade inicial: 73

[3]: # Função para calcular a distância euclidiana entre duas cidades
def distance(city1, city2):
    x1, y1 = cities[city1]
    x2, y2 = cities[city2]
    return np.sqrt((x1 - x2)**2 + (y1 - y2)**2)

[4]: # Função para calcular a distância total de um caminho
def total_distance(path):
    dist = sum(distance(path[i], path[i + 1]) for i in range(len(path) - 1))
    dist += distance(path[-1], path[0]) # Retorno à cidade inicial
    return dist

[5]: # Criando uma população inicial aleatória
def create_population(size):
    other_cities = city_names.copy()
    other_cities.remove(first_city)
```

```

return [[first_city] + random.sample(other_cities, len(other_cities)) for
_ in range(size)]

[6]: # Função de avaliação (fitness) - quanto menor a distância, melhor
def fitness(path):
return 1 / total_distance(path)

[7]: # Seleção por torneio: escolhe o melhor de um grupo aleatório
def selection(population, k=3):
    tournament = random.sample(population, k)
    return min(tournament, key=total_distance)

[8]: # Crossover: PMX (Partially Mapped Crossover)
def crossover(parent1, parent2):
    size = len(parent1)
    start, end = sorted(random.sample(range(size), 2))
    child = [None] * size
    child[start:end] = parent1[start:end]
    mapping = {parent1[i]: parent2[i] for i in range(start, end)}
    for i in range(size):
        if child[i] is None:
            val = parent2[i]
            while val in mapping:
                val = mapping[val]
            child[i] = val
    return child
4

[9]: # Mutação: Troca duas cidades aleatoriamente
def mutate(path, mutation_rate=0.01):
    if random.random() < mutation_rate:
        i, j = random.sample(range(len(path)), 2)
        path[i], path[j] = path[j], path[i]
    return path

[10]: # Algoritmo Genético
def genetic_algorithm(pop_size=100, generations=1000, mutation_rate=0.01,
crossover_prob=0.9):
    population = create_population(pop_size)
    first_population = sorted(population.copy(), key=total_distance)
    for gen in range(generations):
        population = sorted(population, key=total_distance) # Ordena por
        aptidão
        new_population = [population[0]] # Elitismo: mantém o melhor indivíduo
        while len(new_population) < pop_size:
            parent1 = selection(population)
            if random.random() < crossover_prob:
                parent2 = selection(population)
                child = crossover(parent1, parent2)
            else:
                child = parent1
            child = mutate(child, mutation_rate)
            new_population.append(child)
            population = new_population
            if ((1 + gen) % 50 == 0 or gen == 0):
                print(f"Geração {gen+1}, Melhor Distância:
                {total_distance(population[0]):.2f}")
    return population[0], first_population[0]

```

```

[11]: # Executando o algoritmo
best_path, first_path = genetic_algorithm()
print("\nPrimeira Melhor Solução:", first_path)
print("Distância mínima:", total_distance(first_path))
print("\nMelhor Solução:", best_path)
print("Distância mínima:", total_distance(best_path))
Geração 1, Melhor Distância: 4832.79
Geração 50, Melhor Distância: 3306.51
Geração 100, Melhor Distância: 3171.91
5
Geração 150, Melhor Distância: 3135.53
Geração 200, Melhor Distância: 3044.33
Geração 250, Melhor Distância: 3010.79
Geração 300, Melhor Distância: 2997.33
Geração 350, Melhor Distância: 2915.93
Geração 400, Melhor Distância: 2787.57
Geração 450, Melhor Distância: 2678.31
Geração 500, Melhor Distância: 2653.84
Geração 550, Melhor Distância: 2649.43
Geração 600, Melhor Distância: 2643.82
Geração 650, Melhor Distância: 2560.75
Geração 700, Melhor Distância: 2511.26
Geração 750, Melhor Distância: 2498.41
Geração 800, Melhor Distância: 2397.73
Geração 850, Melhor Distância: 2397.73
Geração 900, Melhor Distância: 2376.52
Geração 950, Melhor Distância: 2368.96
Geração 1000, Melhor Distância: 2326.95
Primeira Melhor Solução: [73, 84, 10, 77, 25, 4, 63, 54, 22, 55, 96, 33,
57, 14,
48, 81, 29, 71, 37, 39, 60, 1, 6, 23, 16, 42, 9, 31, 0, 58, 64, 67, 26, 59,
62,
13, 15, 87, 38, 53, 66, 43, 45, 50, 24, 75, 65, 47, 92, 19, 99, 80, 72, 97,
36,
35, 40, 17, 49, 27, 56, 70, 8, 5, 68, 52, 11, 44, 89, 46, 32, 7, 91, 85,
74, 30,
86, 34, 3, 21, 28, 18, 98, 51, 78, 20, 88, 90, 61, 94, 79, 41, 12, 82, 69,
76,
93, 83, 95, 2]
Distância mínima: 4832.7939942534
Melhor Solução: [24, 41, 32, 18, 96, 83, 67, 28, 8, 80, 77, 52, 31, 19, 9,
40,
5, 39, 7, 85, 79, 73, 57, 6, 2, 72, 27, 88, 35, 29, 90, 4, 84, 68, 82, 63,
61,
74, 55, 20, 69, 1, 58, 81, 89, 37, 70, 34, 10, 30, 15, 23, 45, 94, 60, 17,
42,
59, 50, 54, 97, 93, 64, 11, 49, 0, 22, 71, 76, 75, 65, 62, 3, 47, 25, 14,
86,
36, 91, 92, 21, 33, 95, 87, 43, 99, 46, 56, 51, 98, 26, 12, 16, 13, 48, 78,
66,
53, 44, 38]
Distância mínima: 2326.9489389022115

