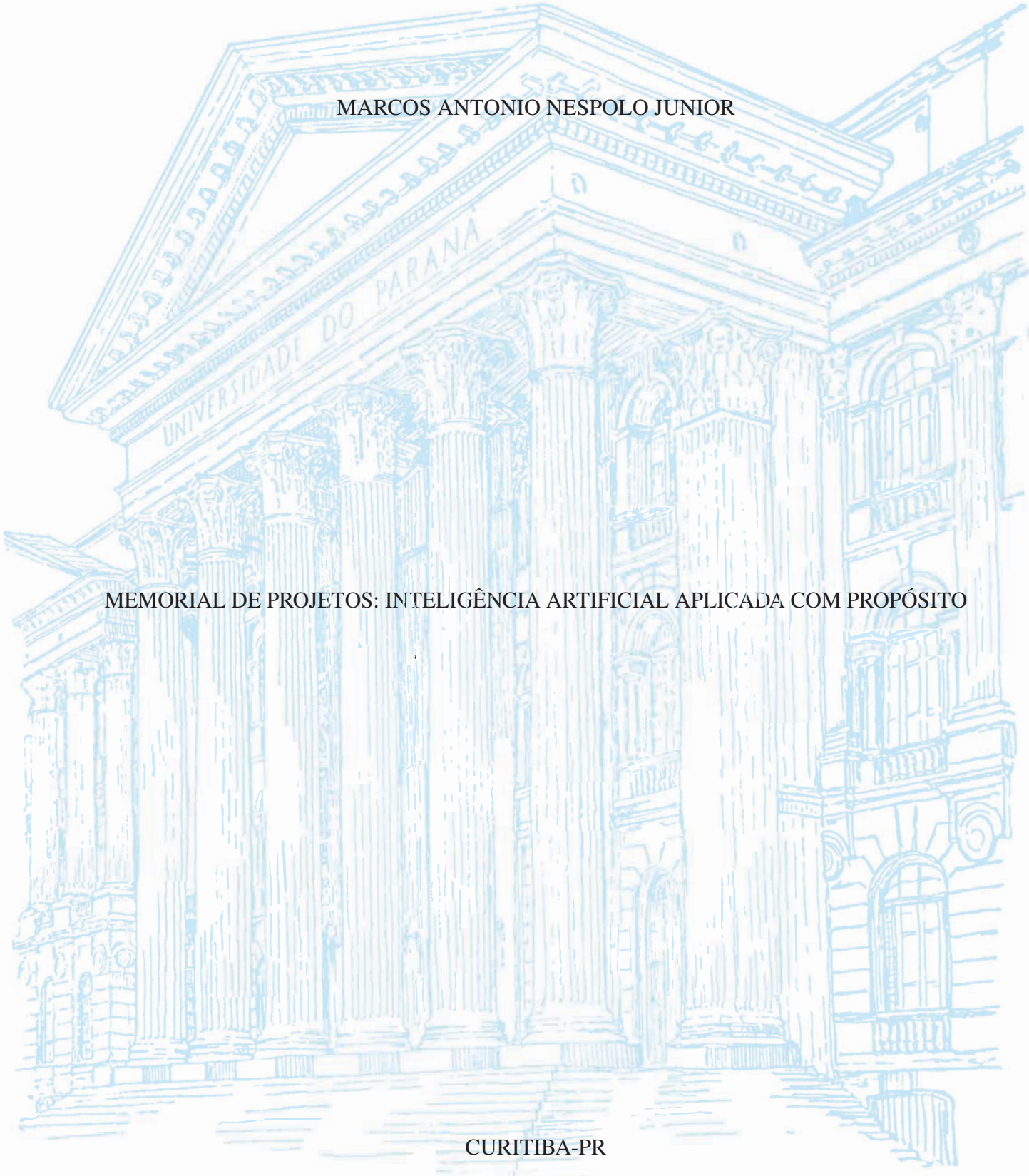
UNIVERSIDADE FEDERAL DO PARANÁ

MARCOS ANTONIO NESPOLO JUNIOR



MEMORIAL DE PROJETOS: INTELIGÊNCIA ARTIFICIAL APLICADA COM PROPÓSITO

CURITIBA-PR

2025

MARCOS ANTONIO NESPOLO JUNIOR

MEMORIAL DE PROJETOS: INTELIGÊNCIA ARTIFICIAL APLICADA COM PROPÓSITO

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.

Área de concentração: *Inteligência Artificial Aplicada*.

Orientador: Profª. Drª. Rafaela Mantovani Fontana.

CURITIBA-PR

2025

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **MARCOS ANTONIO NESPOLO JUNIOR**, intitulada: **MEMORIAL DE PROJETOS: INTELIGÊNCIA ARTIFICIAL APLICADA COM PROPÓSITO**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

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

Curitiba, 04 de Outubro de 2025.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

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

AGRADECIMENTOS

Agradeço primeiramente à minha família, em especial aos meus avós, Almerinda e Antonio, que, com amor e sacrifício, dedicaram suas vidas para suavizar o peso da minha e abriram caminhos que sozinho eu não conseguiria trilhar.

Aos meus amigos, que foram apoio essencial nesta caminhada, agradeço pela companhia, pelo incentivo e pelos momentos de descontração que me ajudaram a manter o equilíbrio enquanto conciliava trabalho e duas pós-graduações ao mesmo tempo.

Por fim, agradeço à vida, ao destino — ou a Deus — por ter colocado no meu caminho os desafios que me lapidaram, as oportunidades que me abriram horizontes e as conquistas que deram sentido à jornada, permitindo-me concluir mais esta etapa da minha trajetória.

"Faça o teu melhor, nas condições que você tem, enquanto você não tem condições melhores para fazer melhor ainda."
Mário Sérgio Cortella

RESUMO

O conceito de IA aplicada com propósito consolidou-se como uma resposta ao uso acrítico de modelos de alto desempenho, priorizando decisões sobre quando, como e por quê aplicar IA de forma útil, segura e alinhada a valores. A partir das disciplinas do curso de pós-graduação, foram exploradas diferentes dimensões da IA aplicada, incluindo fundamentos de IA, aprendizado de máquina e deep learning; estatística e arquitetura de dados; big data; programação em Python e R; visualização de dados e storytelling; visão computacional e frameworks; tópicos em IA (lógica fuzzy, algoritmos genéticos e séries temporais); e gestão de projetos de IA. Essas abordagens evidenciaram como a IA aplicada se manifesta tanto no nível técnico, por meio de formulação do problema, preparação e qualidade dos dados, modelagem e validação, quanto no organizacional, com ênfase em métricas alinhadas ao contexto, comunicação com as partes interessadas e acompanhamento de resultados. Além dos aspectos metodológicos, discutiram-se implicações éticas e sociais, como privacidade, viés algorítmico, explicabilidade e transparência. Em síntese, o memorial destaca que IA aplicada com propósito não é apenas construir modelos, mas integrar pessoas, dados, processos e tecnologia para entregar valor verificável, com responsabilidade e impacto real.

Palavras-chave: inteligência artificial aplicada; aprendizado de máquina; aprendizado profundo; arquitetura de dados; ética em IA.

ABSTRACT

Purpose-driven Applied AI emerges as a response to the uncritical use of high-performing models, prioritizing decisions about when, how, and why to apply AI in useful, safe, and value-aligned ways. Drawing on graduate-level coursework, the report covers multiple dimensions of applied AI: foundations in AI, machine learning, and deep learning; statistics and data architecture; big data; programming in Python and R; data visualization and storytelling; computer vision and frameworks; advanced topics (fuzzy logic, genetic algorithms, time series); and AI project management. These tracks show how applied AI operates both technically—through problem framing, data preparation and quality, modeling, and validation—and organizationally, emphasizing context-aligned metrics, stakeholder communication, and results follow-up. Beyond methodology, the study examines ethical and social implications, including privacy, algorithmic bias, explainability, and transparency. In sum, purpose-driven applied AI is not merely about building models, but about integrating people, data, processes, and technology to deliver verifiable value with responsibility and real impact.

Keywords: applied artificial intelligence; machine learning; deep learning; data architecture; AI ethics.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTELIGÊNCIA ARTIFICIAL APLICADA

A IA Aplicada representa o esforço de transformar métodos computacionais em soluções de valor para problemas concretos. Ao longo do curso, essa perspectiva foi trabalhada tanto em discussões conceituais quanto em atividades práticas, buscando articular teoria, experimentação e impacto real.

1.1 DEFINIÇÃO

A Inteligência Artificial Aplicada refere-se à utilização sistemática de métodos de IA para resolver problemas reais em contextos específicos. Seu foco está na conexão entre modelos computacionais, dados e processos organizacionais, com o objetivo de gerar impacto tangível em produtos, serviços e decisões. Nesse sentido, a ênfase recai sobre a transformação de capacidades algorítmicas em soluções operacionais de valor mensurável, seja no setor privado ou em políticas públicas (Russell e Norvig, 2022; Provost e Fawcett, 2013; Kuhn e Johnson, 2013).

Do ponto de vista técnico, essa abordagem envolve diversas estratégias de aprendizado de máquina, incluindo aprendizado supervisionado, não supervisionado e por reforço (Mitchell, 1997). Também são incorporadas arquiteturas modernas de aprendizado profundo, especialmente em aplicações como visão computacional, processamento de linguagem natural e análise de séries temporais (Bishop, 2006; Goodfellow, Bengio e Courville, 2016). Nessas situações, a modelagem não é um fim em si, mas um meio para atingir objetivos como previsão, classificação, recomendação, detecção de anomalias, otimização e suporte à decisão.

O ciclo de vida da IA aplicada é iterativo e orientado à operação. Ele compreende as etapas de descoberta e preparação dos dados, modelagem e validação, implantação e monitoramento contínuo. Esse fluxo exige práticas de engenharia robustas, capazes de lidar com desafios como acoplamentos entre componentes, mudanças nos dados (*data drift*), e dependências de infraestrutura. Esses fatores compõem o que se denomina como *dívida técnica* em sistemas de aprendizado de máquina (Sculley *et al.*, 2015). Além disso, a atuação colaborativa entre equipes de produto, dados e engenharia é essencial para garantir a qualidade e a continuidade das soluções desenvolvidas (Amershi *et al.*, 2019).

A avaliação dos sistemas de IA deve considerar tanto critérios estatísticos (como acurácia, erro, robustez e capacidade de generalização) quanto métricas de impacto organizacional, como

valor entregue, custo de operação, riscos envolvidos e conformidade regulatória. A comparação entre modelos deve ser conduzida com rigor metodológico e validação compatível com o problema em questão (Kuhn e Johnson, 2013; Bishop, 2006). Ao mesmo tempo, é importante reconhecer os limites dos paradigmas adotados e os vieses que podem emergir durante o processo de modelagem (Breiman, 2001).

Finalmente, a aplicação responsável da IA exige atenção a aspectos éticos e de governança. Entre os principais temas estão a privacidade e segurança dos dados, o viés algorítmico, a explicabilidade dos modelos e a documentação adequada de todo o ciclo de vida. Boas práticas incluem o uso de *model cards* e *datasheets* para promover a transparência, bem como técnicas de interpretação local que possibilitam a análise e compreensão das decisões automatizadas (Mitchell *et al.*, 2019; Gebru *et al.*, 2021; Ribeiro, Singh e Guestrin, 2016; Barocas e Selbst, 2016). Em suma, a Inteligência Artificial Aplicada busca articular pessoas, dados, processos e tecnologia para entregar soluções eficazes, seguras e sustentáveis.

1.2 INTELIGÊNCIA ARTIFICIAL APLICADA COM PROPÓSITO

Ao longo do curso de pós-graduação, enfatizou-se a importância de compreender **quando**, **como** e **por quê** aplicar inteligência artificial. Antes de qualquer linha de código, trabalhou-se a formulação do problema e a hipótese de valor: identificar se a aplicação de IA é pertinente, quais ganhos são esperados, quais custos e restrições estão envolvidos e quais partes interessadas serão impactadas. A partir dessa análise inicial, é possível estabelecer um ciclo consistente de desenvolvimento: preparo dos dados, construção e validação do modelo, entrega e monitoramento dos resultados, sempre com critérios explícitos de sucesso e risco.

No que se refere ao **como** aplicar, os conteúdos das disciplinas de Introdução à IA, Aprendizado de Máquina e *Deep Learning* consolidaram tarefas e técnicas fundamentais, como regressão, classificação, agrupamento, associação e aprendizado por reforço, além de arquiteturas contemporâneas voltadas a visão computacional, linguagem natural e séries temporais (Russell e Norvig, 2022; Mitchell, 1997; Goodfellow, Bengio e Courville, 2016; Bishop, 2006). A ênfase recaiu sobre o alinhamento entre objetivo, métrica e contexto: definir *baselines*, escolher indicadores pertinentes ao problema e interpretar os resultados de forma a subsidiar decisões, e não apenas buscar métricas de desempenho isoladas (Kuhn e Johnson, 2013; Provost e Fawcett, 2013).

A dimensão dos dados estruturou a qualidade da aplicação: técnicas de limpeza, codificação, imputação, construção e seleção de atributos, redução de correlações e mitigação de ruído foram integradas à análise estatística descritiva, testes paramétricos e não paramétricos, análise de viés/variância e comparação de modelos (Kuhn e Johnson, 2013; Bishop, 2006). A disciplina de *Big Data* ampliou a perspectiva para cenários de alto volume, variedade e velocidade, abordando estratégias de armazenamento, processamento e integração com *data lakes*, essenciais quando escala e latência impõem restrições técnicas relevantes.

Para assegurar que o **por quê** permaneça claro e comunicável, disciplinas como Visualização de Dados e *Storytelling* foram centrais na transformação de evidências em narrativas acionáveis, voltadas tanto ao público técnico quanto ao executivo. O uso de Python e R permitiu prototipagem ágil e experimentos reprodutíveis. A disciplina de Visão Computacional abordou desde aquisição e pré-processamento de imagens até *transfer learning* e *fine-tuning* com redes convolucionais. Já os *Frameworks* de IA organizaram o uso de bibliotecas consolidadas para tarefas como regressão, classificação, séries temporais, CNNs, RNNs e modelos de linguagem. Tópicos em IA ampliou o repertório com lógica *fuzzy*, algoritmos genéticos e métodos especializados para séries temporais. Por sua vez, Gestão de Projetos de IA conectou os aspectos técnicos à organização, abordando *workflows*, escopo, riscos, critérios de aceite, documentação e a dinâmica entre equipes, encurtando a distância entre protótipo e solução real.

O componente ético e social reforçou que aplicar IA de forma responsável é indissociável das perguntas sobre **quando**, **como** e **por quê**. Foram discutidos temas como privacidade, segurança de dados, viés algorítmico, explicabilidade, autonomia e responsabilização. Instrumentos como *model cards* e *datasheets*, aliados a técnicas de interpretação local, viabilizam maior transparência e inspeção de decisões (Barocas e Selbst, 2016; Mitchell *et al.*, 2019; Gebru *et al.*, 2021; Ribeiro, Singh e Guestrin, 2016). Em aplicações com grande impacto, como visão computacional, linguagem natural e análise em larga escala, é fundamental equilibrar os benefícios de eficiência e suporte à decisão com salvaguardas éticas e comunicação clara de limitações e incertezas a clientes, usuários e stakeholders.

Em síntese, os projetos realizados ao longo do curso formaram uma base sólida para aplicar IA com propósito. Um profissional capacitado deve ser capaz de responder, com criticidade e responsabilidade, às três questões fundamentais: **quando** aplicar IA (viabilidade e valor), **como** aplicá-la (dados, métodos e validação) e **por quê** aplicá-la (impacto e responsabilidade). Mais

do que construir modelos de alto desempenho, trata-se de decidir e agir de forma útil, segura e alinhada aos valores da organização e da sociedade.

REFERÊNCIAS

AMERSHI, S. *et al.* Software engineering for machine learning: A case study. In: *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. [S.l.: s.n.], 2019. p. 291–300.

BAROCAS, S.; SELBST, A. D. Big data's disparate impact. *California Law Review*, v. 104, n. 3, p. 671–732, 2016.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. [S.l.]: Springer, 2006.

BREIMAN, L. Statistical modeling: The two cultures. *Statistical Science*, v. 16, n. 3, p. 199–231, 2001.

GEBRU, T. *et al.* Datasheets for datasets. *Communications of the ACM*, v. 64, n. 12, p. 86–92, 2021.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016.

KUHN, M.; JOHNSON, K. *Applied Predictive Modeling*. [S.l.]: Springer, 2013.

MITCHELL, M. *et al.* Model cards for model reporting. In: *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*. [S.l.: s.n.], 2019. p. 220–229.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill, 1997.

PROVOST, F.; FAWCETT, T. *Data Science for Business*. [S.l.]: O'Reilly Media, 2013.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should i trust you?": Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. [S.l.: s.n.], 2016. p. 1135–1144.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach, 4th US ed.* [S.l.]: Pearson, 2022.

SCULLEY, D. *et al.* Hidden technical debt in machine learning systems. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2015. p. 2503–2511.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A - ENUNCIADO 1 ChatGPT

a) **(6,25 pontos)** Pergunte ao ChatGPT o que é Inteligência Artificial e cole aqui o resultado.

b) **(6,25 pontos)** Dada essa resposta do ChatGPT, classifique usando as 4 abordagens vistas em sala. Explique o porquê.

c) **(6,25 pontos)** Pesquise sobre o funcionamento do ChatGPT (sem perguntar ao próprio ChatGPT) e escreva um texto contendo no máximo 5 parágrafos. Cite as referências.

d) **(6,25 pontos)** Entendendo o que é o ChatGPT, classifique o próprio ChatGPT usando as 4 abordagens vistas em sala. Explique o porquê.

2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.

Figura 1.1 – EXERCÍCIO 2: BUSCA HEURÍSTICA - MELHOR CAMINHO PARA BUCHAREST

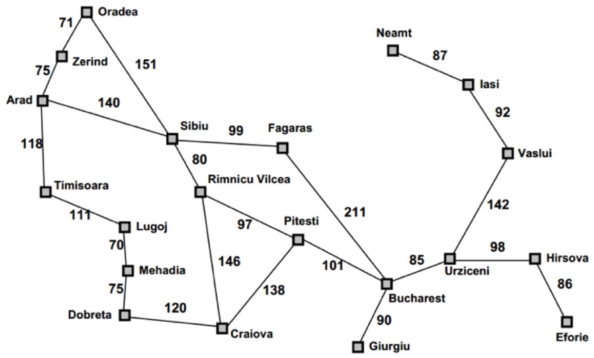


Tabela 1.1: Valores de h_{DLR} — distâncias em linha reta para Bucareste.

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

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

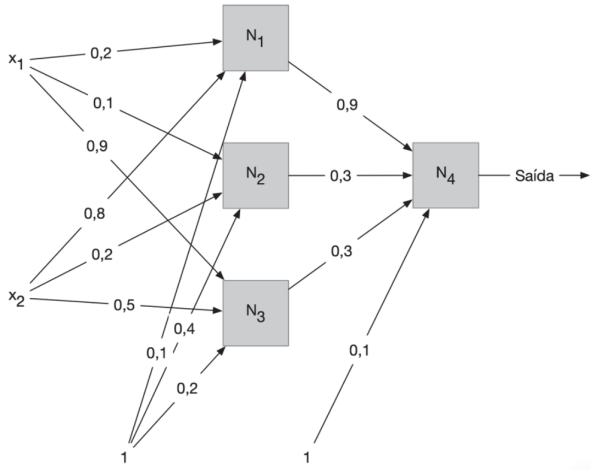
   Equivalência Bicondicional: $(\alpha \leftrightarrow \beta)$ equivale a $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$

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:

Figura 1.2 – EXERCÍCIO 4: REDE NEURAL ARTIFICIAL



Os neurônios N_1, N_2 e N_3 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:

a) **(6,25 pontos)** Valor de saída do neurônio N_1

b) **(6,25 pontos)** Valor de saída do neurônio N_2

c) **(6,25 pontos)** Valor de saída do neurônio N_3

d) **(6,25 pontos)** Valor de saída da rede como um todo

B - RESOLUÇÃO

1 ChatGPT

a) Inteligência Artificial (IA) refere-se à capacidade de uma máquina ou programa de computador realizar tarefas que normalmente exigiria inteligência humana. Essas tarefas incluem aprendizado, raciocínio, percepção, reconhecimento de padrões, compreensão da linguagem natural e tomada de decisões. O objetivo da Inteligência Artificial é criar sistemas que possam executar funções complexas de forma autônoma, imitando ou replicando a inteligência humana. Existem dois tipos principais de IA: IA fraca (ou estreita) e IA forte (ou geral).

1. IA fraca (estreita): Refere-se a sistemas especializados em realizar tarefas específicas sem possuir a capacidade de generalizar para diferentes domínios. Exemplos incluem assistentes virtuais, reconhecimento de voz, carros autônomos e sistemas de recomendação.

2. IA forte (geral): Representa um nível mais avançado de inteligência, comparável à inteligência humana em termos de habilidades cognitivas e adaptação a diferentes contextos. Até o momento, a IA forte permanece mais uma aspiração do que uma realidade prática.

As técnicas comuns em IA incluem aprendizado de máquina (*machine learning*), redes neurais, processamento de linguagem natural, visão computacional e algoritmos de otimização. O campo da Inteligência Artificial continua a evoluir rapidamente, com aplicações em diversas áreas, como saúde, finanças, transporte, manufatura e entretenimento.

b) É categorizável na abordagem "agir como seres humanos", pois no texto o chat GPT ressaltou a imitação de habilidades inerentemente humanas ao tentar definir o conceito de Inteligência Artificial.

c) Como o ChatGPT funciona: ChatGPT Sigla para *Generative Pré-Trained Transformer* (Transformador pré-treinado generativo, em tradução livre) é um sistema ou um modelo de linguagem baseado em inteligência artificial mais especificamente com as técnicas de aprendizado de máquinas, redes neurais, aprendizado por reforço com *feedback* humano (sigla em inglês RHLF) e processamento de linguagem natural usados com o foco em diálogos virtuais. A seguir vamos falar sobre cada uma das partes do algoritmo GPT. O ChatGPT usa como fonte de dados para manter os seus diálogos virtuais e o contexto das mesmas um conjunto de informações disponíveis e acessíveis principalmente através da internet.

O generativo pré-treinado, o GP de GPT, é uma técnica onde ChatGPT recebe uma grande quantidade de regras de linguagens e dados não rotulados. O GPT é então deixado sem supervisão para aprender sozinho sobre as regras e os relacionamentos que governam as linguagens.

A letra T de GPT, *Transfomer architecture*, refere-se à arquitetura do modelo, a qual é baseada na arquitetura *Transformer*. Agora, todos os dados pré-treinados são usados para criar uma rede neural de aprendizagem profunda, permitindo ao ChatGPT aprender padrões e relacionamentos de textos, dando ao ChatGPT a habilidade de criar respostas como humanos e de predizer qual texto vem em seguida em uma dada sentença. O *Transformer* é capaz de ler cada palavra de uma sentença e comparar cada palavra com as outras palavras da mesma sentença. Isso permite que ele direcione a sua atenção à palavra mais importante da sentença. Esse processo é conhecido como *self-attention*. Com isso, é importante ressaltar que o ChatGPT não entende o que é dito e o que ele responde.

No início, a rede neural do ChatGPT não estava inteiramente segura para ser lançada ao público, pois ela foi treinada e sumariamente através de dados da internet aberta sem nenhuma orientação. Então para que o ChatGPT pudesse dar respostas coerentes, sensatas e seguras ao público, foi otimizado com uma técnica chamada aprendizado por reforço com *feedback* humano (sigla em inglês RHLF). Basicamente são demonstrados dados que guiam a rede neural em como responder adequadamente às solicitações humanas. O RHLF, mesmo não sendo uma aprendizado supervisionado puro, permite ao GPT ser efetivamente *fine-tuned*.

Outra técnica igualmente utilizada é o *natural language processing* (NLP), que é o processo de fazer um inteligência artificial a entender as regras e a sintaxe de uma linguagem.

Referências

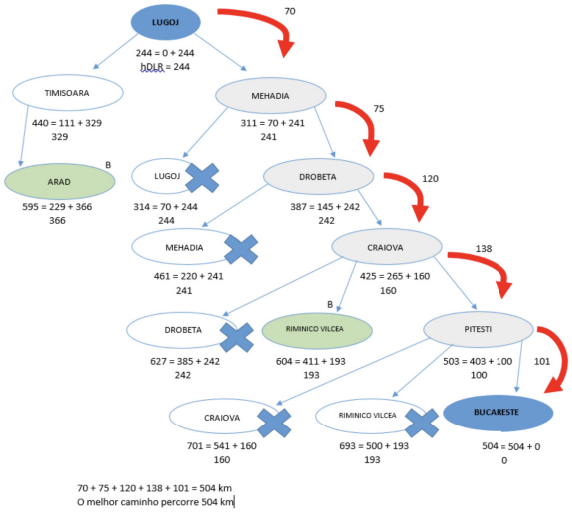
https://zapier.com/blog/how-does-chatgpt-work/

https://investnews.com.br/guias/chatgpt/

d) Pensar racionalmente, pois fornece respostas com base em padrões aprendidos a partir de uma vasta gama de dados. Ele não possui emoções, experiências pessoais ou compreensão profunda das nuances humanas. E ele não entende o que é dito e o que ele responde.

2 Busca Heurística

Figura 1.3 – RESOLUÇÃO EXERCÍCIO 2: CAMINHO DE LUGOJ ATÉ BUCHAREST



3 Lógica

$R4 : r \rightarrow p$ - Equivalência da implicação em R3
$R5 : q \rightarrow p$ - Silogismo hipotético em R2 e R4
$R6 : p \rightarrow q \wedge q \rightarrow p$ - Conjunção em R1 e R5
$R7 : q \leftrightarrow p$ - Equivalência bicondicional em R6

4 Redes Neurais Artificiais

a) $N_1 = 0,3$ função linear onde $fa(u) = u - 3 * 0,2 + 1 * 0,8 + 1 * 0,1 = 0,3$

b) $N_2 = 0,3| - 3 * 0,1 + 1 * 0,2 + 1 * 0,4 = 0,3$

c) $N_3 = -2| - 3 * 0,9 + 1 * 0,5 + 1 * 0,2 = -2$

d) $Sada = -0,139$

$0,3 * 0,9 + 0,3 * 0,3 - 2 * 0,3 + 1 * 0,1 = -0,14$

$TangH = (EXP(-0,14) - EXP(-(-0,14)))/(EXP(-0,14) + EXP(-(-0,14))) = -0,139$

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A - ENUNCIADO

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

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

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

Metadados:

Tabela 2.1: Exercício 2 de LPA: Metados da Base de Dados Carros Brasil

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

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

1 Análise Exploratória dos Dados

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

a) Carregue a base de dados `media_precos_carros_brasil.csv`;

b) Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes;

c) Verifique se há dados duplicados nos dados;

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

e) Imprima a contagem de valores por modelo (`model`) e marca do carro (`brand`);

f) Dê uma breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados.

2 Visualização dos Dados

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

a) Gere um gráfico da distribuição da quantidade de carros por marca;

b) Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro;

c) Gere um gráfico da evolução da média de preço dos carros ao longo dos meses de 2022 (variável de tempo no eixo X);

d) Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem;

e) Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item d;

f) Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de combustível;

g) Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item f.

3 Aplicação de Modelos de *Machine Learning* para Prever o Preço Médio dos Carros

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

a) Escolha as variáveis **numéricas** (modelos de Regressão) para serem as variáveis independentes do modelo. A variável *target* é `avg_price`.

Observação: caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no modelo. Indique foram transformadas e **como** foram transformadas;

b) Crie partições contendo 75% dos dados para treino e 25% para teste;

c) Treine modelos `RandomForestRegressor` e `XGBRegressor` para predição dos preços dos carros. Observação: caso julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo;

d) Grave os valores preditos em variáveis criadas;

e) Realize a análise de importância das variáveis para estimar a variável *target*, para cada modelo treinado;

f) Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis;

g) Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2;

h) Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada.

B - RESOLUÇÃO

1 Análise Exploratória dos Dados

a. Carregue a base de dados media_precos_carros_brasil.csv

```
# Importando bibliotecas necessárias
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
    r2_score

warnings.filterwarnings('ignore')

# Carregando dados do arquivo precos_carros_brasil.csv
dados = pd.read_csv('precos_carros_brasil.csv')

# Listando o nome das colunas
dados.columns

# Imprimindo somente as cinco primeiras linhas
dados.head()

# Imprime o tipo de dado de cada coluna
dados.dtypes

# Número de linhas e colunas
dados.shape
```

b. Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes

```
# Verificando se há valores faltantes nos dados
dados.isna().any()
dados.isna().sum()

# Verificando se há linhas inteiramente vazias e quantas existem
```

```
dados.isnull().all(axis=1).sum()

# Removendo todas as linhas que têm todos os campos vazios
dados.dropna(axis=0, how='all', inplace=True)

# Verificando novamente se há valores faltantes
dados.isna().sum()
```

c. Verifique se há dados duplicados nos dados

```
# Verificando se há dados duplicados
dados.duplicated().sum()

# Removendo dados duplicados
dados.drop_duplicates(inplace=True)
```

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

```
# Convertendo colunas numéricas para tipos adequados
dados['engine_size'] = dados['engine_size'].str.replace(',', '.').
    astype(float)
dados['year_of_reference'] = pd.to_numeric(dados['year_of_reference'
    ], errors='coerce').astype('Int64')
dados['year_model'] = pd.to_numeric(dados['year_model'], errors='
    coerce').astype('Int64')
dados['avg_price_brl'] = pd.to_numeric(dados['avg_price_brl'], errors
    ='coerce')

# Criando duas categorias: 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']

# Estatística descritiva das variáveis numéricas
dados[numericas_cols].describe()

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

e. Imprima a contagem de valores por modelo (model) e marca do carro (brand)

```
# Contagem de valores por modelo e marca
dados.groupby(['model', 'brand']).size().reset_index(name='contagem')
```

f) Dê uma breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados.

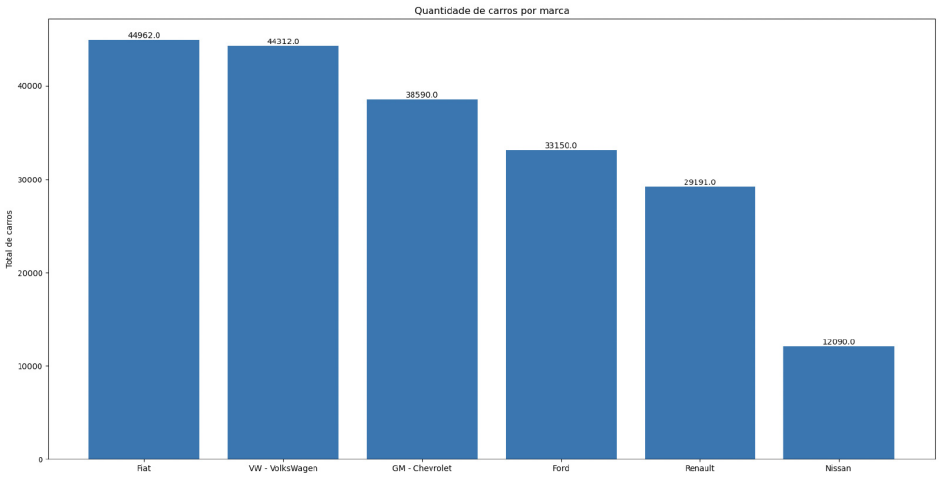
No geral, o preço médio de um carro no Brasil custa aproximadamente R$ 52.756,91. Um dos modelos mais vendidos é Palio Week. Adv/Adv TRYON 1.8 mpi Flex, com câmbio manual e motor 1.6. O menor preço médio de carros foi de R$6.647,00, e o maior preço médio é R$ 979.358,00.

2 Visualização dos Dados

a. Gere um gráfico da distribuição da quantidade de carros por marca

```
# Gráfico: distribuição da quantidade de carros por marca
plt.figure(figsize=(20,10))
grafic_brands_qties = plt.bar(dados['brand'].value_counts().index,
                              dados['brand'].value_counts().values)
plt.title('Quantidade de carros por marca')
plt.ylabel('Total de carros')
plt.bar_label(grafic_brands_qties, size=10, fmt="%.01f", label_type='
    edge')
```

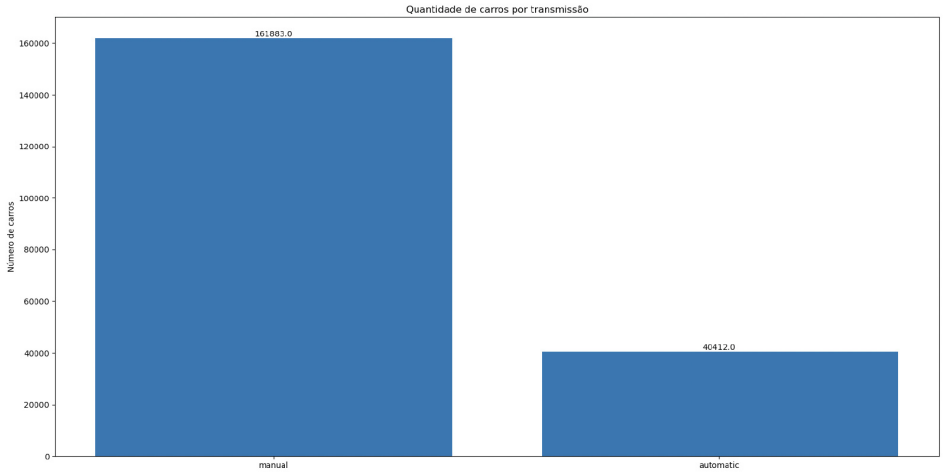
Figura 2.1 – RESOLUÇÃO EXERCÍCIO 2: DISTRIBUIÇÃO DA QUANTIDADE DE CARROS POR MARCA



b. Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro

```
# Gráfico: distribuição da quantidade de carros por tipo de
    engrenagem
plt.figure(figsize=(20,10))
grafic_gear_qties = plt.bar(dados['gear'].value_counts().index,
                            dados['gear'].value_counts().values)
plt.title('Quantidade de carros por transmissão')
plt.ylabel('Número de carros')
plt.bar_label(grafic_gear_qties, size=10, fmt="%.01f", label_type='
    edge')
```

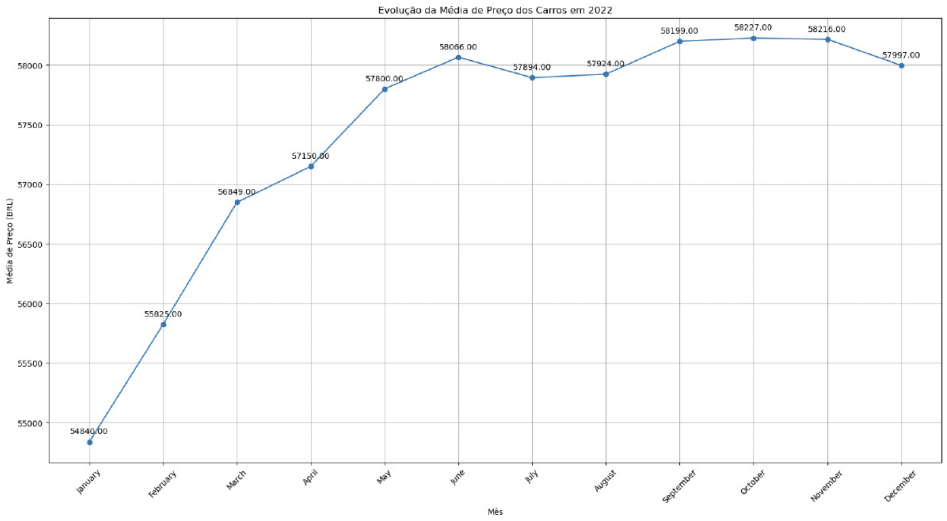
Figura 2.2 – RESOLUÇÃO EXERCÍCIO 2: DISTRIBUIÇÃO DA QUANTIDADE DE CARROS POR TIPO DE ENGRENAGEM



c. Gere um gráfico da evolução da média de preço dos carros ao longo dos meses de 2022 (variável de tempo no eixo X)

```
1  # Evolução da média de preço dos carros ao longo de 2022
2  dados['year_of_reference'] = dados['year_of_reference'].round().
       astype(int)
3  cars_in_2022 = dados[dados['year_of_reference'] == 2022]
4  month_order = ['January','February','March','April','May','June','
       July',
5                 'August','September','October','November','December']
6  cars_in_2022['month_of_reference'] = pd.Categorical(cars_in_2022['
       month_of_reference'],
7                                                      categories=
       month_order, ordered=True)
8  cars_avg_price_in_2022 = cars_in_2022.groupby(['month_of_reference'])
       ['avg_price_brl'].mean().round(0).reset_index(name='average_price'
       )
9
10 plt.figure(figsize=(20,10))
11 plt.plot(cars_avg_price_in_2022['month_of_reference'],
12          cars_avg_price_in_2022['average_price'], marker='o',
       linestyle='-')
13 for x, y in zip(cars_avg_price_in_2022['month_of_reference'],
       cars_avg_price_in_2022['average_price']):
14     plt.annotate(f"{y:.2f}", (x,y), textcoords="offset points",
       xytext=(0,10), ha='center')
15 plt.title('Evolução da Média de Preço dos Carros em 2022')
16 plt.xlabel('Mês')
17 plt.ylabel('Média de Preço (BRL)')
18 plt.xticks(cars_avg_price_in_2022.index, month_order, rotation=45)
19 plt.grid(True)
```

Figura 2.3 – RESOLUÇÃO EXERCÍCIO 2: EVOLUÇÃO DA MÉDIA DE PREÇO DOS CARROS AO LONGO DE 2022

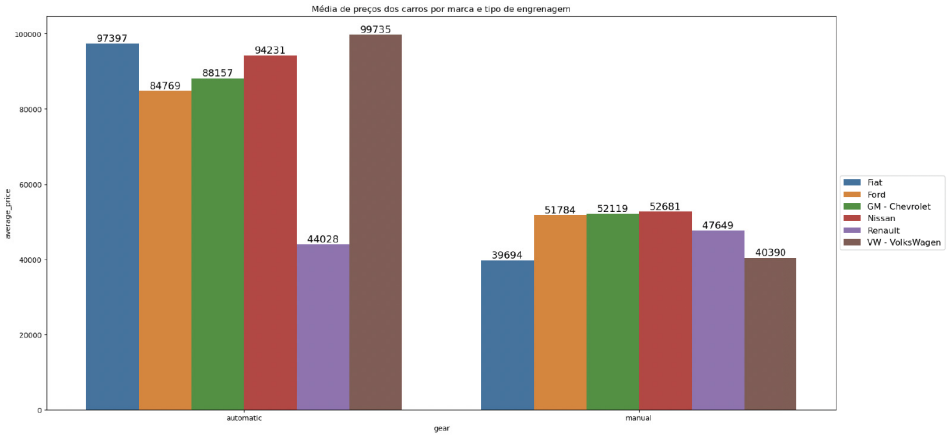


d. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem

```
# Agrupar média de preços médios dos carros por marcar e tipo de engrenagem
car_gear_brand_avg_price = dados.groupby(['brand', 'gear'])['avg_price_brl'].mean().round(0)
car_gear_brand_avg_price.head(12)

# Utilizando a função reset_index para criar uma ordem e facilitar a criação do gráfico
car_gear_brand_avg_price = car_gear_brand_avg_price.reset_index(name='average_price')
car_gear_brand_avg_price.head(12)

# Gerar o gráfico de distruibuição média dos preços dos carros por marca e tipo de engrenagem
plt.figure(figsize=(20,10))
barplot = sns.barplot(
x='gear',
y='average_price',
hue='brand',
data=car_gear_brand_avg_price,
hue_order=car_gear_brand_avg_price['brand'].unique(),
)

barplot.legend(loc='center left', bbox_to_anchor=(1, 0.5), ncol=1, fontsize=12)
barplot.bar_label(barplot.containers[0], fontsize=14);
barplot.bar_label(barplot.containers[1], fontsize=14);
barplot.bar_label(barplot.containers[2], fontsize=14);
barplot.bar_label(barplot.containers[3], fontsize=14);
barplot.bar_label(barplot.containers[4], fontsize=14);
barplot.bar_label(barplot.containers[5], fontsize=14);
```

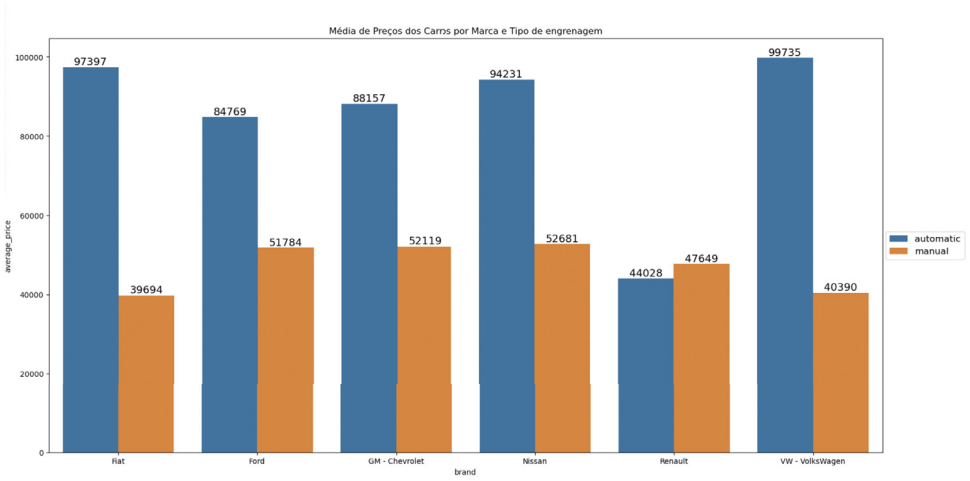
```
26 plt.title('Média de preços dos carros por marca e tipo de engrenagem'
       )
```

Figura 2.4 – RESOLUÇÃO EXERCÍCIO 2: MÉDIA DE PREÇOS DOS CARROS POR MARCA E TIPO DE ENGRENAGEM AGRUPADOS POR TIPO




```
1  # Gerar o segundo tipo de gráfico de distruibuição média dos preços
       dos carros por marca e tipo de engrenagem
2  plt.figure(figsize=(20,10))
3  barplot = sns.barplot(
4  x='brand',
5  y='average_price',
6  hue='gear',
7  data=car_gear_brand_avg_price,
8  hue_order=car_gear_brand_avg_price['gear'].unique(),)
9  barplot.legend(loc='center left', bbox_to_anchor=(1, 0.5), ncol=1,
       fontsize=12)
10 barplot.bar_label(barplot.containers[0], fontsize=14);
11 barplot.bar_label(barplot.containers[1], fontsize=14);
12 plt.title('Média de Preços dos Carros por Marca e Tipo de engrenagem'
       )
```

Figura 2.5 – RESOLUÇÃO EXERCÍCIO 2: MÉDIA DE PREÇOS DOS CARROS POR MARCA E TIPO DE ENGRENAGEM AGRUPADOS POR MARCA



e. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item d:

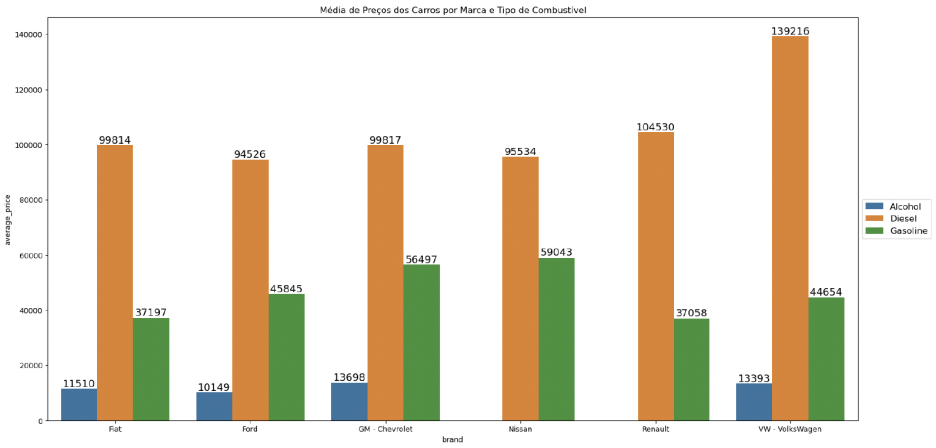
O preço médio do carro com câmbio automático é relativamente mais alto quando comparado aos de câmbio manual para todas as marcas, com exceção da Renault. Volkswagen e Fiat lideram as marcas que possuem maior preço médio para carros de câmbio automático, ao passo que lideram também as marcas com menor preço médio dos carros com câmbio manual.

f. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de combustível

```
# Agrupar média de preços médios dos carros por marcar e tipo de combustível
cars_avg_price_brand_fuel = dados.groupby(['brand', 'fuel'])['avg_price_brl'].mean().round(0)
cars_avg_price_brand_fuel.head(16)

# Utilizando a função reset_index para criar uma ordem e facilitar a criação do gráfico
cars_avg_price_brand_fuel = cars_avg_price_brand_fuel.reset_index(name='average_price')
cars_avg_price_brand_fuel.head(16)

# Gerar gráfico de média de preços médios de carros por marcar e tipo de combustível
plt.figure(figsize=(20, 10))
barplot = sns.barplot(
x='brand',
y='average_price',
hue='fuel',
data=cars_avg_price_brand_fuel,
hue_order=cars_avg_price_brand_fuel['fuel'].unique()
)
barplot.legend(loc='center left', bbox_to_anchor=(1, 0.5), ncol=1, fontsize=12)
barplot.bar_label(barplot.containers[0], fontsize=14)
barplot.bar_label(barplot.containers[1], fontsize=14)
barplot.bar_label(barplot.containers[2], fontsize=14)
```

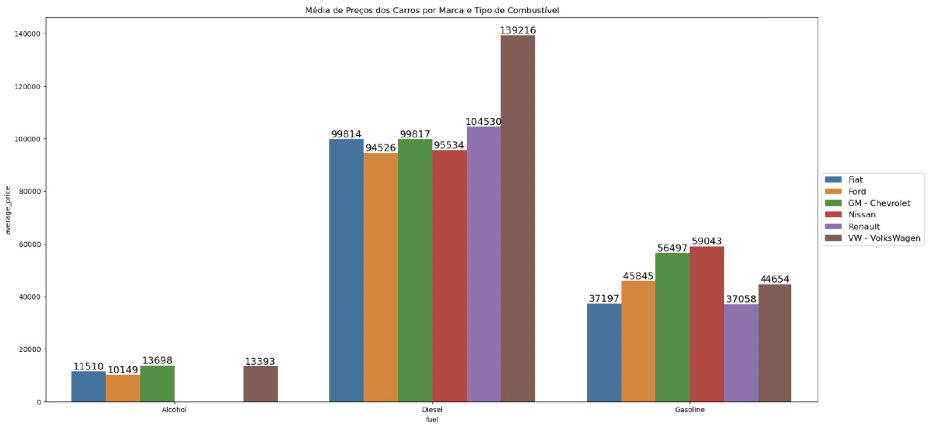
```
22 plt.title('Média de Preços dos Carros por Marca e Tipo de Combustível
       ')
```

Figura 2.6 – RESOLUÇÃO EXERCÍCIO 2: MÉDIA DE PREÇOS DOS CARROS POR MARCA E TIPO DE COMBUSTÍVEL AGRUPADOS POR MARCA




```
1  # Gerar gráfico de média de preços médios de carros por marcar e tipo
       de combustível - Segundo tipo
2  plt.figure(figsize=(20, 10))
3  barplot = sns.barplot(
4  x='fuel',
5  y='average_price',
6  hue='brand',
7  data=cars_avg_price_brand_fuel,
8  hue_order=cars_avg_price_brand_fuel['brand'].unique())
9
10 barplot.legend(loc='center left', bbox_to_anchor=(1, 0.5), ncol=1,
       fontsize=12)
11 barplot.bar_label(barplot.containers[0], fontsize=14)
12 barplot.bar_label(barplot.containers[1], fontsize=14)
13 barplot.bar_label(barplot.containers[2], fontsize=14)
14 barplot.bar_label(barplot.containers[3], fontsize=14)
15 barplot.bar_label(barplot.containers[4], fontsize=14)
16 barplot.bar_label(barplot.containers[5], fontsize=14)
17 plt.title('Média de Preços dos Carros por Marca e Tipo de Combustível
       ')
```

Figura 2.7 – RESOLUÇÃO EXERCÍCIO 2: MÉDIA DE PREÇOS DOS CARROS POR MARCA E TIPO DE COMBUSTÍVEL AGRUPADOS POR TIPO



g. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item f:

1. O preço médio do carro à diesel é muito mais elevado para todas as marcas, sendo a Volkswagen a marca com maior preço médio para esta categoria.
2. O preço médio do carro à álcool lidera com os menores valores.
3. Os carros à gasolina com menor preço médio são das marcas Fiat e Renault.

3 Aplicação de Modelos de Machine Learning para Prever o Preço Médio dos Carros

a) Escolha as variáveis numéricas (modelos de Regressão) para serem as variáveis independentes do modelo. A variável target é avg_price.