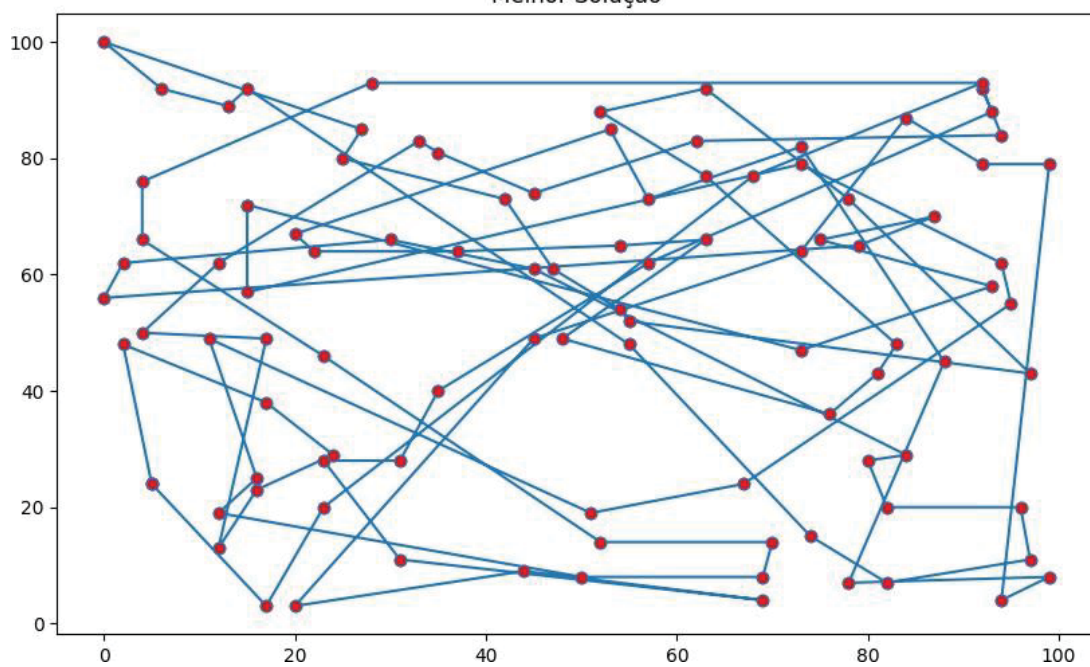
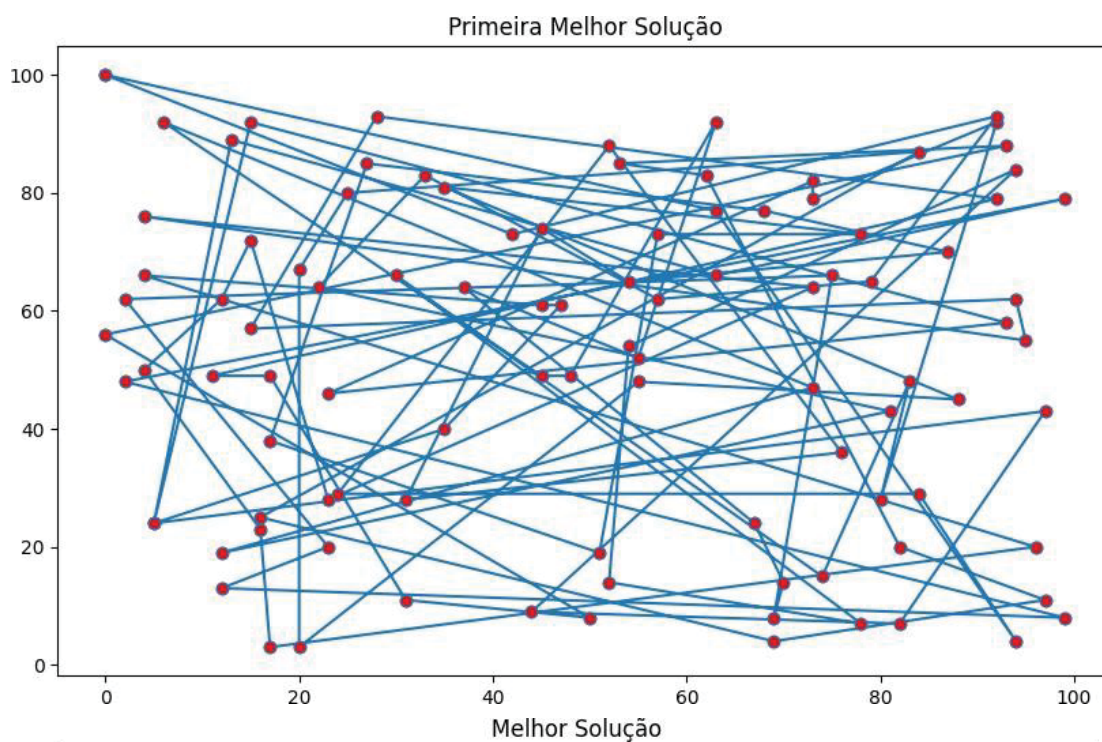
[12]: # Função para plotar o percurso
def plot_route(coords, route, title):
    route.append(route[0])
    ordered_coords = np.array([coords[i] for i in route])
    plt.figure(figsize=(10, 6))
    plt.scatter(coords[:, 0], coords[:, 1], color='red', marker='o')
    plt.plot(ordered_coords[:, 0], ordered_coords[:, 1], linestyle='-', _
↳marker='o', markerfacecolor='red')

```