```
# Gráfico de distribuição boxplot dos preços médios dos carros
sns.boxplot(dados['avg_price_brl']).set_title('Distribuição dos preços médios')

# Transformar mês de referência em uma variável numérica
dados['month_of_reference'] = LabelEncoder().fit_transform(dados['month_of_reference'])

# Transformar marca de carro em uma variável numérica
dados['brand'] = LabelEncoder().fit_transform(dados['brand'])

# Transformar modelo de carro em uma variável numérica
dados['model'] = LabelEncoder().fit_transform(dados['model'])

# Transformar tipo de combustível em uma variável numérica
dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])

# Transformar tipo de transmissão em uma variável numérica
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])

# Remover as variáveis de entrada que não são consideradas importantes para a predição
dados_interesting = dados.drop([
```

```
21 'fipe_code',
22 'authentication',
23 ], axis=1)
24
25 # Mapa de correlação das variáveis numéricas com variável Target
26 sns.heatmap(dados_interesting.corr("spearman"), annot = True)
27 plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize =
       18)
28 plt.show()
29
30 # Variável X contém apenas variáveis numéricas de interesse para a an
       álise excluindo a variável target
31 X = dados_interesting.drop(['avg_price_brl'], axis=1)
32 X.head()
33
34 # Variável Y contém apenas a variável target - avg_price_brl
35 Y = dados_interesting['avg_price_brl']
36 Y.head()
```

Figura 2.8 – RESOLUÇÃO EXERCÍCIO 3: BOXPLOT DA DISTRIBUIÇÃO DOS PREÇOS MÉDIOS

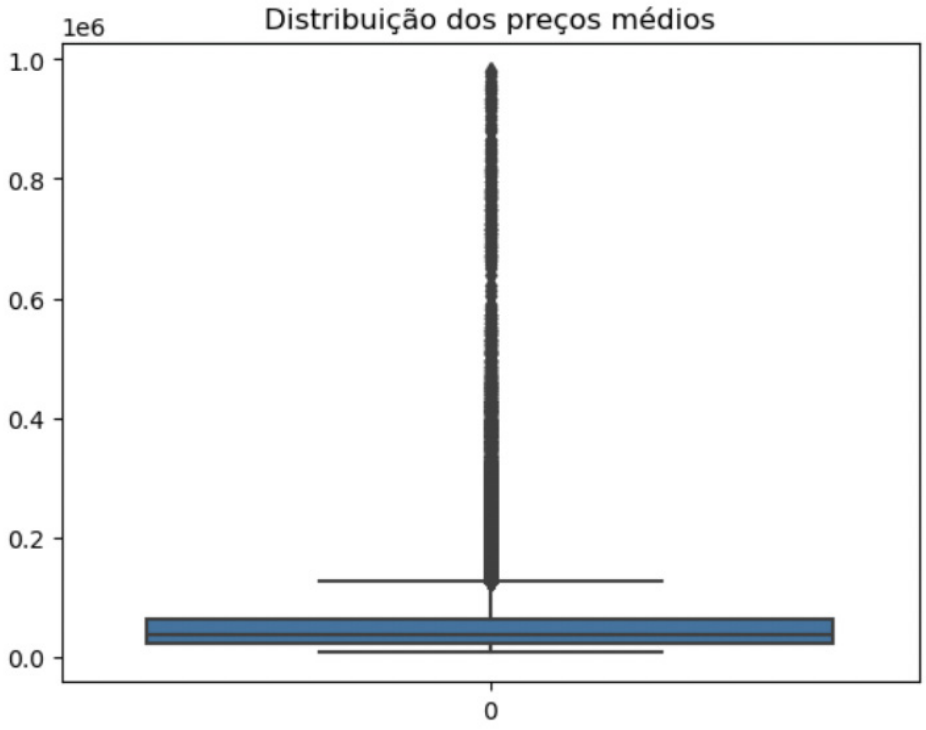
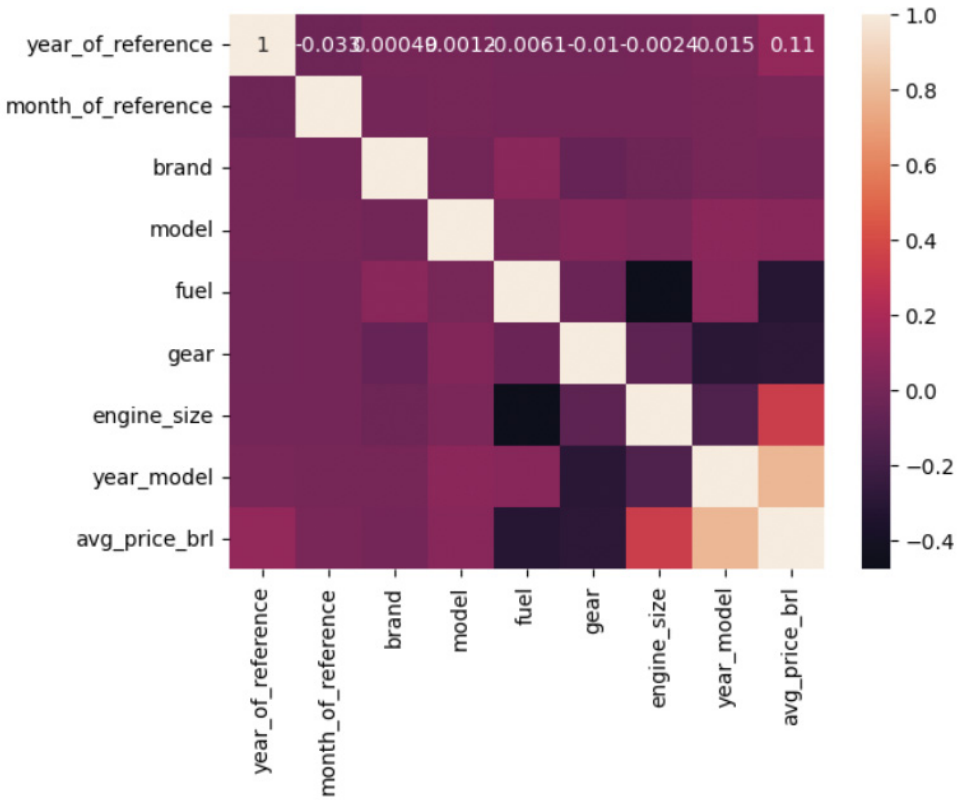


Figura 2.9 – RESOLUÇÃO EXERCÍCIO 3: MAPA DE CORRELAÇÃO ENTRE AS VARIÁVEIS NUMÉRICAS



b. Crie partições contendo 75% dos dados para treino e 25% para teste

```
# 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)
```

c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros

```
#Algoritmo Random Forest, sem especificar nenhum parâmetro (número de
    árvores, número de ramificações, etc)
model_rf = RandomForestRegressor()

# Random Forest com os hiperparametros:
model_rf_paramenters = RandomForestRegressor(
max_depth=29,
min_samples_leaf=32,
min_samples_split=28,
n_estimators=208,
random_state=43
)
# Ajuste do modelo, de acordo com as variáveis de treinamento
model_rf_paramenters.fit(X_train, Y_train)

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

```

```
# Algoritmo XGBoost sem parametros
model_xg = XGBRegressor()

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

#Algoritmo XGBoost com parametros
model_parameters_xg = XGBRegressor(
    learning_rate=1,
    n_estimators=100,
    max_depth=25,
    min_child_weight=1,
    gamma=0,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0,
    reg_lambda=1,
    random_state=42,
    n_jobs=-1
)

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

d. Grave os valores preditos em variáveis criadas

```
# Predição dos valores de preço médio dos carros com base nos dados de teste Random Forest sem parametros
predicted_values_rf = model_rf.predict(x_test)

# Predição dos valores de preço médio dos carros com base nos dados de teste Random Forest com parametros
predicted_values_parameters_rf = model_rf_paramenters.predict(x_test)

# Predição dos valores de preços médios dos carros com base nos dados de teste XGBoost sem parametros
predicted_values_xg = model_xg.predict(x_test)

# Predição dos valores de preços médios dos carros com base nos dados de teste XGBoost com parametros
predicted_values_parameters_xg = model_parameters_xg.predict(x_test)
```

e. Realize a análise de importância das variáveis para estimar a variável target, para cada modelo treinado

```
# Análise da importancia das variáveis para estimar a variável target - Random Forest sem parametros
model_rf.feature_importances_
feature_importances_rf = pd.DataFrame(
model_rf.feature_importances_,
index = X_train.columns,
columns = ['importances']
).sort_values('importances', ascending=True)
```

month_of_reference 0.005168
year_of_reference 0.012553
brand 0.016695
fuel 0.032713
gear 0.035363
model 0.058640
year_model 0.388397
engine_size 0.450472

```
# Análise da importancia das variáveis para estimar a variável target - Random Forest com parametros
model_rf_paramenters.feature_importances_
feature_importances_parameters_rf = pd.DataFrame(
model_rf_paramenters.feature_importances_,
index = X_train.columns,
columns = ['importances']
).sort_values('importances', ascending=True)
feature_importances_parameters_rf
```

month_of_reference 0.000690
year_of_reference 0.010383
brand 0.016098
gear 0.021959
fuel 0.034132
model 0.037028
year_model 0.410473
engine_size 0.469236

```
# Análise da importancia das variáveis para estimar a variável target - XGBoost com parametros
model_parameters_xg.feature_importances_
feature_importances_parameters_xg = pd.DataFrame(
model_rf_paramenters.feature_importances_,
index = X_train.columns,
columns = ['importances']
).sort_values('importances', ascending=True)
feature_importances_parameters_xg
```

month_of_reference 0.000690
year_of_reference 0.010383
brand 0.016098
gear 0.021959
fuel 0.034132
model 0.037028
year_model 0.410473
engine_size 0.469236

```
# Analisando a importância das variáveis para estimar a variável target - XGBoost sem parametros
model_xg.feature_importances_
feature_importances_xg = pd.DataFrame(
model_xg.feature_importances_,
```

```
index=X_train.columns,
columns=['importances']
).sort_values('importances', ascending=True)
feature_importances_xg
```

month_of_reference 0.004271
year_of_reference 0.017475
model 0.023331
brand 0.056276
gear 0.122657
fuel 0.154437
year_model 0.190992
engine_size 0.430561

f. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis:

As variáveis que tiveram maior relevância na medida da performance dos modelos treinados foram o tamanho do motor (engine_size) e ano do modelo (year_model) respectivamente. O mês de referência foi o que obteve menor relevância em todos o modelos treinados.

g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e *R*2:

```
# MSE - calcula o erro quadrático médio das predições do modelo
# Random Forest sem parametro
mse_rf = mean_squared_error(y_test, predicted_values_rf) #12181571.08

# O MAE calcula a média da diferença absoluta entre o valor predito e o valor real
# Random Forest sem parametro
mea_rf = mean_absolute_error(y_test, predicted_values_rf) # 1765.88

# O R2 é uma métrica que varia entre 0 e 1 e é uma razão que indica o quão bom o nosso modelo
# Random Forest sem parametros
r2_score(y_test, predicted_values_rf) # 0.995
```

```
# O MSE - Random Forest com parametros
mse_parameters_rf = mean_squared_error(y_test, predicted_values_parameters_rf) # 151743671.60

# O MAE - Random Forest com parametros
mae_parameters_rf = mean_absolute_error(y_test, predicted_values_parameters_rf) # 4577.49

# R2 score - Random Forest com parametros # 0.943
r2_score(y_test, predicted_values_parameters_rf)
```

```
# MSE - XGBoost sem parametros
mse_xg = mean_squared_error(y_test, predicted_values_xg) # 30952404.229

# MAE - XGBoost sem parametros
mae_xg = mean_absolute_error(y_test, predicted_values_xg) # 3245.31
```

```
6
7 # R2 score - XGBoost sem parametros
8 r2_score(y_test, predicted_values_xg) # 0.988
```

```
1 # MSE - XGBoost com parametros
2 mse_parameters_xg = mean_squared_error(y_test,
     predicted_values_parameters_xg) # 58122007.14
3
4 # MAE - XGBoost com parametros
5 mae_parameters_xg = mean_absolute_error(y_test,
     predicted_values_parameters_xg) # 3988.81
6
7 # R2 score - XGBoost com parametros
8 r2_score(y_test, predicted_values_parameters_xg) # 0.978
```

O melhor modelo com base nessas medidas de acurácia foi o modelo RandomForest sem parâmetros.

h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada:

O modelo que apresentou os melhores resultado foi o RandomForest sem parametros para medir a sua acurácia foi utilizado as médidas de acurácia MSE (*mean squared error*) com o valor 12165425.33231638, MAE (*mean absolute error*) com o valor 12165425.33 e R2_score com 0.995 de precisão.

APÊNDICE 3 – LINGUAGEM R

A - ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

O processo é feito pela coleta de dados (dados observados) através do abate de algumas árvores, onde sua altura, diâmetro na altura do peito (dap), etc, são medidos de forma exata.

Com estes dados, treina-se um modelo de AM que pode estimar o volume de outras árvores da população.

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

$$Volume = b0 + b1 * dap2 * Ht \tag{3.1}$$

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

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

Tarefas

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

$$Volume = b0 + b1 * dap2 * Ht \tag{3.2}$$

```
1    alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, dados, start=list(b0
    =0.5, b1=0.5))
2
```

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

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.3}$$

onde *yi* é o valor observado, *yi* é o valor predito e *y* é a média dos valores *yi* observados. Quanto mais perto de 1 melhor é o modelo;

- Erro padrão da estimativa: S_{yx}

$$S_{yx} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n-2}} \tag{3.4}$$

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

- Syx%:

$$S_{yx}\% = \frac{S_{yx}}{\bar{y}} \times 100 \tag{3.5}$$

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

7. Escolha o melhor modelo.

B - RESOLUÇÃO

1 Pesquisa com Dados de Satélite (Satellite)

```
#### 1.1 Carregue a base de dados Satellite

#instalar o pacotes necessários
#install.packages("mlbench", repos = "http://cran.us.r-project.org")
#install.packages("e1017", repos = "http://cran.us.r-project.org")
#install.packages("randomForest", repos = "http://cran.us.r-project.org")
#install.packages("kernlab", repos = "http://cran.us.r-project.org")
#install.packages("caret", repos = "http://cran.us.r-project.org")

# usar os pacotes necessários
library(mlbench)
library(caret)

# carregar os dados Satellite
data(Satellite)

# exibir estrutura dos dados Satellite
str(Satellite)

# apresentar alguma medidas estatísticas do dados Satellite
summary(Satellite)

# exibir alguns dados do Satellite
head(Satellite, n = 6)

#### 1.2 Crie partições contendo 80% para treino e 20% para teste

# Para reproductibilidade
set.seed(7)

# particionar em 80% para treino e 20% para teste
indices <- createDataPartition(Satellite[["classes"]], p=0.8, list=F)
treino <- Satellite[indices, ]
teste <- Satellite[-indices, ]

#### 1.3 Treine modelos RandomForest, SVM e RNA para predição destes dados.

# treinar modelos RandomForest, SVM e RNA
```

```
rf <- train(classes ~ ., data=treino, method="rf")
svm <- train(classes ~ ., data=treino, method="svmRadial")
rna <- train(classes ~ ., data=treino, method="nnet", trace=F)

#### 1.4 Escolha o melhor modelo com base em suas matrizes de confusão.

# predições
predict.rf <- predict(rf, teste)
predict.svm <- predict(svm, teste)
predict.rna <- predict(rna, teste)

# matrizes de confusões de cada uma das predições

# matriz de confusão para o modelo RF
conf_matrix.rf <- confusionMatrix(predict.rf, teste[["classes"]])

print(conf_matrix.rf)
cat('\n')

# matriz de confusão para o modelo SVM
conf_matrix.svm <- confusionMatrix(predict.svm, teste[["classes"]])

print(conf_matrix.svm)
cat('\n')

# matriz de confusão para o modelo RNA
conf_matrix.rna <- confusionMatrix(predict.rna, teste[["classes"]])

print(conf_matrix.rna)
cat('\n')

#### 1.5 Indique qual modelo dá o melhor o resultado e a métrica utilizada

#O melhor modelo foi random forest com acurácia de 0.9182 e kappa de 0.8987. A métrica utilizada foram a acurácia e kappa.
```

2 Estimativa de Volume de Árvores

```
#### 2.1 Carregar o arquivo Volumes.csv (<http://www.razer.net.br/datasets/Volumes.csv>)

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

#### 2.2 Eliminar a coluna NR, que só apresenta um número sequencial

dados[["NR"]] <- NULL

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

```
11
12 regression.indices <- caret::createDataPartition(dados[["VOL"]], p
      =0.8, list=F)
13 regression.treino <- dados[regression.indices, ]
14 regression.teste <- dados[-regression.indices, ]
15
16 #### 2.4 Usando o pacote "caret", treinar os modelos: Random Forest (
      rf), SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alomé
      trico de SPURR
17
18 # Para reproductibilidade
19 set.seed(7)
20
21 regression.rf <- caret::train(VOL ~ ., data=regression.treino, method
      ="rf")
22 regression.svm <- caret::train(VOL ~ ., data=regression.treino,
      method="svmRadial")
23 regression.rna <- caret::train(VOL ~ ., data=regression.treino,
      method="nnet", trControl=trainControl(method = "LOOCV"), trace=F)
24
25 ##### treino do modelo Spurr
26
27 regression.spurr <- nls( VOL ~ b0 + b1*DAP*DAP*HT, data=regression.
      treino, start=list(b0=0.5, b1=0.5))
28
29 ##### visualizar os resultados de Spurr
30
31 summary(regression.spurr)
32
33 #### 2.5 Efetue as predições nos dados de teste
34
35 # predições
36
37 # Para reproductibilidade
38 set.seed(7)
39
40 predict.regression.rf <- predict(regression.rf, regression.teste)
41 predict.regression.svm <- predict(regression.svm, regression.teste)
42 predict.regression.rna <- predict(regression.rna, regression.teste)
43 predict.regression.suprr <- predict(regression.spurr, regression.
      teste)
44
45 #### 2.6 Crie suas próprias funções (UDF) e calcule as seguintes mé
      tricas entre a predição e os dados observados
46
47 # -   Erro padrão de estimativa: Syx
48
49 Syx <- function(reals, predicteds, n) {
50   return (sqrt(sum((reals - predicteds)^2)/(n - 2)))
51 }
52
```

```
# -   Erro padrão de estimativa em porcentagem: Syx%

SyxPercent <- function(reals, predicteds, n) {
  return ((Syx(reals, predicteds, n)/mean(reals))*100)
}

# -   Coeficientededeterminação:R2

R2 <- function (reals, predicteds) {
  return (1 - sum((reals - predicteds)^2)/sum((reals - mean(reals))
   ^2))
}

#### 2.7 Escolha o melhor modelo.

##### métrica de estimativas para o modelo RandomForest - Regressão

# -   coeficiente de determinação

R2(regression.teste[["VOL"]], predict.regression.rf)

# -   Erro padrão estimativa

n <- nrow(regression.teste)
Syx(regression.teste[["VOL"]], predict.regression.rf, n)

# -   Erro padrão estimativa em porcentagem

n <- nrow(regression.teste)
SyxPercent(regression.teste[["VOL"]], predict.regression.rf, n)

##### métrica de estimativas para o modelo SVM - Regressão

# -   coeficiente de determinação

R2(regression.teste[["VOL"]], predict.regression.svm)

# -   Erro padrão estimativa

n <- nrow(regression.teste)
Syx(regression.teste[["VOL"]], predict.regression.svm, n)

# -   Erro padrão estimativa em porcentagem

n <- nrow(regression.teste)
SyxPercent(regression.teste[["VOL"]], predict.regression.svm, n)

##### métricas de estimativas para o modelo nnet - Regressão

# -   coeficiente de determinação

```

```
R2(regression.teste[["VOL"]], predict.regression.rna)

# -   Erro padrão estimativa

n <- nrow(regression.teste)
Syx(regression.teste[["VOL"]], predict.regression.rna, n)

# -   Erro padrão estimativa em porcentagem

n <- nrow(regression.teste)
SyxPercent(regression.teste[["VOL"]], predict.regression.rna, n)

##### métricas de estimativas para o modelo Spurr

# -   coeficiente de determinação

R2(regression.teste[["VOL"]], predict.regression.suprr)

# -   Erro padrão estimativa

n <- nrow(regression.teste)
Syx(regression.teste[["VOL"]], predict.regression.suprr, n)

# -   Erro padrão estimativa em porcentagem

n <- nrow(regression.teste)
SyxPercent(regression.teste[["VOL"]], predict.regression.suprr, n)

##### 2.7 escolha o melhor modelo

##### Resumo dos resultados

###### RF:

# 1.  coeficiente de determinação: 0.8223603.
# 2.  Erro padrão estimativa: 0.1376052.
# 3.  Erro padrão estimativa em porcentagem: 10.42195

###### SVM:

# 1.  coeficiente de determinação: 0.6254546
# 2.  Erro padrão estimativa: 0.19981
# 3.  Erro padrão estimativa em porcentagem: 15.13322

###### nnet:

# 1.  coeficiente de determinação: -1.069672
# 2.  Erro padrão estimativa: 0.4696948
# 3.  Error padrão estimativa em porcentagem: 35.57377

###### spurr:
```

```
154
155 # 1.  coeficiente de determinação: 0.7734018
156 # 2.  Erro padrão estimativa: 0.1554151
157 # 3.  Error padrão estimativa em porcentagem: 11.77084
158
159 # Com base nas métricas, o modelo que se saiu melhor foi o
      RandomForest, com R2 igual 0.8223603, Erro padrão estimativa de
      0.1376052 e Erro padrão de estima- tiva em porcentagem de 10.42195
```

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A - ENUNCIADO

1. Gráficos e tabelas

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

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

2. Medidas de posição e dispersão

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

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

3. Testes paramétricos ou não paramétricos

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

Obs: Você deve fazer os testes necessários (e mostra-los no documento pdf) para saber se você deve usar o unpaired test (paramétrico) ou o teste U de Mann-Whitney (não paramétrico), justifique sua resposta sobre a escolha. Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

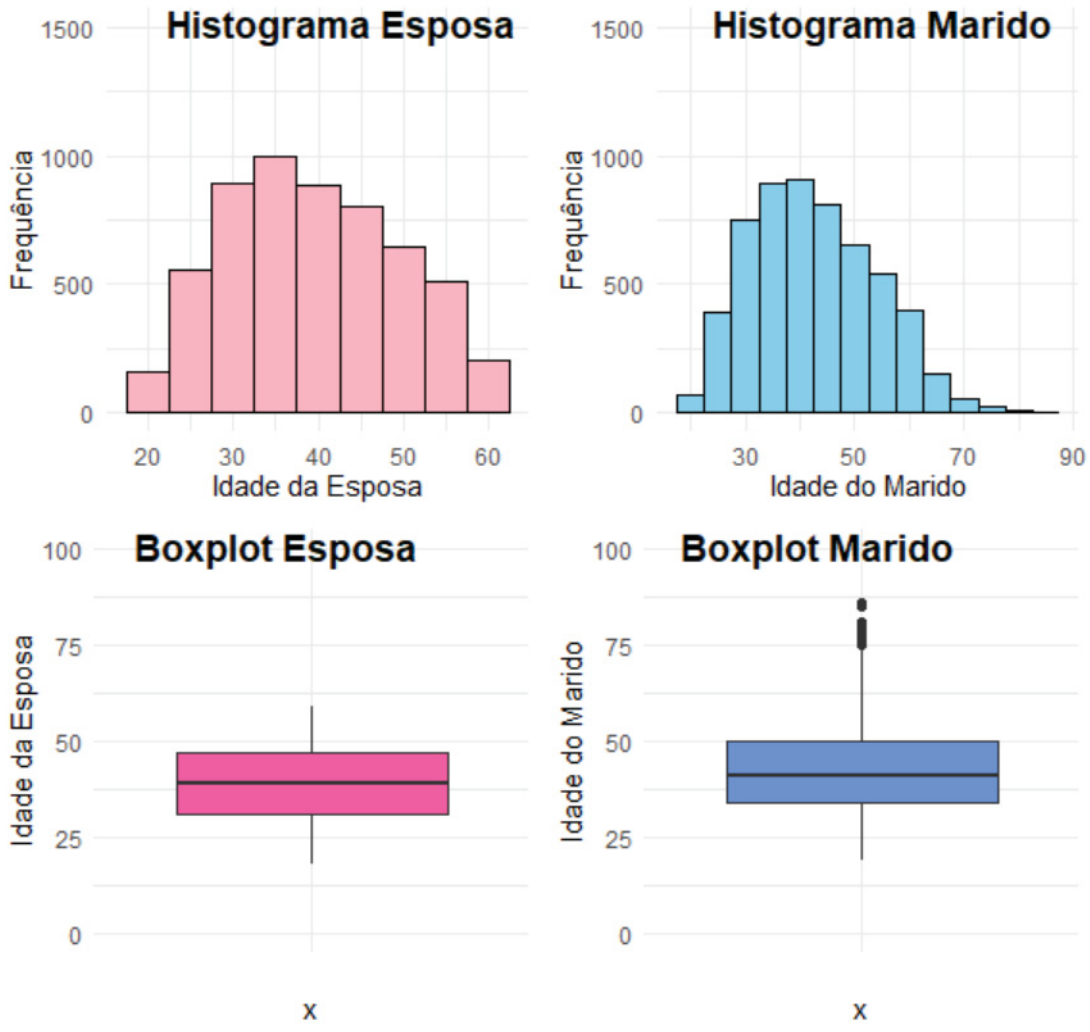
B - RESOLUÇÃO

1. Gráficos e tabelas

```
1  # Criando o histograma da idade do marido
2  histograma_marido <- ggplot(salarios, aes(x = husage)) +
3    geom_histogram(binwidth = 5, bins = 9, fill = "#87CEEB", color = "
       black") +
4    labs(x = "Idade do Marido", y = "Frequência") +
5    theme_minimal() +
6    coord_cartesian(ylim = c(0, 1500))
7
8  # Criando o boxplot da idade do marido
9  boxplot_marido <- ggplot(salarios, aes(x = "", y = husage)) +
10   geom_boxplot(fill = "#6495ED") +
11   labs(y = "Idade do Marido") +
12   theme_minimal() +
13   coord_cartesian(ylim = c(0, 100))
14
15 # Criando o histograma da idade da esposa
16 histograma_esposa <- ggplot(salarios, aes(x = age)) +
```

```
  geom_histogram(binwidth = 5, bins = 9, fill = "#FFB6C1", color = "
   black") +
  labs(x = "Idade da Esposa", y = "Frequência") +
  theme_minimal() +
  coord_cartesian(ylim = c(0, 1500))

# Criando o boxplot da idade da esposa
boxplot_esposa <- ggplot(salarios, aes(x = "", y = age)) +
  geom_boxplot(fill = "#F25EA2") +
  labs(y = "Idade da Esposa") +
  theme_minimal() +
  coord_cartesian(ylim = c(0, 100))

# Combinando os gráficos em uma única visualização
plot_comparativo <- plot_grid(histograma_esposa, histograma_marido,
   boxplot_esposa, boxplot_marido, labels = c("Histograma Esposa", "
   Histograma Marido", "Boxplot Esposa", "Boxplot Marido"), ncol = 2)
```

Figura 4.1 – EXERCÍCIO 1: HISTOGRAMA E BOX-PLOT



A análise mostra que os maridos são, em média, 3 anos mais velhos que as esposas (42,46 anos contra 39,43 anos). A variação de idade é maior entre os maridos, com idades entre 19 e 86 anos, enquanto as esposas têm idades entre 18 e 59 anos.

```
# Tabela de frequência das idades dos maridos
tabela.freq.M <- fdt(salarios$husage)
```

Tabela 4.1: Tabela de frequência das idades (maridos)

Class limits	f	rf	rf(%)	cf	cf(%)
[18.81, 23.671)	102	0.02	1.81	102	1.81
[23.671, 28.531)	466	0.08	8.27	568	10.08
[28.531, 33.392)	809	0.14	14.36	1377	24.44
[33.392, 38.253)	895	0.16	15.89	2272	40.33
[38.253, 43.114)	917	0.16	16.28	3189	56.60
[43.114, 47.974)	629	0.11	11.16	3818	67.77
[47.974, 52.835)	649	0.12	11.52	4467	79.29
[52.835, 57.696)	541	0.10	9.60	5008	88.89
[57.696, 62.556)	394	0.07	6.99	5402	95.88
[62.556, 67.417)	152	0.03	2.70	5554	98.58
[67.417, 72.278)	51	0.01	0.91	5605	99.49
[72.278, 77.139)	21	0.00	0.37	5626	99.86
[77.139, 81.999)	6	0.00	0.11	5632	99.96
[81.999, 86.86)	2	0.00	0.04	5634	100.00

```
# Tabela de frequência das idades das esposas
tabela.freq.E <- fdt(salarios$age)
```

Tabela 4.2: Tabela de frequência das idades (esposas)

Class limits	f	rf	rf(%)	cf	cf(%)
[17.82, 20.804)	61	0.01	1.08	61	1.08
[20.804, 23.787)	161	0.03	2.86	222	3.94
[23.787, 26.771)	312	0.06	5.54	534	9.48
[26.771, 29.754)	505	0.09	8.96	1039	18.44
[29.754, 32.738)	562	0.10	9.98	1601	28.42
[32.738, 35.721)	571	0.10	10.13	2172	38.55
[35.721, 38.705)	624	0.11	11.08	2796	49.63
[38.705, 41.689)	510	0.09	9.05	3306	58.68
[41.689, 44.672)	542	0.10	9.62	3848	68.30
[44.672, 47.656)	432	0.08	7.67	4280	75.97
[47.656, 50.639)	389	0.07	6.90	4669	82.87
[50.639, 53.623)	358	0.06	6.35	5027	89.23
[53.623, 56.606)	304	0.05	5.40	5331	94.62
[56.606, 59.59)	303	0.05	5.38	5634	100.00

Aproximadamente metade das esposas tem entre 29 e 41 anos, enquanto quase metade dos maridos está entre 28 e 43 anos.

2. Medidas de posição e dispersão

```
# Média da idade dos maridos = 42.45
media.marido <- mean(salarios$husage)

# Média da idade das esposas = 39.43
media.esposa <- mean(salarios$age)

# Mediana da idade dos maridos = 41
mediana.marido <- median(salarios$husage)

# Mediana da idade das esposas = 39
mediana.esposa <- median(salarios$age)

# Moda da idade dos maridos = a idade 44 aparece 201 vezes
moda.marido <- subset(table(salarios$husage), table(salarios$husage)
   == max(table(salarios$husage)))
moda.marido.calc <- names(moda.marido)
moda.marido.calc <- as.numeric(moda.marido.calc)

# Moda da idade das esposas = a idade 37 aparece 217 vezes
moda.esposa <- subset(table(salarios$age), table(salarios$age) == max
   (table(salarios$age)))
moda.esposa.calc <- names(moda.esposa)
moda.esposa.calc <- as.numeric(moda.esposa.calc)
```

Os maridos são 7,68% mais velhos, em média, que as esposas. 5,13%, considerando a mediana da amostra, e, a idade mais comum entre eles (moda) é 18,92% maior que a delas.

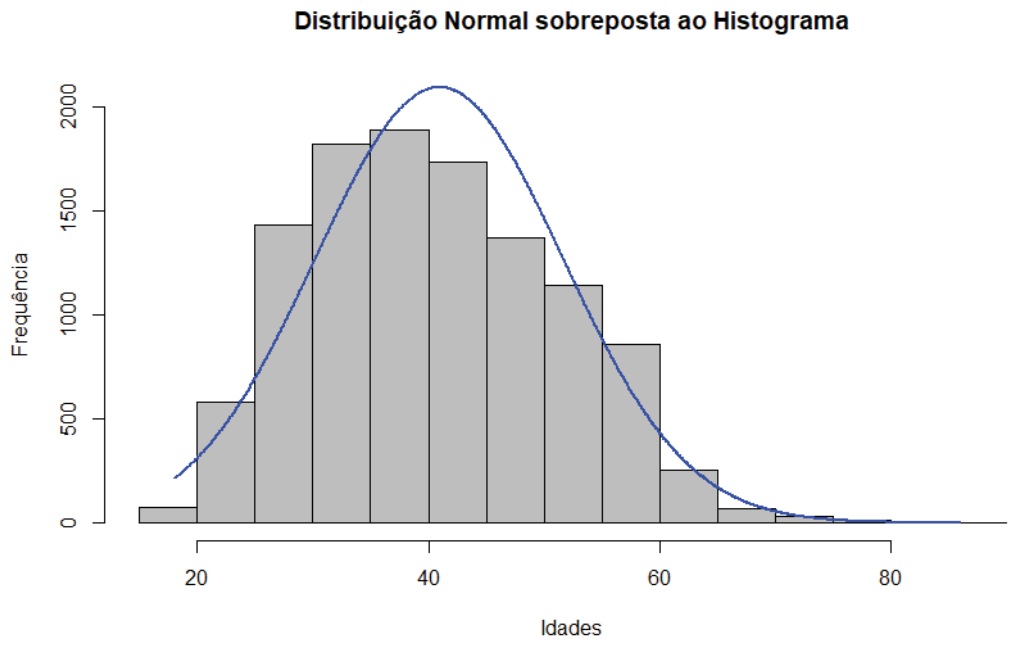
```
# Variância da idade dos maridos = 126,07
var(salarios$husage)

# Variância da idade das esposas = 99,75
var(salarios$age)

# Desvio padrão da idade dos maridos = 11,23
sd(salarios$husage)

# Desvio padrão da idade das esposas = 9,99
sd(salarios$age)

# Coeficiente de variação da idade dos maridos = 26,45%_
(sd(salarios$husage)/mean(salarios$husage))*100

# Coeficiente de variação da idade das esposas = 25,33%_
(sd(salarios$age)/mean(salarios$age))*100
```

Em resumo, a variação de idade é um pouco maior entre os maridos, mas no geral, as idades tanto das esposas quanto dos maridos não variam muito dentro da amostra.

3. Testes paramétricos ou não paramétricos

```
# Agrupando os dados
idades.df <- data.frame(
  group = rep(c("Esposas", "Maridos"), each = 5634),
  idades = c(salarios$age, salarios$husage)
)

# Media e distancia interquartílica
group_by(idades.df, group) %>%
  summarise(
    count = n(),
    mean =  mean(idades, na.rm = TRUE),
    sd = sd(idades, na.rm = TRUE)
  )

# Média e mediana do grupo
mean(idades.df$idades)
median(idades.df$idades)

# Desvio padrao
sd(idades.df$idades)

# Histograma com curva normal
plotNormalHistogram(idades.df$idades, prob = FALSE, main = "Distribui
   ção Normal sobreposta ao Histograma", length = 1000, xlab="Idades"
   , ylab = "Frequência" )
```

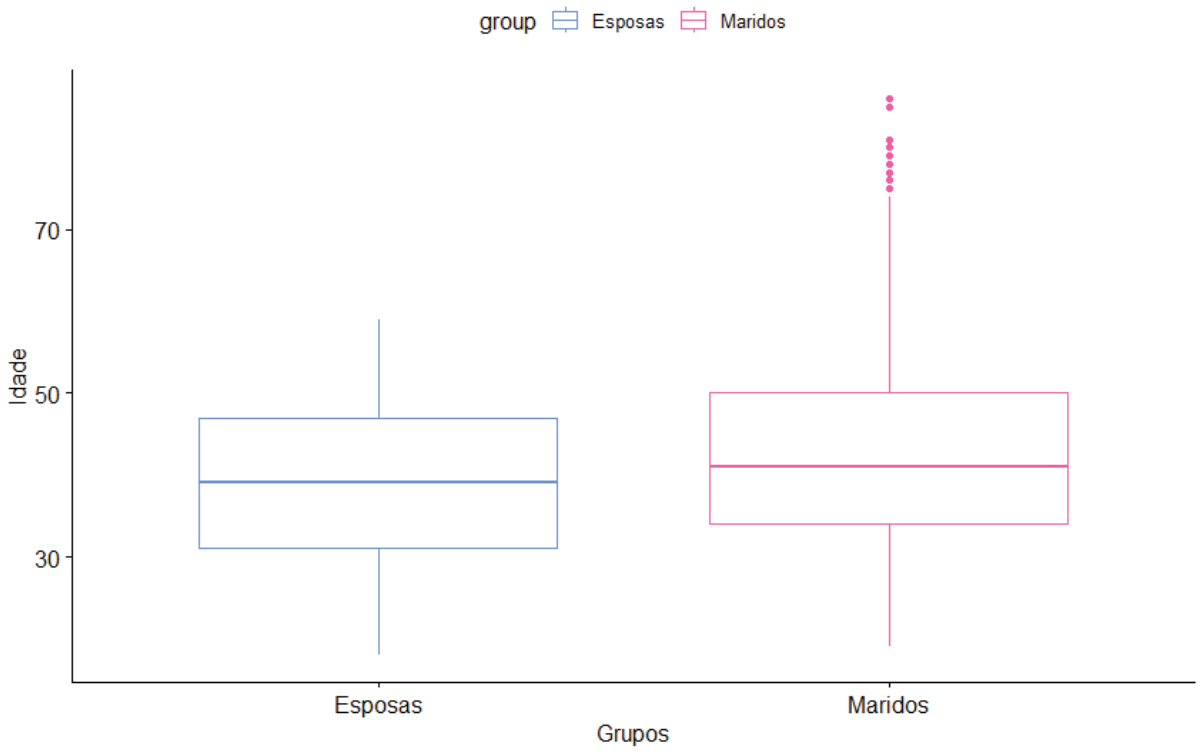
Figura 4.2 – EXERCÍCIO 3: DISTRIBUIÇÃO NORMAL E HISTOGRAMA




```
# Teste de normalidade de Kolmogorov-Smirnov
```

```
lillie.test(idades.df$idades)
D = 0.060083
p-value < 2.2e-16

# Sumario estatistico
group_by(idades.df, group) %>%
  summarise(
    count = n(),
    median = median(idades, na.rm = TRUE),
    IQR = IQR(idades, na.rm = TRUE)
  )
# Box-plots
ggboxplot(idades.df, x = "group", y = "idades", color = "group",
   palette=c("#6495ED", "#F25EA2"), ylab = "Idade", xlab = "Grupos")
```

Figura 4.3 – EXERCÍCIO 3: BOX-PLOTS




```
# Teste se a idade mediana dos maridos é igual a mediana das esposas
wilcox.test(idades ~ group, data = idades.df, exact = FALSE, conf.int
   =TRUE)
```

Estatisticamente, as idades medianas dos maridos e das esposas são diferentes. O resultado do teste estatístico é altamente significativo (p-valor = 0,00000000000000022), o que significa que podemos rejeitar a hipótese de medianas iguais. O intervalo de confiança da diferença entre as medianas também reforça essa conclusão, indicando que a mediana dos maridos é entre 2 e 3 anos menor que a mediana das esposas.

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A - ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B - RESOLUÇÃO

1. Regressões Ridge, Lasso e ElasticNet

```r
# Carregar os dados
setwd("C:/Users/marco/UFPR/IAA/IAA005-Estatística 2")
load("trabalhosalarios.RData")

# Verificar a estrutura dos dados
str(trabalhosalarios)
```

Tabela 5.1: Estrutura do data.frame (N = 2574, 16 variáveis)

variável	tipo	exemplos (início da coluna)
husage	num	56, 31, 33, 34, 42, 45, 33, 31, 31, 44, ...
husunion	num	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
husearns	num	1500, 800, 950, 1000, 730, ...
huseduc	num	14, 17, 13, 14, 14, 16, 16, 18, 12, 12, ...
husblck	num	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
hushisp	num	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
hushrs	num	40, 40, 60, 50, 40, 38, 40, 55, 40, 40, ...
kidge6	num	1, 0, 0, 1, 1, 1, 0, 0, 0, 1, ...
age	num	49, 29, 30, 31, 41, 45, 32, 27, 30, 42, ...
black	num	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
educ	num	12, 14, 12, 12, 12, 18, 12, 14, 15, 12, ...
hispanic	num	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
union	num	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
exper	num	31, 9, 12, 13, 23, 21, 14, 7, 9, 24, ...
kidlt6	num	0, 0, 1, 0, 0, 0, 0, 1, 1, 0, ...
lwage	num	1.90, 2.48, 2.43, 1.63, 2.30, ...

```r
# Separar a variável dependente (lwage) e as variáveis independentes
y <- trabalhosalarios$lwage
X <- trabalhosalarios[, setdiff(names(trabalhosalarios), "lwage")]

# Instalar e carregar os pacotes necessários
if(!require(glmnet)) install.packages("glmnet", dependencies=TRUE)
if(!require(caret)) install.packages("caret", dependencies=TRUE)
library(glmnet)
library(caret)

# Definir os parâmetros para os modelos
train_control <- trainControl(method = "cv", number = 10)

# Modelo Ridge
set.seed(123)
model_ridge <- train(
  lwage ~ ., data = trabalhosalarios, method = "glmnet",
  trControl = train_control,
  tuneGrid = expand.grid(alpha = 0, lambda = seq(0.0001, 1, length =
    100))
)

# Modelo Lasso
set.seed(123)
```

```
model_lasso <- train(
  lwage ~ ., data = trabalhosalarios, method = "glmnet",
  trControl = train_control,
  tuneGrid = expand.grid(alpha = 1, lambda = seq(0.0001, 1, length =
   100))
)

# Modelo ElasticNet
set.seed(123)
model_elastic <- train(
  lwage ~ ., data = trabalhosalarios, method = "glmnet",
  trControl = train_control,
  tuneLength = 10
)

# Comparar os modelos
results <- resamples(list(Ridge = model_ridge, Lasso = model_lasso,
   ElasticNet = model_elastic))
summary(results)
```

Tabela 5.2: MAE por modelo (10 reamostragens)

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Ridge	0.308 521 7	0.317 837 4	0.333 715 7	0.330 771 4	0.340 441 1	0.360 415 0	0
Lasso	0.309 219 9	0.317 431 0	0.333 607 5	0.330 620 3	0.339 852 7	0.359 883 4	0
ElasticNet	0.308 933 6	0.318 400 0	0.333 550 7	0.330 532 9	0.338 915 8	0.360 162 7	0

Tabela 5.3: RMSE por modelo (10 reamostragens)

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Ridge	0.390 759 5	0.409 507 5	0.434 672 9	0.441 576 2	0.448 293 3	0.549 436 2	0
Lasso	0.391 678 2	0.409 944 1	0.434 323 5	0.441 793 9	0.449 545 3	0.549 327 3	0
ElasticNet	0.391 631 7	0.410 748 7	0.433 751 9	0.441 500 8	0.446 619 7	0.550 325 0	0

Tabela 5.4: R^2 por modelo (10 reamostragens)

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Ridge	0.208 528 8	0.260 878 7	0.270 270 6	0.283 137 8	0.320 461 0	0.344 000 5	0
Lasso	0.210 658 8	0.261 484 9	0.269 788 0	0.282 500 7	0.320 377 8	0.343 299 7	0
ElasticNet	0.209 416 1	0.264 155 0	0.273 555 9	0.283 372 5	0.321 804 2	0.338 680 8	0

```
# Coletar as estatísticas dos modelos
rmse_ridge <- min(model_ridge$results$RMSE)
r2_ridge <- max(model_ridge$results$Rsquared)
mae_ridge <- min(model_ridge$results$MAE)

rmse_lasso <- min(model_lasso$results$RMSE)
r2_lasso <- max(model_lasso$results$Rsquared, na.rm = TRUE)
```

```r
mae_lasso <- min(model_lasso$results$MAE)

rmse_elastic <- min(model_elastic$results$RMSE)
r2_elastic <- max(model_elastic$results$Rsquared)
mae_elastic <- min(model_elastic$results$MAE)

# Criar a tabela das estatísticas dos modelos
stats <- data.frame(
  Model = c("Ridge", "Lasso", "ElasticNet"),
  RMSE = c(rmse_ridge, rmse_lasso, rmse_elastic),
  R2 = c(r2_ridge, r2_lasso, r2_elastic),
  MAE = c(mae_ridge, mae_lasso, mae_elastic)
)
print(stats)
```

Tabela 5.5: Comparativo de modelos

Model	RMSE	R^2	MAE
Ridge	0.441 576 2	0.283 226 9	0.330 771 4
Lasso	0.441 793 9	0.282 809 5	0.330 620 3
ElasticNet	**0.4415008**	**0.2833890**	**0.3305329**

```r
# Valores fornecidos para a predição
new_data <- data.frame(
  husage = 40, husunion = 0, husearns = 600, huseduc = 13, husblck =
   1, hushisp = 0, hushrs = 40, kidge6 = 1, age = 38, black = 0, educ
    = 13, hispanic = 1, union = 0, exper = 18, kidlt6 = 1
)

# Previsão com o melhor modelo (ElasticNet)
pred <- predict(model_elastic, new_data)

# Aplicar o antilogaritmo para obter o salário-hora
pred_salary <- exp(pred)

# Exibir o resultado
print(pred_salary)
```

Resultado: 7.998988

APÊNDICE 6 – ARQUITETURA DE DADOS

A - ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B - RESOLUÇÃO

1 Construção de Características: Identificador automático de idioma

```
1  ingles = [
2      "Hello, how are you?",
3      "I love to read books.",
4      "The weather is nice today.",
5      "Where is the nearest restaurant?",
6      "What time is it?",
7      "I enjoy playing soccer.",
8      "Can you help me with this?",
9      "I'm going to the movies tonight.",
10     "This is a beautiful place.",
11     "I like listening to music.",
12     "Do you speak English?",
13     "What is your favorite color?",
14     "I'm learning to play the guitar.",
15     "Have a great day!",
```

```
    "I need to buy some groceries.",
    "Let's go for a walk.",
    "How was your weekend?",
    "I'm excited for the concert.",
    "Could you pass me the salt, please?",
    "I have a meeting at 2 PM.",
    "I'm planning a vacation.",
    "She sings beautifully.",
    "The cat is sleeping.",
    "I want to learn French.",
    "I enjoy going to the beach.",
    "Where can I find a taxi?",
    "I'm sorry for the inconvenience.",
    "I'm studying for my exams.",
    "I like to cook dinner at home.",
    "Do you have any recommendations for restaurants?",
]
espanhol = [
    "Hola, cómo estás?",
    "Me encanta leer libros.",
    "El clima está agradable hoy.",
    "Dónde está el restaurante más cercano?",
    "Qué hora es?",
    "Voy al parque todos los días.",
    "Puedes ayudarme con esto?",
    "Me gustaría ir de vacaciones.",
    "Este es mi libro favorito.",
    "Me gusta bailar salsa.",
    "Hablas espaol?",
    "Cuál es tu comida favorita?",
    "Estoy aprendiendo a tocar el piano.",
    "Que tengas un buen día!",
    "Necesito comprar algunas frutas.",
    "Vamos a dar un paseo.",
    "Cómo estuvo tu fin de semana?",
    "Estoy emocionado por el concierto.",
    "Me pasas la sal, por favor?",
    "Tengo una reunión a las 2 PM.",
    "Estoy planeando unas vacaciones.",
    "Ella canta hermosamente.",
    "El perro está jugando.",
    "Quiero aprender italiano.",
    "Disfruto ir a la playa.",
    "Dónde puedo encontrar un taxi?",
    "Lamento las molestias.",
    "Estoy estudiando para mis exámenes.",
    "Me gusta cocinar la cena en casa.",
    "Tienes alguna recomendación de restaurantes?",
]
portugues = [
    "Estou indo para o trabalho agora.",
```

```
    "Adoro passar tempo com minha família.",
    "Preciso comprar leite e pão.",
    "Vamos ao cinema no sábado.",
    "Gosto de praticar esportes ao ar livre.",
    "O trânsito está terrível hoje.",
    "A comida estava deliciosa!",
    "Você já visitou o Rio de Janeiro?",
    "Tenho uma reunião importante amanhã.",
    "A festa começa às 20h.",
    "Estou cansado depois de um longo dia de trabalho.",
    "Vamos fazer um churrasco no final de semana.",
    "O livro que estou lendo é muito interessante.",
    "Estou aprendendo a cozinhar pratos novos.",
    "Preciso fazer exercícios físicos regularmente.",
    "Vou viajar para o exterior nas férias.",
    "Você gosta de dançar?",
    "Hoje é meu aniversário!",
    "Gosto de ouvir música clássica.",
    "Estou estudando para o vestibular.",
    "Meu time de futebol favorito ganhou o jogo.",
    "Quero aprender a tocar violão.",
    "Vamos fazer uma viagem de carro.",
    "O parque fica cheio aos finais de semana.",
    "O filme que assisti ontem foi ótimo.",
    "Preciso resolver esse problema o mais rápido possível.",
    "Adoro explorar novos lugares.",
    "Vou visitar meus avós no domingo.",
    "Estou ansioso para as férias de verão.",
    "Gosto de fazer caminhadas na natureza.",
    "O restaurante tem uma vista incrível.",
    "Vamos sair para jantar no sábado."
]