```

plt.title(title)
plt.show()
# Exibir os resultados
6
plot_route(cities, first_path, "Primeira Melhor Solução")
plot_route(cities, best_path, "Melhor Solução")

```



2 Compare a representação de dois modelos vetoriais

```

[13]: # Importações
import numpy as np
import matplotlib.pyplot as plt

```

```

import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
import re

[14]: # Texto base para vetorização
text = """O poeta é um fingidor.
Finge tão completamente
Que chega a fingir que é dor
A dor que deveras sente.
E os que leem o que escreve,
Na dor lida sentem bem,
Não as duas que ele teve,
Mas só a que eles não têm.
E assim nas calhas de roda
Gira, a entreter a razão,
Esse comboio de corda
Que se chama coração."""

[15]: # Separar em sentenças (considerando quebras de linha e pontos como
separadores)
sentence = [s.strip() for s in re.split(r'[\n]+', text) if s.strip()]
print_sentence = sentence.copy()
print("Sentenças identificadas:")
for i, print_sentence in enumerate(print_sentence):
    print(f"Texto {i+1}: {print_sentence}")
Sentenças identificadas:
Texto 1: O poeta é um fingidor
Texto 2: Finge tão completamente
Texto 3: Que chega a fingir que é dor
Texto 4: A dor que deveras sente
Texto 5: E os que leem o que escreve,
Texto 6: Na dor lida sentem bem,
Texto 7: Não as duas que ele teve,
Texto 8: Mas só a que eles não têm
Texto 9: E assim nas calhas de roda
Texto 10: Gira, a entreter a razão,
Texto 11: Esse comboio de corda
8
Texto 12: Que se chama coração

[16]: # Vetorização com TF-IDF
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(sentence).toarray()

[17]: # Redução de dimensionalidade com PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

[18]: # Mostrar os vetores reduzidos
print("\nCoordenadas PCA das sentenças:")
for i, (pc1, pc2) in enumerate(X_pca):
    print(f"Texto {i+1}: PC1={pc1:.2f}, PC2={pc2:.2f}")
Coordenadas PCA das sentenças:
Texto 1: PC1=-0.30, PC2=0.00
Texto 2: PC1=-0.30, PC2=0.00
Texto 3: PC1=0.51, PC2=0.20
Texto 4: PC1=0.39, PC2=0.41
Texto 5: PC1=0.41, PC2=-0.14
Texto 6: PC1=-0.00, PC2=0.65
Texto 7: PC1=0.18, PC2=-0.48

```

Texto 8: $PC1=0.18$, $PC2=-0.48$
 Texto 9: $PC1=-0.49$, $PC2=-0.01$
 Texto 10: $PC1=-0.30$, $PC2=0.00$
 Texto 11: $PC1=-0.49$, $PC2=-0.01$
 Texto 12: $PC1=0.21$, $PC2=-0.15$

```

[19]: # Plot da projeção PCA
sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))
for i, txt in enumerate(sentence):
    plt.scatter(X_pca[i, 0], X_pca[i, 1], label=f'Texto {i+1}')
    plt.text(X_pca[i, 0], X_pca[i, 1], f'{i+1}', fontsize=12, ha='right')
    plt.plot([0, X_pca[i, 0]], [0, X_pca[i, 1]], color='gray', linestyle='--')
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.title("Projeção PCA dos Textos")
plt.legend()
plt.show()

```