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

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

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

random.shuffle(pre_padroes)

dados = pd.DataFrame(pre_padroes)

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

```
118   for t in tamanhos:
119     soma=soma+t
120   return soma / len(tamanhos)
121 
122 # Calcula a frequência relativa de caracteres acentuados no texto
123 def frequenciaAcentos(texto):
124     texto_lower = texto.lower()
125     acentuados = 'ãõáéíóúàâêôç'
126     total_letras = sum(c.isalpha() for c in texto_lower)
127     if total_letras == 0:
128         return 0
129     count = sum(c in acentuados for c in texto_lower)
130     return count / total_letras
131 
132 # Calcula a frequência relativa de vogais no texto
133 def frequenciaVogais(texto):
134     texto_lower = texto.lower()
135     vogais = 'aeiou'
136     total_letras = sum(c.isalpha() for c in texto_lower)
137     if total_letras == 0:
138         return 0
139     count = sum(c in vogais for c in texto_lower)
140     return count / total_letras
141 
142 def extraiCaracteristicas(frase):
143   # frase é um vetor [ 'texto', 'lingua' ]
144   texto = frase[0]
145   pattern_regex = re.compile('[^\w+]', re.UNICODE)
146   texto = re.sub(pattern_regex,' ',texto)
147   caracteristica1=tamanhoMedioFrases(texto)
148   caracteristica2=frequenciaAcentos(texto)
149   caracteristica3=frequenciaVogais(texto)
150 
151   # Contagem de palavras típicas de cada idioma
152   palavras = texto.lower().split()
153   caracteristica4  = sum(w in palavras_portugues for w in palavras)
154   caracteristica5 = sum(w in palavras_ingles   for w in palavras)
155   caracteristica6 = sum(w in palavras_espanhol for w in palavras)
156 
157 
158   # acrescente as suas funcoes no vetor padrao
159   padrao = [caracteristica1, caracteristica2, caracteristica3,
       caracteristica4, caracteristica5, caracteristica6, frase[1] ]
160   return padrao
161 
162 # Conjuntos de palavras comuns em cada idioma
163 palavras_portugues = {'que','de','em','um','uma','por','não','vou','j
    á','estou','é','no','nos','sua','dos','das','os','as','pra'}
164 palavras_ingles    = {'the','and','to','of','a','in','that','i','is','
    it','for', 'you','he','she','they','do','we','not','are','have','
    just','so'}
```

```
palavras_espanhol = {'los','las', 'en','no','una','un', 'lo', 'eso','muy','sin','pero','él','ella'}

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

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

dados = pd.DataFrame(padroes)

##Treinando o modelo

####Separando o conjunto de treinamento do conjunto de testes
vet = np.array(padroes)
classes = vet[:,-1]
padroes_sem_classe = vet[:,0:-1]
X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe, classes, test_size=0.25, stratify=classes)

####Treinando o modelo com SVM
treinador = svm.SVC()  #algoritmo escolhido
modelo = treinador.fit(X_train, y_train)

# score com os dados de treinamento
acuracia = modelo.score(X_train, y_train)
y_pred = modelo.predict(X_train)
cm = confusion_matrix(y_train, y_pred)

# com dados de teste que não foram usados no treinamento
y_pred2 = modelo.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
```

Tabela 6.1: Matriz de confusão - Treino (acurácia: 84,06%)

	predito		
verdadeiro	**espanhol**	**inglês**	**português**
espanhol	19	0	3
inglês	2	21	0
português	5	1	18

Tabela 6.2: Relatório de classificação — Treino

classe	precisão	revocação	f1-score	suporte
espanhol	0.73	0.86	0.79	22
inglês	0.95	0.91	0.93	23
português	0.86	0.75	0.80	24
accuracy		**0.84**		**69**
macro avg	0.85	0.84	0.84	69
weighted avg	0.85	0.84	0.84	69

Tabela 6.3: Matriz de confusão - Amostra (acurácia: 70%)

	predito		
verdadeiro	**espanhol**	**inglês**	**português**
espanhol	5	1	2
inglês	1	6	0
português	3	0	5

Tabela 6.4: Relatório de classificação - Amostra

classe	precisão	revocação	f1-score	suporte
espanhol	0.56	0.62	0.59	8
inglês	0.86	0.86	0.86	7
português	0.71	0.62	0.67	8
accuracy		**0.70**		**23**
macro avg	0.71	0.70	0.70	23
weighted avg	0.70	0.70	0.70	23

APÊNDICE 7 – APRENDIZADO DE MÁQUINA

A - ENUNCIADO

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

B - RESOLUÇÃO

Classificação - Veículos

Tabela 7.1: Veículos - Resultados por técnica com parâmetros, acurácia e matrizes de confusão.

Técnica	**Parâmetro**	**Acurácia**	**Matriz de Confusão**				
SVM – CV	C=100, Sigma=0.01	0.8443	Reference				
			Prediction	bus	opel	saab	van
			bus	40	0	1	0
			opel	0	31	10	0
			saab	0	10	31	0
			van	3	1	1	39
RNA – CV	size=31, decay=0.7	0.8204	Reference				
			Prediction	bus	opel	saab	van
			bus	41	0	0	0
			opel	0	27	13	0
			saab	0	13	30	0
			van	2	0	0	39
SVM – Hold-out	C=1, Sigma=0.07	0.7485	Reference				
			Prediction	bus	opel	saab	van
			bus	40	0	0	1
			opel	0	20	13	0
			saab	0	20	26	0
			van	3	0	0	39
RF – Hold-out	mtry=10	0.7425	Reference				
			Prediction	bus	opel	saab	van
			bus	40	0	1	0
			opel	0	19	13	0
			saab	0	20	26	0
			van	3	3	3	39

Continua na próxima página

Continuação da Tabela 7.1

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – CV	mtry=5	0.7305	Reference Prediction bus opel saab van bus 40 0 1 0 opel 0 18 14 0 saab 0 21 25 0 van 3 3 0 39
RNA – Hold-out	size=5, decay=0.1	0.6707	Reference Prediction bus opel saab van bus 35 1 0 0 opel 0 0 0 0 saab 2 9 43 5 van 6 2 0 34
KNN	k=1	0.6353	Reference Prediction bus opel saab van bus 39 1 3 2 opel 1 18 19 2 saab 7 23 15 0 van 0 4 0 36

Classificação - Diabetes

Tabela 7.2: Diabetes - Resultados por técnica com parâmetros, acurácia e matriz de confusão.

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – Hold-out	C=0.5, Sigma=0.14	0.7647	Reference Prediction neg pos neg 92 28 pos 8 25
RF – Hold-out	mtry=2	0.7647	Reference Prediction neg pos neg 86 22 pos 14 31
RF – CV	mtry=9	0.7647	Reference Prediction neg pos neg 83 19 pos 17 34
SVM – CV	C=2, Sigma=0.015	0.7582	Reference Prediction neg pos neg 91 28 pos 9 25

Continua na próxima página

Continuação da Tabela 7.2

Técnica	Parâmetro	Acurácia	Matriz de Confusão
KNN	k=7	0.7273	Reference Prediction neg pos neg 85 31 pos 11 27
RNA – CV	size=21, decay=0.4	0.7059	Reference Prediction neg pos neg 81 26 pos 19 27
RNA – Hold-out	size=5, decay=0.1	0.6732	Reference Prediction neg pos neg 78 28 pos 22 25

Regressão - Admissão

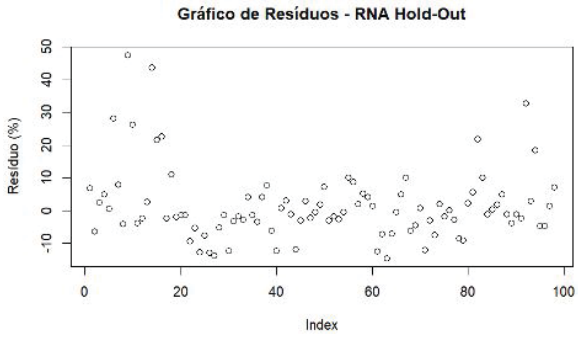
Tabela 7.3: Admissão - Resultados por técnica com parâmetros e métricas (R^2, S_{yx}, coeficiente de Pearson, RMSE e MAE).

Técnica	Parâmetro	R^2	S_{yx}	Pearson	RMSE	MAE
RNA – Hold-out	size=3, decay=1e-4	0.8234	0.0606	0.9083	0.0583	0.0434
RF – Hold-out	mtry=2	0.8231	0.0606	0.9089	0.0584	0.0431
RF – CV	mtry=2	0.8196	0.0612	0.9072	0.0590	0.0436
SVM – CV	C=50, Sigma=0.01	0.8160	0.0618	0.9063	0.0595	0.0430
SVM – Hold-out	C=1, Sigma=0.18	0.8014	0.0642	0.9007	0.0619	0.0438
KNN	k=9	0.7384	0.0737	0.8600	0.0710	0.0547
RNA – CV	size=41, decay=0.1	0.7344	0.0743	0.8703	0.0716	0.0534

Tabela 7.4: Amostra com GRE, TOEFL, rating universitário e métricas preditivas.

GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predict.rna
316.99	110.72	4.66	1.19	2.20	9.70	0.71	0.89
303.68	99.41	2.67	4.11	2.01	8.08	0.41	0.59
295.93	97.60	4.72	3.48	2.84	9.80	0.80	0.82

Figura 7.1 – ADMISSÃO - GRÁFICO DE RESÍDUOS (MELHOR MODELO)



Regressão - Biomassa

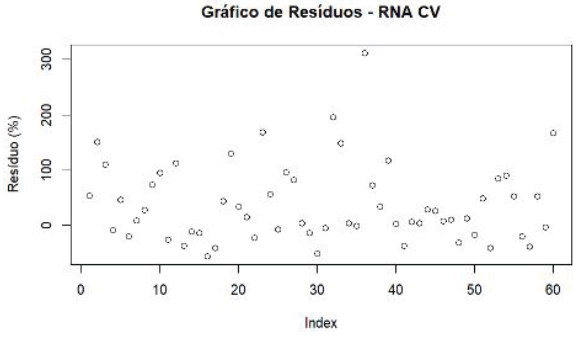
Tabela 7.5: Biomassa - Resultados por técnica com parâmetros e métricas (R^2, S_{yx}, coeficiente de Pearson, RMSE e MAE).

Técnica	**Parâmetro**	R^2	S_{yx}	**Pearson**	**RMSE**	**MAE**
RNA – CV	size=11, decay=0.4	0.9877	141.04	0.9939	137.47	64.407
RF – CV	mtry=5	0.9846	163.73	0.9923	153.88	61.807
RF – Hold-out	mtry=2	0.9843	165.45	0.9921	155.50	63.321
SVM – CV	C=100, Sigma=0.01	0.9448	310.55	0.9881	291.87	94.531
KNN	k=3	0.8576	498.83	0.9578	468.83	124.52
RNA – Hold-out	size=5, decay=0.1	0.5433	861.62	0.8205	839.80	186.86
SVM – Hold-out	C=1, Sigma=0.60	0.3435	1071.3	0.6175	1006.89	210.16

Tabela 7.6: Amostra com variáveis *dap*, *h*, *Me* e predição da RNA.

dap	**h**	**Me**	**predict.rna**
132.31429	23.76053	0.4378101	16744.171
55.74442	48.14359	0.6876391	2724.243
45.74574	14.74192	0.3821269	14206.252

Figura 7.2 – BIOMASSA - GRÁFICO DE RESÍDUOS (MELHOR MODELO)



Agrupamento - Veículo

K-means clustering with 10 clusters of size 75, 67, 54, 131, 24, 137, 78, 109, 70, 101

Tabela 7.7: K-means (médias por cluster) - Parte 1

Cl	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect
1	89.96	39.17	75.71	168.43	65.49	9.40	149.69	44.17	19.00	135.45
2	95.99	45.96	90.64	196.25	64.34	8.31	182.42	35.90	21.49	148.67
3	85.67	36.74	57.31	122.44	56.19	5.44	120.81	55.72	17.22	128.41
4	85.61	43.20	70.41	138.50	58.93	7.20	148.49	45.30	18.94	143.68
5	107.25	55.54	103.42	190.25	55.79	5.79	248.71	26.88	27.08	168.08
6	104.12	53.66	102.67	201.88	62.42	10.47	217.18	30.70	24.42	169.28
7	102.14	49.71	100.51	204.21	63.67	9.37	200.67	32.81	22.95	156.59
8	88.88	38.89	66.83	138.25	57.81	7.27	134.41	49.92	18.03	135.41
9	93.47	41.33	80.15	188.79	66.96	9.14	165.79	39.61	20.14	139.43
10	91.05	45.54	80.15	157.95	63.08	10.02	156.26	43.02	19.56	152.19

Tabela 7.8: K-means (médias por cluster) - Parte 2

Cl	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HolIRa
1	173.88	335.92	142.24	69.19	4.72	14.37	193.47	200.41
2	203.27	502.81	179.22	68.01	6.24	12.97	193.78	200.34
3	140.31	215.50	134.94	74.96	7.00	12.19	185.94	190.83
4	170.03	324.98	173.22	77.76	6.00	9.14	183.64	188.79
5	271.29	910.54	247.17	83.63	6.50	14.21	182.67	183.63
6	227.10	697.25	212.86	71.92	7.58	15.88	187.91	197.39
7	218.59	603.91	199.73	69.38	6.46	14.76	191.15	199.00
8	156.91	267.55	142.82	71.35	6.09	11.05	189.72	194.93
9	192.69	414.83	155.57	69.83	5.40	15.00	194.31	200.69
10	176.25	360.31	178.36	73.12	7.11	9.27	187.40	195.52

Associação - Musculação

Tabela 7.9: Regras de associação — top 10 linhas.

Lhs	Rhs	Support	Confidence	Coverage	Lift	Count
{V2=Gêmeos}	{V3=Bicicleta}	0.026	1.0	0.026	3.2	1
{V2=Gêmeos}	{V4=Esteira}	0.026	1.0	0.026	3.2	1
{V2=Gêmeos}	{V1=Extensor}	0.026	1.0	0.026	1.0	1
{V3=Gêmeos}	{V2=AgachamentoSmith}	0.051	1.0	0.051	3.9	2
{V3=Gêmeos}	{V4=Esteira}	0.051	1.0	0.051	3.2	2
{V4=Crucifixo}	{V3=Afundo}	0.051	1.0	0.051	6.5	2
{V4=Crucifixo}	{V2=Gêmeos}	0.051	1.0	0.051	5.5	2
{V4=Crucifixo}	{V1=LegPress}	0.051	1.0	0.051	2.6	2
{V2=Flexor}	{V3=Bicicleta}	0.051	1.0	0.051	3.2	2
{V2=Flexor}	{V4=Esteira}	0.051	1.0	0.051	3.2	2

Código

```
install.packages("e1071")
install.packages("caret")
install.packages("Metrics")

library("caret")
library("Metrics")

% CLASSIFICACAO - VEICULOS

setwd("/Users/blb/Documents/ESPECIALIZAÇÃO EM IA/APRENDIZADO DE MAQUINA/CLASSIFICACAO/")
dados <- read.csv("6 - Veiculos - Dados.csv")
dados[[a]] <- NULL

% KNN

set.seed(202410)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]
set.seed(202410)
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
knn <- train(tipo ~ ., data = treino, method = "knn",tuneGrid= tuneGrid)
predict.knn <- predict(knn, teste)
confusionMatrix(predict.knn, as.factor(teste[["tipo"]]))

% RNA HOLD OUT

set.seed(202410)
indices <- createDataPartition(dados[["tipo"]], p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rna <- train(tipo ~ ., data=treino, method="nnet",trace=FALSE)
predicoes.rna <- predict(rna, teste)
confusionMatrix(predicoes.rna, as.factor(teste[["tipo"]]))

% RNA - Cross-Validation

ctrl <- trainControl(method = "cv", number = 10)
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay = seq(from = 0.1, to = 0.9,
by = 0.3))
set.seed(202410)
rna <- train(
    form = tipo~.,
    data = treino,
    method = "nnet",
    tuneGrid = grid,
    trControl = ctrl,
```

```
    maxit = 2000,trace=FALSE)
predicoes.rna <- predict(rna, teste)
confusionMatrix(predicoes.rna, as.factor(teste[["tipo"]]))

% SVM - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["tipo"]], p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
svm <- train(tipo~., data=treino, method="svmRadial")
predicoes.svm <- predict(svm, teste)
confusionMatrix(predicoes.svm, as.factor(teste[["tipo"]]))

% SVM - Cross-Validation

set.seed(202410)
indices <- createDataPartition(dados[["tipo"]], p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
   0.2))
set.seed(202410)
svm <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl,
    tuneGrid=tuneGrid)
predicoes.svm <- predict(svm, teste)
confusionMatrix(predicoes.svm, as.factor(teste[["tipo"]]))

% RANDOM FOREST - Hold-out

dados_aux <- dados
dados_aux[["tipo"]] <- NULL
newdata <-lapply(dados_aux, function(coluna) {
return(runif(3, min = min(coluna), max = max(coluna)))
})
predicoes.svm <- predict(svm, data.frame(newdata))
resultado <- cbind(data.frame(newdata), predicoes.svm)
View(resultado)

set.seed(202410)
indices <- createDataPartition(dados[["tipo"]], p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rf <- train(tipo~., data=treino, method="rf")

predicoes.rf <- predict(rf, teste)
confusionMatrix(predicoes.rf, as.factor(teste[["tipo"]]))

```

```
% RANDOM FOREST - Cross-Validation

set.seed(202410)
indices <- createDataPartition(dados[["tipo"]], p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202410)
rf <- train(tipo~., data=treino, method="rf", trControl=ctrl,
   tuneGrid=tuneGrid)
predicoes.rf <- predict(rf, teste)
confusionMatrix(predicoes.rf, as.factor(teste[["tipo"]]))
```

```
% CLASSIFICACAO - DIABETES

setwd("/Users/blb/Documents/ESPECIALIZAÇÃO EM IA/APRENDIZADO DE
   MAQUINA/CLASSIFICACAO/")
dados <- read.csv("10 - Diabetes - Dados.csv")
dados[["num"]] <- NULL

% KNN

set.seed(202410)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]
set.seed(202410)
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
knn <- train(diabetes ~ ., data = treino, method = "knn",tuneGrid=
   tuneGrid)
predict.knn <- predict(knn, teste)
confusionMatrix(predict.knn, as.factor(teste[["diabetes"]]))

% RNA - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["diabetes"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rna <- train(diabetes ~ ., data=treino, method="nnet",trace=FALSE)
predicoes.rna <- predict(rna, teste)
confusionMatrix(predicoes.rna, as.factor(teste[["diabetes"]]))

% RNA - Cross-Validation

ctrl <- trainControl(method = "cv", number = 10)
    grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay
   = seq(from = 0.1, to = 0.9,
```

```
    by = 0.3))
    set.seed(202410)
    rna <- train(
    form = diabetes~.,
    data = treino,
    method = "nnet",
    tuneGrid = grid,
    trControl = ctrl,
    maxit = 2000,trace=FALSE)
predicoes.rna <- predict(rna, teste)
confusionMatrix(predicoes.rna, as.factor(teste[["diabetes"]]))

% SVM - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["diabetes"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
svm <- train(diabetes~., data=treino, method="svmRadial")
predicoes.svm <- predict(svm, teste)
confusionMatrix(predicoes.svm, as.factor(teste[["diabetes"]]))

dados_aux <- dados
dados_aux[["diabetes"]] <- NULL
newdata <-lapply(dados_aux, function(coluna) {
return(runif(3, min = min(coluna), max = max(coluna)))
})
predicoes.svm <- predict(svm, data.frame(newdata))
resultado <- cbind(data.frame(newdata), predicoes.svm)

% SVM - Cross-Validation

set.seed(202410)
indices <- createDataPartition(dados[["diabetes"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
   0.2))
set.seed(202410)
svm <- train(diabetes~., data=treino, method="svmRadial", trControl=
   ctrl, tuneGrid=tuneGrid)
predicoes.svm <- predict(svm, teste)
confusionMatrix(predicoes.svm, as.factor(teste[["diabetes"]]))

% RANDOM FOREST - Hold-out

set.seed(202410)
```

```
indices <- createDataPartition(dados[["diabetes"]], p=0.80, list=
    FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rf <- train(diabetes~., data=treino, method="rf")
predicoes.rf <- predict(rf, teste)
confusionMatrix(predicoes.rf, as.factor(teste[["diabetes"]]))

% RANDOM FOREST - Cross-Validation

set.seed(202410)
indices <- createDataPartition(dados[["diabetes"]], p=0.80, list=
    FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202410)
rf <- train(diabetes~., data=treino, method="rf", trControl=ctrl,
    tuneGrid=tuneGrid)
predicoes.rf <- predict(rf, teste)
confusionMatrix(predicoes.rf, as.factor(teste[["diabetes"]]))
```

```
% REGRESSAO - ADMISSAO

setwd("/Users/blb/Documents/ESPECIALIZAÇÃO EM IA/APRENDIZADO DE MÁ
    QUINA/REGRESSÃO/")
dados <- read.csv("9 - Admissao - Dados.csv", header=T)
dados[["num"]] <- NULL

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

syx <- function(predito, observado, num_variaveis) {
return(sqrt((sum((predito-observado)^2) / (length(predito) -
    num_variaveis))))
}

regression_metrics <- function(predito, observado){
    r2_aux <- r2(predito, observado)
    syx_aux <- syx(predito, observado, 7)
    pearson_aux <- cor(predito, observado, method='pearson')
    rsme_aux <- rmse(predito, observado)
    mae_aux <- mae(predito, observado)
    return(c(r2_aux, syx_aux, pearson_aux, rsme_aux, mae_aux))
}

set.seed(202410)
```

```
ind <- createDataPartition(dados[["ChanceOfAdmit"]], p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
set.seed(202410)
knn <- train(ChanceOfAdmit ~ ., data = treino, method = "knn", tuneGrid=tuneGrid)
predict.knn <- predict(knn, teste)
regression_metrics(predict.knn,teste[["ChanceOfAdmit"]])

% RNA - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["ChanceOfAdmit"]], p=0.80, list =FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rna <- train(ChanceOfAdmit ~ ., data=treino, method="nnet",trace= FALSE)
predict.rna <- predict(rna, teste)
regression_metrics(predict.rna, teste[["ChanceOfAdmit"]])
residuos <- function(predito, observado){
return(((predito - observado)/observado) * 100)
}
plot(residuos(predict.rna, teste[["ChanceOfAdmit"]]), ylab='Resíduo (%)', main='Gráfico de Resíduo
s - RNA Hold-Out')

dados_aux <- dados
dados_aux[["ChanceOfAdmit"]] <- NULL
newdata <-lapply(dados_aux, function(coluna) {
    return(runif(3, min = min(coluna), max = max(coluna)))
})

predict.rna <- predict(rna, data.frame(newdata))
resultado <- cbind(data.frame(newdata), predict.rna)

% RNA - Cross-Validation

ctrl <- trainControl(method = "cv", number = 10)
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay = seq(from = 0.1, to = 0.9,
by = 0.3))
set.seed(202410)
rna <- train(
    form = ChanceOfAdmit~.,
    data = treino,
    method = "nnet",
    tuneGrid = grid,
```

```
    trControl = ctrl,
    maxit = 2000,trace=FALSE)

predict.rna <- predict(rna, teste)
regression_metrics(predict.rna,teste[["ChanceOfAdmit"]])

% SVM - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["ChanceOfAdmit"]], p=0.80, list
   =FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
svm <- train(ChanceOfAdmit~., data=treino, method="svmRadial")
predict.svm <- predict(svm, teste)
regression_metrics(predict.svm,teste[["ChanceOfAdmit"]])

% SVM - Cross-Validation

set.seed(202410)
indices <- createDataPartition(dados[["ChanceOfAdmit"]], p=0.80, list
   =FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
   0.2))
set.seed(202410)
svm <- train(ChanceOfAdmit~., data=treino, method="svmRadial",
   trControl=ctrl, tuneGrid=tuneG
rid)
predict.svm <- predict(svm, teste)
regression_metrics(predict.svm,teste[["ChanceOfAdmit"]])

% RANDOM FOREST - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["ChanceOfAdmit"]], p=0.80, list
   =FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rf <- train(ChanceOfAdmit~., data=treino, method="rf")
predict.rf <- predict(rf, teste)
regression_metrics(predict.rf,teste[["ChanceOfAdmit"]])

% RANDOM FOREST - Cross-Validation

set.seed(202410)
```

```
indices <- createDataPartition(dados[["ChanceOfAdmit"]], p=0.80, list
    =FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202410)
rf <- train(ChanceOfAdmit~., data=treino, method="rf", trControl=ctrl
    , tuneGrid=tuneGrid)
predict.rf <- predict(rf, teste)
regression_metrics(predict.rf,teste[["ChanceOfAdmit"]])
```

```
% REGRESSAO - BIOMASSA

setwd("/Users/blb/Documents/ESPECIALIZAÇÃO EM IA/APRENDIZADO DE MÁ
    QUINA/REGRESSÃO/")
dados <- read.csv("5 - Biomassa - Dados.csv", header=T)
dados[["num"]] <- NULL

% KNN

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

regression_metrics <- function(predito, observado){
    r2_aux <- r2(predito, observado)
    syx_aux <- syx(predito, observado, 3)
    pearson_aux <- cor(predito, observado, method='pearson')
    rsme_aux <- rmse(predito, observado)
    mae_aux <- mae(predito, observado)
    return(c(r2_aux, syx_aux, pearson_aux, rsme_aux, mae_aux))
}

residuos <- function(predito, observado){
    return(((predito - observado)/observado) * 100)
}

set.seed(202410)
ind <- createDataPartition(dados[["biomassa"]], p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
set.seed(202410)
knn <- train(biomassa ~ ., data = treino, method = "knn",tuneGrid=
    tuneGrid)
```

```
predict.knn <- predict(knn, teste)
regression_metrics(predict.knn,teste[["biomassa"]])

% RNA - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["biomassa"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rna <- train(biomassa ~ ., data=treino, method="nnet", trace=FALSE,
   linout=T)
predict.rna <- predict(rna, teste)
regression_metrics(predict.rna, teste[["biomassa"]])

% RNA - Cross-Validation

ctrl <- trainControl(method = "cv", number = 10)
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
   seq(from = 0.1, to = 0.9,
by = 0.3))
set.seed(202410)
rna <- train(
form = biomassa~.,
data = treino,
method = "nnet",
tuneGrid = grid,
trControl = ctrl,
maxit = 2000,trace=FALSE, linout=T)
predict.rna <- predict(rna, teste)
regression_metrics(predict.rna,teste[["biomassa"]])

dap = runif(3, min = min(dados[["dap"]]), max = max(dados[["dap"]]))
h = runif(3, min = min(dados[["h"]]), max = max(dados[["h"]]))
Me = runif(3, min = min(dados[["Me"]]), max = max(dados[["Me"]]))
newdata <- data.frame(dap=dap,h=h, Me=Me)
predict.rna <- predict(rna, newdata)
resultado <- cbind(newdata, predict.rna)

% SVM - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["biomassa"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
svm <- train(biomassa~., data=treino, method="svmRadial")
predict.svm <- predict(svm, teste)
```

```
regression_metrics(predict.svm,teste[["biomassa"]])

% SVM - Cross-Validation

set.seed(202410)
indices <- createDataPartition(dados[["biomassa"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
   0.2))
set.seed(202410)
svm <- train(biomassa~., data=treino, method="svmRadial", trControl=
   ctrl, tuneGrid=tuneGrid)
predict.svm <- predict(svm, teste)
regression_metrics(predict.svm,teste[["biomassa"]])

% RANDOM FOREST - Hold-out

set.seed(202410)
indices <- createDataPartition(dados[["biomassa"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
set.seed(202410)
rf <- train(biomassa~., data=treino, method="rf")
predict.rf <- predict(rf, teste)
regression_metrics(predict.rf,teste[["biomassa"]])

% RANDOM FOREST - Cross-Validation

set.seed(202410)
indices <- createDataPartition(dados[["biomassa"]], p=0.80, list=
   FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
ctrl <- trainControl(method = "cv", number = 10)
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202410)
rf <- train(biomassa~., data=treino, method="rf", trControl=ctrl,
   tuneGrid=tuneGrid, linout=
T)
predict.rf <- predict(rf, teste)
regression_metrics(predict.rf,teste[["biomassa"]])
```

```
% AGRUPAMENTO - VEICULOS

setwd("/Users/blb/Documents/ESPECIALIZAÇÃO EM IA/APRENDIZADO DE MÁ
   QUINA/AGRUPAMENTO/")
dados <- read.csv("6 - Veiculos - Dados.csv")
```

```
dados[["a"]] <- NULL
dados[["tipo"]] <- NULL

set.seed(202410)
dados_normalizados <- scale(dados)
km.res = kmeans(dados, 10)
```

```
% REGRAS DE ASSOCIACAO - MUSCULACAO

install.packages('arules', dep=T)
library(arules)
library(datasets)

setwd("/Users/blb/Documents/ESPECIALIZAÇÃO EM IA/APRENDIZADO DE MÁQUINA/ASSOCIAÇÃO/")
dados <- read.csv("2 - Musculacao - Dados(in).csv", header=0, sep=';')

set.seed(202410)
rules <- apriori(dados, parameter = list(supp = 0.001, conf = 0.7, minlen=2))
summary(rules)
options(digits=2)
inspect(sort(rules[1:20], by="confidence"))
```

APÊNDICE 8 – DEEP LEARNING

A - ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B - RESOLUÇÃO

1 Classificação de Imagens

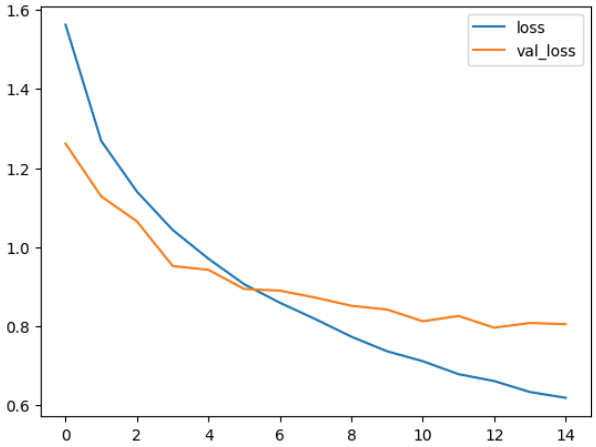
```
1  import tensorflow as tf
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten,
       Dropout
5  from tensorflow.keras.models import Model
6  from mlxtend.plotting import plot_confusion_matrix
7  from sklearn.metrics import confusion_matrix
8
9  cifar10 = tf.keras.datasets.cifar10
10 (x_train, y_train), (x_test, y_test) = cifar10.load_data()
11
12 x_train, x_test = x_train/255.0, x_test/255.0
13 y_train, y_test = y_train.flatten(), y_test.flatten()
14
15 print("x_train.shape", x_train.shape)
16 print("y_train.shape", y_train.shape)
17 print("x_test.shape", x_test.shape)
18 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,)

```
1  K = len(set(y\_train))
2
3  i = Input(shape=x\_train[0].shape)
4
5  x = Conv2D(32, (3,3), strides=2, activation="relu")(i)
```

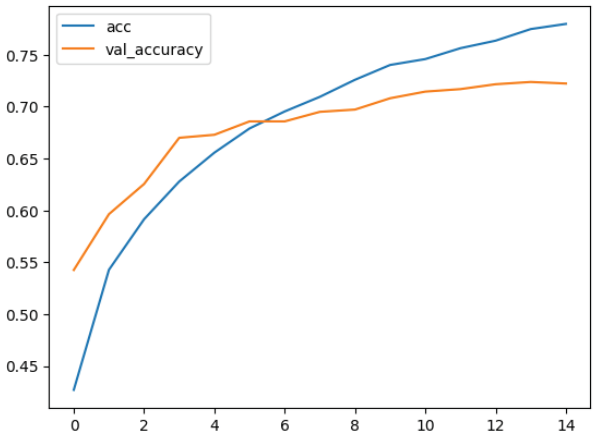
```
x = Conv2D(64, (3,3), strides=2, activation="relu")(x)
x = Conv2D(128, (3,3), strides=2, activation="relu")(x)

x = Flatten()(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)
model = Model(i, x)

model.summary()

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

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

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

Figura 8.1 – CLASSIFICAÇÃO DE IMAGENS COM CNN: GRÁFICO DE PERDA



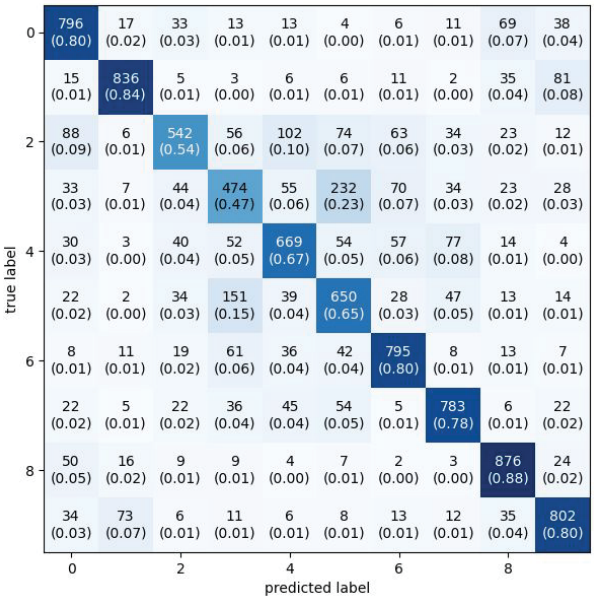

```
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_accuracy")
plt.legend()
plt.show()
```

Figura 8.2 – CLASSIFICAÇÃO DE IMAGENS COM CNN: GRÁFICO DE ACURÁCIA



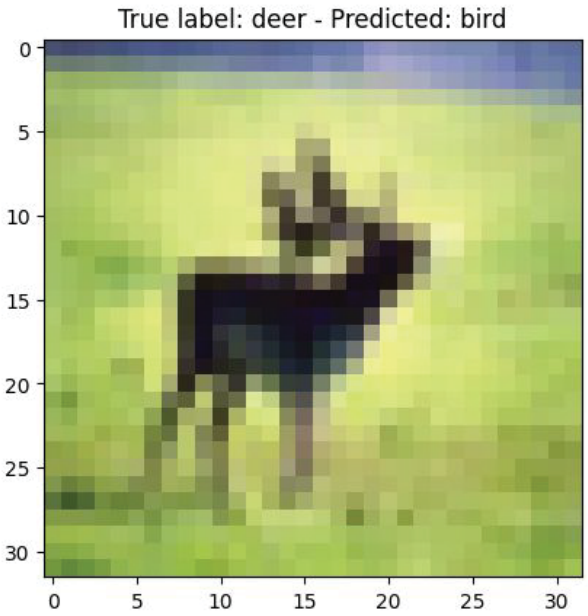

```
y_pred = model.predict(x_test).argmax(axis=1)
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7,7), show_normed=True)
```

Figura 8.3 – CLASSIFICAÇÃO DE IMAGENS COM CNN: MATRIZ DE CONFUSÃO




```

abels = ["airplane", "automobile", "bird", "cat", "deer", "dog", "
    frog", "horse", "ship", "truck"]

misclassified = np.where(y\_pred != y\_test)[0]

i = np.random.choice(misclassified)

plt.imshow(x\_test[i], cmap="gray")
plt.title("True label: \%s - Predicted: \%s" \% (labels[y\_test[i]],
    labels[y\_pred[i]]))
```

Figura 8.4 – CLASSIFICAÇÃO DE IMAGENS COM CNN: RESULTADO DA PREDIÇÃO



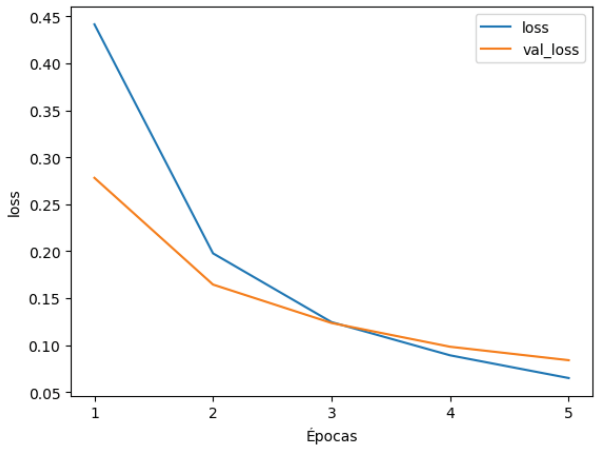
2 Detector de SPAM

```
1  import tensorflow as tf
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import pandas as pd
5  from sklearn.model_selection import train_test_split
6  from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
7  from tensorflow.keras.layers import GlobalMaxPooling1D
8  from tensorflow.keras.models import Model
9  from tensorflow.keras.preprocessing.sequence import pad_sequences
10 from tensorflow.keras.preprocessing.text import Tokenizer
11 from google.colab import files
12 import io
13
14 uploaded = files.upload()
15
16 df = pd.read_csv(io.BytesIO(uploaded['spam.csv']), encoding="ISO
       -8859-1")
17 df.head()
18
19 df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
20 df.columns = ["labels", "data"]
21 df["b_labels"] = df["labels"].map({"ham": 0, "spam": 1})
22 y = df["b_labels"].values
23
24 x_train, x_test, y_train, y_test = train_test_split(df["data"], y,
       test_size=0.33)
25
26 num_words = 2000
27 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)

data_train = pad_sequences(sequences_train)
T = data_train.shape[1]
data_test = pad_sequences(sequences_test, maxlen=T)
print("data_train.shape = ", data_train.shape)
print("data_test.shape = ", data_test.shape)

D = 20
M = 5

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

model = Model(i,x)

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

epochs = 5

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

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, 1), labels=range(1, epochs + 1))
plt.legend()
plt.show()
```

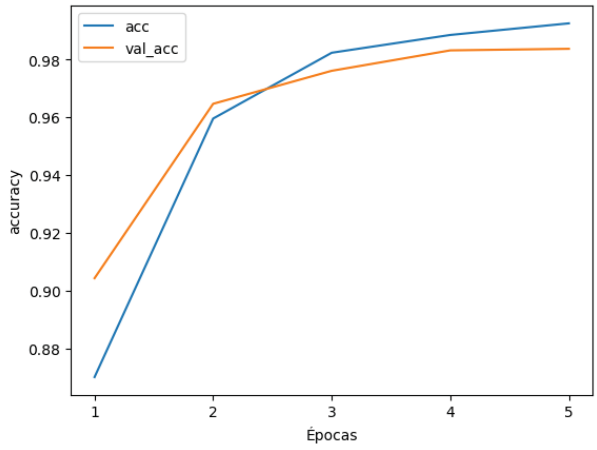
```
7242 tokens
data_train.shape = (3733, 175)
data_test.shape = (1839, 175)
```

Figura 8.5 – DETECTOR DE SPAM COM RNN: GRÁFICO DE PERDA




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

Figura 8.6 – DETECTOR DE SPAM COM RNN: GRÁFICO DE ACURÁCIA




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

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

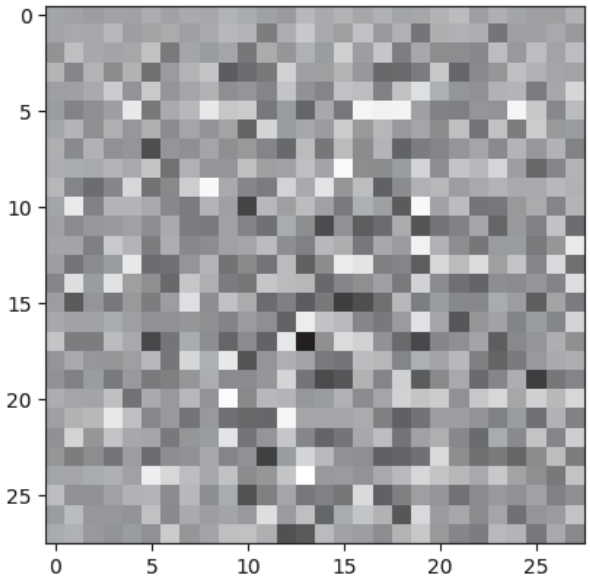
[[0.6281646]] SPAM

3 Gerador de Dígitos Fake

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

import glob
import imageio
import numpy as np
import os
import PIL
import time
import tensorflow as tf
import matplotlib.pyplot as plt

from tensorflow.keras import layers
from IPython import display

(train_images, train_labels) , (_,_) = tf.keras.datasets.mnist.load_data()
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).astype('float32')
train_images = (train_images - 127.5) / 127.5

BUFFER_SIZE = 60000
BATCH_SIZE = 256

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

def make_generator_model():
  model = tf.keras.Sequential()
  model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))

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

  model.add(layers.Reshape((7, 7, 256)))
  assert model.output_shape == (None, 7, 7, 256)

  model.add(layers.Conv2DTranspose(128, (5, 5), strides=1, 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())
```

```
46
47    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding
        ='same', use_bias=False, activation='tanh'))
48    assert model.output_shape == (None, 28, 28, 1)
49
50    return model
51
52  generator = make_generator_model()
53  noise = tf.random.normal([1, 100])
54  generated_image = generator(noise, training=False)
55  plt.imshow(generated_image[0, :, :, 0], cmap='gray')
```

Figura 8.7 – GERANDO DADOS COM GAN: GERAÇÃO DE IMAGEM RUIDOSA




```
1   def make_discriminator_model():
2     model = tf.keras.Sequential()
3     model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
          input_shape=[28, 28, 1]))
4     model.add(layers.LeakyReLU())
5     model.add(layers.Dropout(0.3))
6
7     model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'
        ))
8     model.add(layers.LeakyReLU())
9     model.add(layers.Dropout(0.3))
10
11    model.add(layers.Flatten())
12    model.add(layers.Dense(1))
13
14    return model
15
16  discriminator = make_discriminator_model()
17  decision = discriminator(generated_image)
```

```
18
19 cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
20
21 def discriminator_loss(real_output, fake_output):
22   real_loss = cross_entropy(tf.ones_like(real_output), real_output)
23   fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
24   total_loss = real_loss + fake_loss
25   return total_loss
26
27 def generator_loss(fake_output):
28   return cross_entropy(tf.ones_like(fake_output), fake_output)
29
30 generator_optimizer = tf.keras.optimizers.Adam(1e-4)
31 discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
32
33 checkpoint_dir = './training_checkpoints'
34 checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
35 checkpoint = tf.train.Checkpoint(generator_optimizer=
    generator_optimizer,
36                                  discriminator_optimizer=
    discriminator_optimizer,
37                                  generator=generator,
38                                  discriminator=discriminator)
39
40 EPOCHS = 100
41 noise_dim = 100
42 num_examples_to_generate = 16
43
44 seed = tf.random.normal([num_examples_to_generate, noise_dim])
45
46 @tf.function
47 def train_step(images):
48   noise = tf.random.normal([BATCH_SIZE, noise_dim])
49
50   with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
51     generated_images = generator(noise, training=True)
52
53     real_output = discriminator(images, training=True)
54     fake_output = discriminator(generated_images, training=True)
55
56     gen_loss = generator_loss(fake_output)
57     disc_loss = discriminator_loss(real_output, fake_output)
58
59     gradients_of_generator = gen_tape.gradient(gen_loss, generator.
    trainable_variables)
60     gradients_of_discriminator = disc_tape.gradient(disc_loss,
    discriminator.trainable_variables)
61
62     generator_optimizer.apply_gradients(zip(gradients_of_generator,
    generator.trainable_variables))
```

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

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

    for image_batch in dataset:
      train_step(image_batch)

    display.clear_output(wait=True)
    generate_and_save_images(generator, epoch + 1, seed)

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

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

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

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

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

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

train(train_dataset, EPOCHS)
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))
```

Figura 8.8 – GERANDO DADOS COM GAN: CHECKPOINT DA GERAÇÃO DE DÍGITOS

```
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)
```

4 Transformer

```
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0
import collections
import logging
import os
import pathlib
import re
import string
import sys
```

```
import time
import numpy as np
import matplotlib.pyplot as plt
import tensorflow_datasets as tfds
import tensorflow_text as text
import tensorflow as tf
logging.getLogger('tensorflow').setLevel(logging.ERROR)

examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en',
with_info=True, as_supervised=True)

train_examples, val_examples = examples['train'], examples['
    validation']

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

  print()

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

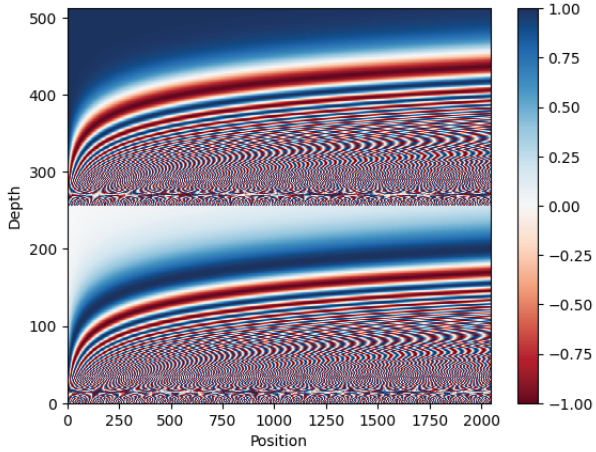
e quando melhoramos a procura , tiramos a única vantagem da impressão ,
que é a serendipidade .
mas e se estes fatores fossem ativos ?
mas eles não tinham a curiosidade de me testar .
and when you improve searchability , you actually take away the one
advantage of print , which is serendipity .
but what if it were active ?
but they did n't test for curiosity .

```
# Tokenização e Destokenização do texto
model_name = "ted_hrlr_translate_pt_en_converter"

tf.keras.utils.get_file(f"{model_name}.zip",
f"https://storage.googleapis.com/download.tensorflow.org/models/{
    model_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
```

```
20    en = en.to_tensor()
21
22    return pt, en
23
24  # Pipeline simpes: processa, embaralha, agrupa os dados, prefetch
25  # Datasets de entrada terminam com prefetch
26  BUFFER_SIZE = 20000
27  BATCH_SIZE = 64
28
29  def make_batches(ds):
30    return (
31      ds
32      .cache()
33      .shuffle(BUFFER_SIZE)
34      .batch(BATCH_SIZE)
35      .map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)
36      .prefetch(tf.data.AUTOTUNE))
37
38  train_batches = make_batches(train_examples)
39  val_batches = make_batches(val_examples)
40
41  # CODIFICAÇÃO POSICIONAL
42  def get_angles(pos, i, d_model):
43    angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model
       ))
44    return pos * angle_rates
45
46  def positional_encoding(position, d_model):
47    angle_rads = get_angles(np.arange(position)[:, np.newaxis], np.
       arange(d_model)[np.newaxis, :], d_model)
48
49    # sin em índices pares no array; 2i
50    angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
51
52    # cos em índices ímpares no array; 2i+1
53    angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
54
55    # newaxis, aumenta a dimensão [] -> [ [] ]
56    pos_encoding = angle_rads[np.newaxis, ...]
57
58    return tf.cast(pos_encoding, dtype=tf.float32)
59
60  # CODIFICAÇÃO POSICIONAL
61  n, d = 2048, 512
62  pos_encoding = positional_encoding(n, d)
63  print(pos_encoding.shape)
64  pos_encoding = pos_encoding[0]
65
66  # Arrumar as dimensões
67  pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
68  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()
```

Figura 8.9 – TRANFORMER: CODIFICAÇÃO POSICIONAL




```
# 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)
```

```
33
34      return out3, attn_weights_block1, attn_weights_block2
```

```
1  class Encoder(tf.keras.layers.Layer):
2    def __init__(self, num_layers, d_model, num_heads, dff,
       input_vocab_size, maximum_position_encoding, rate=0.1):
3      super(Encoder, self).__init__()
4
5      self.d_model = d_model
6      self.num_layers = num_layers
7      self.embedding = tf.keras.layers.Embedding(input_vocab_size,
       d_model)
8      self.pos_encoding = positional_encoding(maximum_position_encoding
       , self.d_model)
9      self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate)
       for _ in range(num_layers)]
10     self.dropout = tf.keras.layers.Dropout(rate)
11
12   def call(self, x, training, mask):
13     seq_len = tf.shape(x)[1]
14
15     # adding embedding and position encoding.
16     x = self.embedding(x) # (batch_size, input_seq_len, d_model)
17     x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
18     x += self.pos_encoding[:, :seq_len, :]
19     x = self.dropout(x, training=training)
20
21     for i in range(self.num_layers):
22       x = self.enc_layers[i](x, training, mask)
23
24     return x # (batch_size, input_seq_len, d_model)
```

```
1  class Decoder(tf.keras.layers.Layer):
2    def __init__(self, num_layers, d_model, num_heads, dff,
       target_vocab_size, maximum_position_encoding, rate=0.1):
3      super(Decoder, self).__init__()
4
5      self.d_model = d_model
6      self.num_layers = num_layers
7
8      self.embedding = tf.keras.layers.Embedding(target_vocab_size,
       d_model)
9      self.pos_encoding = positional_encoding(maximum_position_encoding
       , d_model)
10
11     self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate)
       for _ in range(num_layers)]
12     self.dropout = tf.keras.layers.Dropout(rate)
13
14   def call(self, x, enc_output, training, look_ahead_mask,
       padding_mask):
```

```
    seq_len = tf.shape(x)[1]
    attention_weights = {}

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

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

    for i in range(self.num_layers):
      x, block1, block2 = self.dec_layers[i](x, enc_output, training,
    look_ahead_mask, padding_mask)

      attention_weights[f'decoder_layer{i+1}_block1'] = block1
      attention_weights[f'decoder_layer{i+1}_block2'] = block2

    # x.shape == (batch_size, target_seq_len, d_model)
    return x, attention_weights
```

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

  def call(self, inputs, training):
    # Keras models prefer if you pass all your inputs in the first
    argument
    inp, tar = inputs

    enc_padding_mask, look_ahead_mask, dec_padding_mask = self.
    create_masks(inp, tar)

    # (batch_size, inp_seq_len, d_model)
    enc_output = self.encoder(inp, training, enc_padding_mask)

    # dec_output.shape == (batch_size, tar_seq_len, d_model)
    dec_output, attention_weights = self.decoder(tar, enc_output,
    training, look_ahead_mask, dec_padding_mask)

    # (batch_size, tar_seq_len, target_vocab_size)
    final_output = self.final_layer(dec_output)

    return final_output, attention_weights
```

```

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

    return enc_padding_mask, look_ahead_mask, dec_padding_mask

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

```

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

  def __call__(self, step):
    step = tf.cast(step, tf.float32) # Adicionado para evitar ERRO
    arg1 = tf.math.rsqrt(step)
    arg2 = step * (self.warmup_steps ** -1.5)
    return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)
```

```

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

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True, reduction='none')

def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)
```

```
  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask
  return tf.reduce_sum(loss_)/tf.reduce_sum(mask)

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

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

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

# Checkpoint
checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(transformer=transformer, optimizer=
   optimizer)
ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path,
   max_to_keep=5)

# if a checkpoint exists, restore the latest checkpoint.
if ckpt_manager.latest_checkpoint:
  ckpt.restore(ckpt_manager.latest_checkpoint)
  print('Latest checkpoint restored!!')

EPOCHS = 20
train_step_signature = [
  tf.TensorSpec(shape=(None, None), dtype=tf.int64),
  tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]
@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
  tar_inp = tar[:, :-1]
  tar_real = tar[:, 1:]
  with tf.GradientTape() as tape:
    predictions, _ = transformer([inp, tar_inp], training = True)
    loss = loss_function(tar_real, predictions)

```

```
    gradients = tape.gradient(loss, transformer.trainable_variables)
    optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))

    train_loss(loss)
    train_accuracy(accuracy_function(tar_real, predictions))

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

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

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

    def __call__(self, sentence, max_length=20):
        # input sentence is portuguese, hence adding the start and end token
        assert isinstance(sentence, tf.Tensor)
        if len(sentence.shape) == 0:
            sentence = sentence[tf.newaxis]

        sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
        encoder_input = sentence

        # as the target is english, the first token to the transformer should be the
        # english start token.
        start_end = self.tokenizers.en.tokenize([''])[0]
        start = start_end[0][tf.newaxis]
        end = start_end[1][tf.newaxis]

        output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
        output_array = output_array.write(0, start)
```

```
23
24      for i in tf.range(max_length):
25        output = tf.transpose(output_array.stack())
26        predictions, _ = self.transformer([encoder_input, output],
    training=False)
27        predictions = predictions[:, -1:, :] # (batch_size, 1,
    vocab_size)
28        predicted_id = tf.argmax(predictions, axis=-1)
29        output_array = output_array.write(i+1, predicted_id[0])
30        if predicted_id == end:
31          break
32
33      output = tf.transpose(output_array.stack())
34      # output.shape (1, tokens)
35      text = tokenizers.en.detokenize(output)[0]
36      tokens = tokenizers.en.lookup(output)[0]
37      _, attention_weights = self.transformer([encoder_input, output
    [:,:-1]], training=False)
38
39      return text, tokens, attention_weights
40
41  translator = Translator(tokenizers, transformer)
42  sentence = "Eu li sobre triceratops na enciclopédia."
43  translated_text, translated_tokens, attention_weights = translator(tf
    .constant(sentence))
44
45  print(f'{"Prediction":15s}: {translated_text}')
```

Prediction : b'i read about trieps in the encycloody .'

APÊNDICE 9 – BIG DATA

A - ENUNCIADO

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

B - RESOLUÇÃO

Estudo de Caso: A Aplicação de Big Data no Airbnb

Introdução

O Airbnb é uma plataforma online que conecta pessoas que desejam alugar suas casas com viajantes que buscam acomodação. Desde a sua fundação em 2008, a empresa cresceu exponencialmente, operando em mais de 190 países. Esse crescimento é sustentado pela capacidade do Airbnb de coletar, processar e analisar grandes volumes de dados, permitindo uma experiência personalizada para usuários e anfitriões. Neste estudo de caso, analisaremos como os dados do Airbnb podem ser classificados como Big Data, explorando os "Vs"do Big Data presentes em suas operações, e detalharemos as ferramentas utilizadas pela empresa no gerenciamento desses dados.

Classificação dos Dados do Airbnb como Big Data

Os dados gerados e utilizados pelo Airbnb apresentam características que os classificam como Big Data. Os cinco "Vs"do Big Data, Volume, Velocidade, Variedade, Veracidade e Valor, estão presentes de forma evidente nos dados da empresa.

1. Volume: O Airbnb lida com uma quantidade massiva de dados. São milhões de usuários, reservas, avaliações, mensagens e interações diariamente. Esse grande volume requer sistemas capazes de armazenar e processar dados em escala petabyte.

2. Velocidade: Os dados são gerados e precisam ser processados em tempo real ou quase real. Reservas, cancelamentos e atualizações de disponibilidade ocorrem constantemente, exigindo processamento rápido para manter a plataforma atualizada.

3. Variedade: Os dados do Airbnb são diversos, incluindo dados estruturados (informações de reservas, perfis de usuários), semiestruturados (logs de atividades) e não estruturados (comentários, imagens). Essa variedade demanda sistemas flexíveis para gerenciamento.

4. Veracidade: A qualidade e a confiabilidade dos dados são cruciais. O Airbnb precisa garantir que as informações sejam precisas para manter a confiança entre hóspedes e anfitriões.

5. Valor: A análise desses dados gera insights valiosos que permitem melhorar a experiência do usuário, otimizar preços e expandir os negócios. O valor extraído dos dados é fundamental para a estratégia da empresa.

Ferramentas Utilizadas

O Airbnb utiliza uma arquitetura sofisticada de Big Data, incorporando diversas ferramentas e tecnologias para atender às suas necessidades. Abaixo, detalhamos cada ferramenta e seu papel no processo de dados.

1. Amazon Web Services (AWS): S3 (Simple Storage Service): Utilizado para armazenamento de objetos, o S3 permite que o Airbnb armazene grandes volumes de dados de forma escalável e segura. E EC2 (Elastic Compute Cloud): Fornece capacidade computacional elástica, permitindo que a empresa escale seus recursos de processamento conforme a demanda.

2. Hadoop: O Hadoop é um framework de software que facilita o processamento distribuído de grandes conjuntos de dados. No Airbnb, é usado para armazenar e processar dados em clusters de servidores, permitindo a análise de grandes volumes de informação.

3. HBase: Um banco de dados NoSQL que funciona em cima do Hadoop, o HBase é usado para armazenamento em tempo real de dados grandes e dispersos. Ele permite acessos rápidos a dados específicos dentro de conjuntos massivos.

4. Hive: O Hive permite consultas tipo SQL em dados armazenados no Hadoop. Isso facilita para os analistas de dados executarem consultas complexas sem a necessidade de programação em baixo nível.

5. Presto: Presto é um motor de consultas distribuídas que permite ao Airbnb executar consultas SQL interativas e de baixa latência em grandes volumes de dados, espalhados por várias fontes de dados.

6. Apache Spark: O Spark é utilizado para processamento em lote e em tempo real. Com sua capacidade de processamento na memória, o Spark acelera o processamento de tarefas como análise de dados, aprendizado de máquina e processamento de gráficos.

7. Apache Airflow: Desenvolvido pelo próprio Airbnb, o Airflow é uma plataforma de orquestração de fluxos de trabalho que permite a automação de pipelines de dados. Ele gerencia a execução de tarefas, dependências e agendamento de processos.

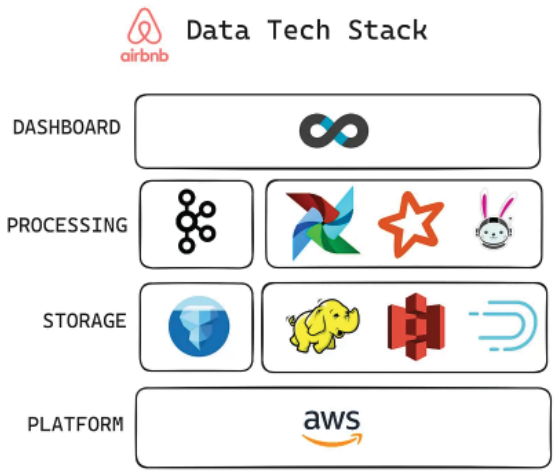
8. Apache Superset: Outra criação do Airbnb, o Superset é uma plataforma de visualização de dados. Permite a criação de dashboards interativos e gráficos, auxiliando na interpretação e análise dos dados.

9. Druid: Druid é um sistema de banco de dados analítico projetado para consultas rápidas em grandes conjuntos de dados. No Airbnb, é usado para análise em tempo real e interativa, especialmente em dados de séries temporais.

Aplicação das Ferramentas

O processo de dados no Airbnb pode ser dividido em várias etapas: coleta, armazenamento, processamento, análise e visualização. A Figura abaixo mostra uma visão geral das ferramentas utilizadas em cada etapa, sendo estas especificadas nas seções seguintes.

Figura 9.1 – VISÃO GERAL DAS FERRAMENTAS UTILIZADAS PELO AIRBNB



1. Coleta: Dados são coletados de diversas fontes, incluindo interações de usuários, informações de reservas, avaliações, logs de atividades e dados de terceiros (como eventos locais e clima).

2. Armazenamento: S3 e Hadoop são utilizados para armazenar os dados coletados. O S3 armazena objetos de dados, enquanto o Hadoop permite armazenamento distribuído e escalável.

3. Processamento: Apache Spark processa dados em lote e em tempo real, permitindo análises complexas e treinamento de modelos de machine learning. Hive e Presto são usados para consultas e manipulação de dados, facilitando o acesso a informações específicas. HBase oferece acesso rápido a dados em tempo real, essencial para aplicações que requerem baixa latência.

4. Análise: Druid permite análises rápidas e interativas, especialmente útil para monitoramento em tempo real e análise de séries temporais. Modelos de machine learning são implementados para diversas finalidades, como previsão de demanda, definição de preços dinâmicos e recomendações personalizadas.

5. Orquestração e Automação: Apache Airflow gerencia e automatiza os fluxos de trabalho de dados, garantindo que os processos sejam executados corretamente e no tempo certo.

6. Visualização: Apache Superset é utilizado para criar dashboards e relatórios que auxiliam na tomada de decisões baseadas em dados.

Conclusão

O Airbnb é um exemplo notável de como Big Data pode ser utilizado para impulsionar negócios digitais. Através da coleta e análise de grandes volumes de dados, a empresa consegue oferecer experiências personalizadas, otimizar operações e inovar continuamente. Os cinco "Vs"do Big Data estão profundamente integrados nas operações do Airbnb, e o uso eficiente de ferramentas como AWS, Hadoop, Spark e outras permite que a empresa transforme dados brutos em insights valiosos. Como estudante de Big Data, observar o caso do Airbnb destaca a importância de uma arquitetura de dados robusta e da integração de diversas tecnologias para lidar com os desafios e oportunidades que o Big Data apresenta.

Referências

https://www.linkedin.com/pulse/what-technology-stack-airbnb-john-doe?utm_source=share utm_medium=member_android&utm_campaign=share_via

https://www.junaideffendi.com/p/airbnb-data-tech-stack

https://vutr.substack.com/p/groupby-40-data-infrastructure-at

https://medium.com/airbnb-engineering/data-infrastructure-at-airbnb-8adfb34f169c

APÊNDICE 10 – VISÃO COMPUTACIONAL

A - ENUNCIADO

Extração de Características

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

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

Tarefas:

a) Carregue a base de dados de Treino.

b) Crie partições contendo 80% para treino e 20% para validação (atenção aos pacientes).

c) Extraia características utilizando LBP e a CNN VGG16 (gerando um csv para cada extrator).

d) Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos.

e) Carregue a base de Teste e execute a tarefa 3 nesta base.

f) Aplique os modelos treinados nos dados de treino

g) Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.

h) Indique qual modelo dá o melhor o resultado e a métrica utilizada

Redes Neurais

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

Tarefas:

a) Utilize a base de dados de Treino já separadas em treino e validação do exercício anterior

b) Treine modelos VGG16 e Resnet50 adaptadas com e sem Data Augmentation

c) Aplique os modelos treinados nas imagens da base de Teste

d) Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.

e) Indique qual modelo dá o melhor o resultado e a métrica utilizada

B - RESOLUÇÃO

```
1
2  import os
3  import cv2
4  import shutil
5  import numpy as np
6  import pandas as pd
7  import matplotlib.pyplot as plt
8
9  from tqdm import tqdm
10 from pathlib import Path
11
12 from sklearn.svm import SVC
13 from sklearn.metrics import recall_score
14 from sklearn.preprocessing import LabelEncoder
15 from sklearn.ensemble import RandomForestClassifier
16 from sklearn.model_selection import train_test_split
17 from sklearn.metrics import classification_report, confusion_matrix
18
19 from tensorflow.keras.layers import Dense
20 from tensorflow.keras.models import Model
21 from tensorflow.keras.models import Sequential
22 from tensorflow.keras.applications import VGG16
23 from tensorflow.keras.preprocessing import image
24 from tensorflow.keras.applications.vgg16 import preprocess_input
25
26 !unzip Train_Warwick.zip
27
28 # Diretório das amostras de treino
29 images_dir = Path('/content/Train_4cls_amostra')
30
31 # Diretórios de destino para treino, validação e teste
32 train_dir = images_dir.parent / 'train'
33 val_dir = images_dir.parent / 'val'
34
35 # Classes disponíveis
36 classes = ['0', '1', '2', '3']
37
38 patient_dict = {}
39
40 # Percorrer cada classe e organizar pacientes
41 for cls in classes:
42     class_dir = images_dir / cls
43     for img_name in os.listdir(class_dir):
44         if img_name.endswith('.png'):
45             # Extrair o ID do paciente do nome do arquivo (
    XX_HER_YYYY.png)
46             patient_id = img_name.split('_')[0]
```

```
47              img_path = class_dir / img_name
48              if cls not in patient_dict:
49                  patient_dict[cls] = {}
50              if patient_id not in patient_dict[cls]:
51                  patient_dict[cls][patient_id] = []
52              patient_dict[cls][patient_id].append((img_path, cls))
53
54  # Função para dividir pacientes em 80% treino e 20% validação
55  def split_patients(patients):
56      patient_ids = list(patients.keys())
57      train_size = int(0.8 * len(patient_ids))
58      train_patient_ids = patient_ids[:train_size]
59      val_patient_ids = patient_ids[train_size:]
60      return train_patient_ids, val_patient_ids
61
62  # Função para mover as imagens para os diretórios de treino e validaç
        ão
63  def move_images(patient_ids, patient_dict, target_dir):
64      for pid in patient_ids:
65          for img_path, cls in patient_dict[pid]:
66              # Diretório da classe no conjunto alvo
67              class_dir = target_dir / cls
68              class_dir.mkdir(parents=True, exist_ok=True)
69              # Caminho de destino da imagem
70              dest_path = class_dir / img_path.name
71              # Mover a imagem
72              shutil.move(str(img_path), str(dest_path))
73
74  # Para cada classe, dividir pacientes e mover as imagens
75  for cls in classes:
76      print(f"Processando classe {cls}...")
77
78      # Obter os pacientes dessa classe
79      patients_in_class = patient_dict[cls]
80
81      # Dividir em treino e validação
82      train_patient_ids, val_patient_ids = split_patients(
        patients_in_class)
83
84      print('Número total de pacientes:', len(train_patient_ids) + len(
        val_patient_ids))
85      print('Número de pacientes para treino:', len(train_patient_ids))
86      print('Número de pacientes para validação:', len(val_patient_ids)
        )
87
88      # Mover as imagens de treino
89      move_images(train_patient_ids, patients_in_class, train_dir)
90
91      # Mover as imagens de validação
92      move_images(val_patient_ids, patients_in_class, val_dir)
93
```

```
print("Divisão de treino e validação concluída.")

# Processando classe 0...
# Número total de pacientes: 5
# Número de pacientes para treino: 4
# Número de pacientes para validação: 1
# Processando classe 1...
# Número total de pacientes: 5
# Número de pacientes para treino: 4
# Número de pacientes para validação: 1
# Processando classe 2...
# Número total de pacientes: 5
# Número de pacientes para treino: 4
# Número de pacientes para validação: 1
# Processando classe 3...
# Número total de pacientes: 5
# Número de pacientes para treino: 4
# Número de pacientes para validação: 1
# Divisão de treino e validação concluída.

def get_pixel(img, center, x, y):

        new_value = 0

        try:
                # If local neighbourhood pixel
                # value is greater than or equal
                # to center pixel values then
                # set it to 1
                if img[x][y] >= center:
                        new_value = 1

        except:
                # Exception is required when
                # neighbourhood value of a center
                # pixel value is null i.e. values
                # present at boundaries.
                pass

        return new_value


# Function for calculating LBP
def lbp_calculated_pixel(img, x, y, radius):

        center = img[x][y]

        val_ar = []

        # top_left
        val_ar.append(get_pixel(img, center, x-radius, y-radius))
```

```
145
146         # top
147         val_ar.append(get_pixel(img, center, x-radius, y))
148
149         # top_right
150         val_ar.append(get_pixel(img, center, x-radius, y + radius))
151
152         # right
153         val_ar.append(get_pixel(img, center, x, y + radius))
154
155         # bottom_right
156         val_ar.append(get_pixel(img, center, x + radius, y + radius))
157
158         # bottom
159         val_ar.append(get_pixel(img, center, x + radius, y))
160
161         # bottom_left
162         val_ar.append(get_pixel(img, center, x + radius, y-radius))
163
164         # left
165         val_ar.append(get_pixel(img, center, x, y-radius))
166
167         # Now, we need to convert binary
168         # values to decimal
169         power_val = [1, 2, 4, 8, 16, 32, 64, 128]
170
171         val = 0
172
173         for i in range(len(val_ar)):
174                 val += val_ar[i] * power_val[i]
175
176         return val
177
178 # Função para calcular o LBP
179 def extract_lbp_image(image):
180     img_gray = cv2.cvtColor(image,
181                 cv2.COLOR_BGR2GRAY)
182     img_gray = cv2.bilateralFilter(img_gray, 7, 75, 75)
183
184     height, width, _ = image.shape
185
186     img_lbp = np.zeros((height, width),
187             np.uint8)
188
189     for i in range(0, height):
190       for j in range(0, width):
191         img_lbp[i, j] = lbp_calculated_pixel(img_gray, i, j, 1)
192
193     # Calcular o histograma
194     h = cv2.calcHist([img_lbp], [0], None, [256], [0, 256])
195     hist,bins = np.histogram(h.flatten(),256,[0,256])
```

```
196     h_normalized = 100*((hist - hist.min()) /(hist.max() - hist.min()
        ))
197
198     return img_lbp, h_normalized
199
200 def extract_lbp_from_directory(directory, classes):
201     features = []
202     labels = []
203
204     print(f"Extraindo características LBP das imagens no diretório: {
        directory}")
205
206     # Iterar sobre cada classe
207     for cls in classes:
208         class_dir = os.path.join(directory, cls)
209         image_names = os.listdir(class_dir)
210         for img_name in tqdm(image_names, desc=f"Classe {cls}"):
211             img_path = os.path.join(class_dir, img_name)
212             # Ler a imagem
213             image = cv2.imread(img_path, 1)
214             # Verificar se a imagem tem 4 canais
215             if len(image.shape) == 3 and image.shape[2] == 4:
216                 # Remover o canal alfa
217                 image = image[:, :, :3]
218             # Extrair características LBP
219             img_lbp, h_normalized = extract_lbp_image(image)
220             features.append(h_normalized)
221             # Armazenar o label
222             labels.append(cls)
223
224     return features, labels
225
226 # Extrair características do conjunto de treino
227 train_features, train_labels = extract_lbp_from_directory(train_dir,
        classes)
228
229 # Extrair características do conjunto de validação
230 val_features, val_labels = extract_lbp_from_directory(val_dir,
        classes)
231
232 # Extraindo características LBP das imagens no diretório: /content/
        val
233
234 # Converter as listas em arrays numpy
235 train_features = np.array(train_features)
236 train_labels = np.array(train_labels)
237 val_features = np.array(val_features)
238 val_labels = np.array(val_labels)
239
240 # Criar DataFrames
241 train_df = pd.DataFrame(train_features)
```

```
train_df['label'] = train_labels

val_df = pd.DataFrame(val_features)
val_df['label'] = val_labels

# Salvar em CSV
train_df.to_csv('train_lbp_features.csv', index=False)
val_df.to_csv('val_lbp_features.csv', index=False)

print("Características LBP salvas nos arquivos 'train_lbp_features.csv' e 'val_lbp_features.csv'")

# Características LBP salvas nos arquivos 'train_lbp_features.csv' e 'val_lbp_features.csv'

# Carregar o modelo VGG16 pré-treinado, sem a camada de classificação final
# pooling='avg' para obter um vetor de características de saída
base_model = VGG16(weights='imagenet', include_top=False, pooling='avg')

# Função para carregar uma imagem, redimensioná-la e pré-processá-la
def process_image(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)  # Pré-processamento VGG16
    return img_array

# Função para extrair as características usando o modelo VGG16
def extract_vgg16_features(img_path, model):
    img_array = process_image(img_path)
    features = model.predict(img_array)
    return features.flatten()  # Converter para vetor de características

  # Extrair características de um diretório de imagens
def extract_features_from_directory(directory, model):
    features = []
    labels = []

    for class_label in os.listdir(directory):
        class_dir = os.path.join(directory, class_label)

        if os.path.isdir(class_dir):
            for img_name in tqdm(os.listdir(class_dir), desc=f"Classe {class_label}"):
                img_path = os.path.join(class_dir, img_name)

                # Extrair características
```

```
286                 img_features = extract_vgg16_features(img_path, model
    )
287
288                 # Armazenar as características e o rótulo (classe)
289                 features.append(img_features)
290                 labels.append(class_label)
291
292     return np.array(features), np.array(labels)
293
294 # Extração das características das imagens de treino usando o modelo
    VGG16
295 print("Extraindo características do conjunto de treino usando o
    modelo VGG16")
296 train_features, train_labels = extract_features_from_directory(
    train_dir, base_model)
297
298 # Extração das características das imagens de validação usando o
    modelo VGG16
299 print("Extraindo características do conjunto de validação usando o
    modelo VGG16")
300 val_features, val_labels = extract_features_from_directory(val_dir,
    base_model)
301
302
303 # Converter para DataFrame e salvar como CSV
304 train_df = pd.DataFrame(train_features)
305 train_df['label'] = train_labels
306 train_df.to_csv('train_vgg16_features.csv', index=False)
307
308 val_df = pd.DataFrame(val_features)
309 val_df['label'] = val_labels
310 val_df.to_csv('val_vgg16_features.csv', index=False)
311
312 print("Características VGG16 salvas nos arquivos '
    train_vgg16_features.csv' e 'val_vgg16_features.csv'")
313
314 # Carregar características LBP e CNN VGG16
315 train_lbp = pd.read_csv('train_lbp_features.csv')
316 train_vgg16 = pd.read_csv('train_vgg16_features.csv')
317
318 # Separar características e rótulos (labels)
319 X_train_lbp = train_lbp.drop(columns=['label'])
320 y_train_lbp = train_lbp['label']
321
322 X_train_vgg16 = train_vgg16.drop(columns=['label'])
323 y_train_vgg16 = train_vgg16['label']
324
325 # Codificar as labels para valores numéricos
326 label_encoder = LabelEncoder()
327 y_train_lbp = label_encoder.fit_transform(y_train_lbp)
328 y_train_vgg16 = label_encoder.fit_transform(y_train_vgg16)
```

```
329
330 # Treinando o modelo SVM com características LBP
331 svm_lbp = SVC(kernel='linear', random_state=42)
332 svm_lbp.fit(X_train_lbp, y_train_lbp)
333
334 # Treinando o modelo SVM com características VGG16
335 svm_vgg16 = SVC(kernel='linear', random_state=42)
336 svm_vgg16.fit(X_train_vgg16, y_train_vgg16)
337
338 # Função para criar o modelo RNA
339 def create_rna_model(input_dim):
340     model = Sequential()
341     model.add(Dense(128, activation='relu', input_dim=input_dim))
342     model.add(Dense(64, activation='relu'))
343     model.add(Dense(32, activation='relu'))
344     model.add(Dense(len(label_encoder.classes_), activation='softmax'
    ))  # Saída com número de classes
345     model.compile(optimizer='adam', loss='
    sparse_categorical_crossentropy', metrics=['accuracy'])
346     return model
347
348 # Treinando a RNA com características LBP
349 rna_lbp = create_rna_model(X_train_lbp.shape[1])
350 rna_lbp.fit(X_train_lbp, y_train_lbp, epochs=10, batch_size=32,
    validation_split=0.2)
351
352 # Treinando a RNA com características VGG16
353 rna_vgg16 = create_rna_model(X_train_vgg16.shape[1])
354 rna_vgg16.fit(X_train_vgg16, y_train_vgg16, epochs=10, batch_size=32,
    validation_split=0.2)
355
356 # Avaliando Random Forest (LBP)
357 y_pred_rf_lbp = rf_lbp.predict(X_train_lbp)
358 print("Relatório de Classificação (Random Forest - LBP):")
359 print(classification_report(y_train_lbp, y_pred_rf_lbp))
360
361 # Avaliando SVM (LBP)
362 y_pred_svm_lbp = svm_lbp.predict(X_train_lbp)
363 print("Relatório de Classificação (SVM - LBP):")
364 print(classification_report(y_train_lbp, y_pred_svm_lbp))
365
366 # Avaliando RNA (LBP)
367 y_pred_rna_lbp = rna_lbp.predict(X_train_lbp)
368 y_pred_rna_lbp = np.argmax(y_pred_rna_lbp, axis=1)
369 print("Relatório de Classificação (RNA - LBP):")
370 print(classification_report(y_train_lbp, y_pred_rna_lbp))
371
372 # Avaliando Random Forest (VGG16)
373 y_pred_rf_vgg16 = rf_vgg16.predict(X_train_vgg16)
374 print("Relatório de Classificação (Random Forest - VGG16):")
375 print(classification_report(y_train_vgg16, y_pred_rf_vgg16))
```

```
376
377 # Avaliando SVM (VGG16)
378 y_pred_svm_vgg16 = svm_vgg16.predict(X_train_vgg16)
379 print("Relatório de Classificação (SVM - VGG16):")
380 print(classification_report(y_train_vgg16, y_pred_svm_vgg16))
381
382 # Avaliando RNA (VGG16)
383 y_pred_rna_vgg16 = rna_vgg16.predict(X_train_vgg16)
384 y_pred_rna_vgg16 = np.argmax(y_pred_rna_vgg16, axis=1)
385 print("Relatório de Classificação (RNA - VGG16):")
386 print(classification_report(y_train_vgg16, y_pred_rna_vgg16))
387
388 #Relatório de Classificação (Random Forest - LBP):
389 #               precision    recall  f1-score   support
390 #
391 #            0       1.00      1.00      1.00       116
392 #            1       1.00      1.00      1.00       117
393 #            2       1.00      1.00      1.00       120
394 #            3       1.00      1.00      1.00       120
395
396 #    accuracy                           1.00       473
397 #   macro avg       1.00      1.00      1.00       473
398 #weighted avg       1.00      1.00      1.00       473
399
400 #Relatório de Classificação (SVM - LBP):
401 #               precision    recall  f1-score   support
402 #
403 #            0       1.00      1.00      1.00       116
404 #            1       1.00      1.00      1.00       117
405 #            2       1.00      1.00      1.00       120
406 #            3       1.00      1.00      1.00       120
407
408 #    accuracy                           1.00       473
409 #   macro avg       1.00      1.00      1.00       473
410 #weighted avg       1.00      1.00      1.00       473
411
412 #Relatório de Classificação (RNA - LBP):
413 #               precision    recall  f1-score   support
414
415 #            0       0.76      0.73      0.75       116
416 #            1       0.71      0.89      0.79       117
417 #            2       0.43      0.74      0.54       120
418 #            3       1.00      0.07      0.12       120
419 #
420 #    accuracy                           0.60       473
421 #   macro avg       0.73      0.61      0.55       473
422 # weighted avg       0.73      0.60      0.55       473
423
424 #Relatório de Classificação (Random Forest - VGG16):
425 #               precision    recall  f1-score   support
426 #
```

```
427 #           0       1.00      1.00      1.00       116
428 #           1       1.00      1.00      1.00       117
429 #           2       1.00      1.00      1.00       120
430 #           3       1.00      1.00      1.00       120
431 
432 #    accuracy                           1.00       473
433 #   macro avg       1.00      1.00      1.00       473
434 #weighted avg       1.00      1.00      1.00       473
435 
436 # Relatório de Classificação (SVM - VGG16):
437 #               precision    recall  f1-score   support
438 
439 #           0       1.00      1.00      1.00       116
440 #           1       1.00      1.00      1.00       117
441 #           2       1.00      1.00      1.00       120
442 #           3       1.00      1.00      1.00       120
443 
444 #    accuracy                           1.00       473
445 #   macro avg       1.00      1.00      1.00       473
446 #weighted avg       1.00      1.00      1.00       473
447 
448 #Relatório de Classificação (RNA - VGG16):
449 #               precision    recall  f1-score   support
450 
451 #           0       0.98      1.00      0.99       116
452 #           1       0.92      1.00      0.96       117
453 #           2       1.00      0.85      0.92       120
454 #           3       0.95      1.00      0.98       120
455 
456 #    accuracy                           0.96       473
457 #   macro avg       0.96      0.96      0.96       473
458 #weighted avg       0.96      0.96      0.96       473
459 
460 # !unzip Test_Warwick.zip
461 
462 # Diretório das amostras de teste
463 test_dir = Path('/content/Test_4cl_amostra')
464 
465 # Extrair características do conjunto de teste
466 test_features, test_labels = extract_lbp_from_directory(test_dir,
        classes)
467 
468 # Converter as listas em arrays numpy
469 test_features = np.array(test_features)
470 test_labels = np.array(test_labels)
471 
472 # Criar DataFrame
473 test_df = pd.DataFrame(test_features)
474 test_df['label'] = test_labels
475 
476 # Salvar em CSV
```

```
test_df.to_csv('test_lbp_features.csv', index=False)

print("Características LBP salvas no arquivo 'test_lbp_features.csv'")

# Extração das características das imagens de teste
print("Extraindo características do conjunto de teste...")
test_features, test_labels = extract_features_from_directory(test_dir, base_model)

# Converter para DataFrame e salvar como CSV
test_df = pd.DataFrame(test_features)
test_df['label'] = test_labels
test_df.to_csv('test_vgg16_features.csv', index=False)

print("Características VGG16 salvas nos arquivos 'test_vgg16_features.csv'")

# Carregar os dados de teste
test_lbp = pd.read_csv('test_lbp_features.csv')
test_vgg16 = pd.read_csv('test_vgg16_features.csv')

# Separar as características e os rótulos (labels)
X_test_lbp = test_lbp.drop(columns=['label'])
y_test_lbp = test_lbp['label']

X_test_vgg16 = test_vgg16.drop(columns=['label'])
y_test_vgg16 = test_vgg16['label']

# Codificar os rótulos de teste (mesma codificação usada para os dados de treino)
label_encoder = LabelEncoder()
y_test_lbp = label_encoder.fit_transform(y_test_lbp)

label_encoder = LabelEncoder()
y_test_vgg16 = label_encoder.fit_transform(y_test_vgg16)

# Fazer previsões com o modelo Random Forest
y_pred_rf_lbp = rf_lbp.predict(X_test_lbp)

y_pred_rf_vgg16 = rf_vgg16.predict(X_test_vgg16)

# Fazer previsões com o modelo SVM
y_pred_svm_lbp = svm_lbp.predict(X_test_lbp)

y_pred_svm_vgg16 = svm_vgg16.predict(X_test_vgg16)

# Fazer previsões com a RNA
y_pred_rna_lbp = rna_lbp.predict(X_test_lbp)
y_pred_rna_lbp = np.argmax(y_pred_rna_lbp, axis=1)

```

```
y_pred_rna_vgg16 = rna_vgg16.predict(X_test_vgg16)
y_pred_rna_vgg16 = np.argmax(y_pred_rna_vgg16, axis=1)

# Calcular Especificidade a partir da Matriz de Confusão
def calculate_specificity(conf_matrix):
    TN = conf_matrix[0, 0]
    FP = conf_matrix[0, 1]
    specificity = TN / (TN + FP)
    return specificity

print("Random Forest\n")

# Avaliar o desempenho do Random Forest (LBP)
print("LBP")
print(classification_report(y_test_lbp, y_pred_rf_lbp))
conf_matrix_rf_lbp = confusion_matrix(y_test_lbp, y_pred_rf_lbp)
specificity_rf_lbp = calculate_specificity(conf_matrix_rf_lbp)
print(f"Especificidade: {specificity_rf_lbp}")

# Avaliar o desempenho do Random Forest (VGG16)
print("\n\nVGG16")
print(classification_report(y_test_vgg16, y_pred_rf_vgg16))
conf_matrix_rf_vgg16 = confusion_matrix(y_test_vgg16, y_pred_rf_vgg16
    )
specificity_rf_vgg16 = calculate_specificity(conf_matrix_rf_vgg16)
print(f"Especificidade: {specificity_rf_vgg16}")


#Random Forest

#LBP
#              precision    recall  f1-score   support
#
#           0       0.38      0.50      0.43       101
#           1       0.29      0.27      0.28        90
#           2       0.43      0.29      0.35        90
#           3       0.77      0.82      0.80        90

#    accuracy                           0.47       371
#   macro avg       0.47      0.47      0.46       371
#weighted avg       0.47      0.47      0.46       371

#Especificidade: 0.6493506493506493


#VGG16
#              precision    recall  f1-score   support

#           0       0.80      0.99      0.88       101
#           1       0.58      0.17      0.26        90
#           2       0.59      0.83      0.69        90
```

```
#           3       0.95      0.98      0.96         90

#    accuracy                           0.75        371
#   macro avg       0.73      0.74      0.70        371
#weighted avg       0.73      0.75      0.70        371

#Especificidade: 0.9900990099009901

print("SVM\n")

# Avaliar o desempenho do SVM (LBP)
print("LBP")
print(classification_report(y_test_lbp, y_pred_svm_lbp))
conf_matrix_svm_lbp = confusion_matrix(y_test_lbp, y_pred_svm_lbp)
specificity_svm_lbp = calculate_specificity(conf_matrix_svm_lbp)
print(f"Especificidade: {specificity_svm_lbp}")

# Avaliar o desempenho do SVM (LBP)
print("\n\nVGG16")
print(classification_report(y_test_vgg16, y_pred_svm_vgg16))
conf_matrix_svm_vgg16 = confusion_matrix(y_test_vgg16,
    y_pred_svm_vgg16)
specificity_svm_vgg16 = calculate_specificity(conf_matrix_svm_vgg16)
print(f"Especificidade: {specificity_svm_vgg16}")

#SVM

#LBP
#              precision    recall  f1-score   support

#           0       0.38      0.50      0.43       101
#           1       0.35      0.32      0.34        90
#           2       0.50      0.43      0.46        90
#           3       0.74      0.63      0.68        90

#    accuracy                           0.47        371
#   macro avg       0.49      0.47      0.48        371
#weighted avg       0.49      0.47      0.48        371

#Especificidade: 0.6538461538461539

#VGG16
#              precision    recall  f1-score   support

#           0       0.85      0.92      0.88       101
#           1       0.77      0.30      0.43        90
#           2       0.64      0.96      0.77        90
#           3       0.98      1.00      0.99        90

#    accuracy                           0.80        371
#   macro avg       0.81      0.79      0.77        371
```

```
#weighted avg       0.81      0.80      0.77       371

#Especificidade: 0.9393939393939394

print("RNA\n")

# Avaliar o desempenho da RNA (LBP)
print("LBP")
print(classification_report(y_test_lbp, y_pred_rna_lbp, zero_division=1))
conf_matrix_rna_lbp = confusion_matrix(y_test_lbp, y_pred_rna_lbp)
specificity_rna_lbp = calculate_specificity(conf_matrix_rna_lbp)
print(f"Especificidade: {specificity_rna_lbp}")

# Avaliar o desempenho da RNA (LBP)
print("\n\nVGG16")
print(classification_report(y_test_vgg16, y_pred_rna_vgg16, zero_division=1))
conf_matrix_rna_vgg16 = confusion_matrix(y_test_vgg16, y_pred_rna_vgg16)
specificity_rna_vgg16 = calculate_specificity(conf_matrix_rna_vgg16)
print(f"Especificidade: {specificity_rna_vgg16}")

#RNA

#LBP
#               precision    recall  f1-score   support

#           0       0.41      0.45      0.43       101
#           1       0.33      0.42      0.37        90
#           2       0.28      0.46      0.35        90
#           3       1.00      0.01      0.02        90

#    accuracy                           0.34       371
#   macro avg       0.51      0.33      0.29       371
#weighted avg       0.50      0.34      0.30       371

#Especificidade: 0.5357142857142857

#VGG16
#               precision    recall  f1-score   support

#           0       0.92      0.95      0.94       101
#           1       0.81      0.58      0.68        90
#           2       0.69      0.73      0.71        90
#           3       0.84      1.00      0.91        90

#    accuracy                           0.82       371
#   macro avg       0.82      0.82      0.81       371
#weighted avg       0.82      0.82      0.81       371

```

```
#Especificidade: 0.9504950495049505

# Resultados: O modelo que deu melhores resultados foi o **RNA**, considerando a métrica de **sensibilidade** (recall).

### RNA - Sensibilidade
#*   CNN VGG16: **0.82**
#*   LBP: **0.34**
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
import os

img_height, img_width = 224, 224
batch_size = 32
num_classes = 4  # Classes: 0, 1, 2, 3

train_datagen = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.vgg16.preprocess_input  # Pré-processamento específico
)

val_datagen = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.vgg16.preprocess_input
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

def create_vgg16_model():
```

```
    # Carregar o modelo base VGG16 sem as camadas totalmente conectadas (top)
    base_model_vgg = VGG16(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))

    # Congelar as camadas convolucionais
    for layer in base_model_vgg.layers:
        layer.trainable = False

    # Adicionar novas camadas
    x = base_model_vgg.output
    x = Flatten()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(num_classes, activation='softmax')(x)

    # Criar o modelo final
    model_vgg = Model(inputs=base_model_vgg.input, outputs=predictions)

    return model_vgg

model_vgg = create_vgg16_model()

model_vgg.compile(optimizer=Adam(learning_rate=1e-4),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

epochs = 5

history_vgg = model_vgg.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)

def create_resnet50_model():
    base_model_resnet = ResNet50(weights='imagenet', include_top=False, input_shape=(img_height, img_width, 3))
    for layer in base_model_resnet.layers:
        layer.trainable = False
    x = base_model_resnet.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    predictions = Dense(num_classes, activation='softmax')(x)
    model_resnet = Model(inputs=base_model_resnet.input, outputs=predictions)
    return model_resnet

model_resnet = create_resnet50_model()
```

```
86
87  model_resnet.compile(optimizer=Adam(learning_rate=1e-4),
88                       loss='categorical_crossentropy',
89                       metrics=['accuracy'])
90
91  history_resnet = model_resnet.fit(
92      train_generator,
93      steps_per_epoch=train_generator.samples // batch_size,
94      epochs=epochs,
95      validation_data=validation_generator,
96      validation_steps=validation_generator.samples // batch_size
97  )
98
99  train_datagen_aug = ImageDataGenerator(
100     preprocessing_function=tf.keras.applications.vgg16.
       preprocess_input,
101     rotation_range=20,
102     width_shift_range=0.1,
103     height_shift_range=0.1,
104     horizontal_flip=True
105 )
106
107 train_generator_aug = train_datagen_aug.flow_from_directory(
108     train_dir,
109     target_size=(img_height, img_width),
110     batch_size=batch_size,
111     class_mode='categorical'
112 )
113
114 # Treinamento com Data Augmentation
115 model_vgg_aug = create_vgg16_model()
116 model_vgg_aug.compile(optimizer=Adam(learning_rate=1e-4),
117                       loss='categorical_crossentropy',
118                       metrics=['accuracy'])
119
120 history_vgg_aug = model_vgg_aug.fit(
121     train_generator_aug,
122     steps_per_epoch=train_generator_aug.samples // batch_size,
123     epochs=epochs,
124     validation_data=validation_generator,
125     validation_steps=validation_generator.samples // batch_size
126 )
127
128 # Treinamento com Data Augmentation
129 model_resnet_aug = create_vgg16_model()
130 model_resnet_aug.compile(optimizer=Adam(learning_rate=1e-4),
131                          loss='categorical_crossentropy',
132                          metrics=['accuracy'])
133
134 history_resnet_aug = model_resnet_aug.fit(
135     train_generator_aug,
```

```
    steps_per_epoch=train_generator_aug.samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size
)

test_datagen = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.vgg16.
    preprocess_input
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(img_height, img_width),
    batch_size=1,  # Para avaliar imagem por imagem
    class_mode='categorical',
    shuffle=False  # Importante para manter a ordem das imagens
)

# VGG sem Data Augmentation
test_generator.reset()
preds_vgg = model_vgg.predict(test_generator, steps=test_generator.
    samples)

# VGG com Data Augmentation
test_generator.reset()
preds_vgg_aug = model_vgg_aug.predict(test_generator, steps=
    test_generator.samples)

# Converter as predições para classes
y_pred_vgg = np.argmax(preds_vgg, axis=1)
y_pred_vgg_aug = np.argmax(preds_vgg_aug, axis=1)
y_pred_resnet = np.argmax(preds_resnet, axis=1)
y_pred_resnet_aug = np.argmax(preds_resnet_aug, axis=1)

# Obter as classes verdadeiras
y_true = test_generator.classes
class_labels = list(test_generator.class_indices.keys())

from sklearn.metrics import confusion_matrix

# VGG16 sem Data Augmentation
cm_vgg = confusion_matrix(y_true, y_pred_vgg)

# VGG16 com Data Augmentation
cm_vgg_aug = confusion_matrix(y_true, y_pred_vgg_aug)

# ResNet50 sem Data Augmentation
cm_resnet = confusion_matrix(y_true, y_pred_resnet)

# ResNet50 com Data Augmentation
```

```
cm_resnet_aug = confusion_matrix(y_true, y_pred_resnet_aug)

from sklearn.metrics import classification_report, confusion_matrix

def calculate_specificity(cm):
    # cm é a matriz de confusão
    FP = cm.sum(axis=0) - np.diag(cm)
    TN = cm.sum() - (FP + cm.sum(axis=1) - np.diag(cm) + np.diag(cm))
    specificity = TN / (TN + FP)
    return specificity

# Avaliar o desempenho do modelo VGG16 sem Data Augmentation
print("VGG16 sem Data Augmentation\n")
print(classification_report(y_true, y_pred_vgg, target_names=
    class_labels))
conf_matrix_vgg = confusion_matrix(y_true, y_pred_vgg)
specificity_vgg = calculate_specificity(conf_matrix_vgg)
print(f"Especificidade: {specificity_vgg}")

# Avaliar o desempenho do modelo VGG16 com Data Augmentation
print("\nVGG16 com Data Augmentation\n")
print(classification_report(y_true, y_pred_vgg_aug, target_names=
    class_labels))
conf_matrix_vgg_aug = confusion_matrix(y_true, y_pred_vgg_aug)
specificity_vgg_aug = calculate_specificity(conf_matrix_vgg_aug)
print(f"Especificidade: {specificity_vgg_aug}")

# Avaliar o desempenho do modelo ResNet50 sem Data Augmentation
print("\nResNet50 sem Data Augmentation\n")
print(classification_report(y_true, y_pred_resnet, target_names=
    class_labels))
conf_matrix_resnet = confusion_matrix(y_true, y_pred_resnet)
specificity_resnet = calculate_specificity(conf_matrix_resnet)
print(f"Especificidade: {specificity_resnet}")

# Avaliar o desempenho do modelo ResNet50 com Data Augmentation
print("\nResNet50 com Data Augmentation\n")
print(classification_report(y_true, y_pred_resnet_aug, target_names=
    class_labels))
conf_matrix_resnet_aug = confusion_matrix(y_true, y_pred_resnet_aug)
specificity_resnet_aug = calculate_specificity(conf_matrix_resnet_aug
    )
print(f"Especificidade: {specificity_resnet_aug}")

#VGG16 sem Data Augmentation

#               precision    recall  f1-score   support

#           0       0.77      0.90      0.83       101
#           1       0.52      0.18      0.26        90
#           2       0.63      0.91      0.75        90
```

```
#           3       0.97      0.99      0.98        90

#    accuracy                           0.75       371
#   macro avg       0.72      0.74      0.70       371
#weighted avg       0.72      0.75      0.71       371

#Especificidade: [0.9        0.94661922 0.82918149 0.98932384]

#VGG16 com Data Augmentation

#              precision    recall  f1-score   support

#           0       0.81      0.99      0.89       101
#           1       0.81      0.29      0.43        90
#           2       0.65      0.84      0.73        90
#           3       0.92      1.00      0.96        90

#    accuracy                           0.79       371
#   macro avg       0.80      0.78      0.75       371
#weighted avg       0.80      0.79      0.76       371

#Especificidade: [0.91111111 0.97864769 0.85409253 0.97153025]

#ResNet50 sem Data Augmentation

#              precision    recall  f1-score   support

#           0       0.99      0.97      0.98       101
#           1       0.93      0.56      0.69        90
#           2       0.69      0.97      0.81        90
#           3       0.98      1.00      0.99        90

#    accuracy                           0.88       371
#   macro avg       0.90      0.87      0.87       371
#weighted avg       0.90      0.88      0.87       371

#Especificidade: [0.9962963  0.98576512 0.86120996 0.99288256]

#ResNet50 com Data Augmentation

#              precision    recall  f1-score   support

#           0       0.75      0.99      0.85       101
#           1       0.58      0.12      0.20        90
#           2       0.61      0.81      0.70        90
#           3       0.91      1.00      0.95        90

#    accuracy                           0.74       371
#   macro avg       0.71      0.73      0.68       371
#weighted avg       0.71      0.74      0.68       371
```

```
281 #Especificidade: [0.87407407 0.97153025 0.83629893 0.96797153]
282
283 # Resultados: O modelo que deu melhores resultados foi o **ResNet50
    sem Data Augmentation**, considerando a métrica de **sensibilidade
    ** (recall).
284
285
286 ### ResNet50 sem Data Augmentation - Sensibilidade
287 #*   **0.88**
```

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A - ENUNCIADO

Título do Trabalho: "Estudo de Caso: Implicações Éticas do Uso do ChatGPT"
Trabalho em Grupo: O trabalho deverá ser realizado em grupo de alunos de no máximo seis (06) integrantes.
Objetivo do Trabalho: Investigar as implicações éticas do uso do ChatGPT em diferentes contextos e propor soluções responsáveis para lidar com esses dilemas.
Parâmetros para elaboração do Trabalho:
1. Relevância Ética: O trabalho deve abordar questões éticas significativas relacionadas ao uso da inteligência artificial, especialmente no contexto do ChatGPT. Os alunos devem identificar dilemas éticos relevantes e explorar como esses dilemas afetam diferentes partes interessadas, como usuários, desenvolvedores e a sociedade em geral.
2. Análise Crítica: Os alunos devem realizar uma análise crítica das implicações éticas do uso do ChatGPT em estudos de caso específicos. Eles devem examinar como o algoritmo pode influenciar a disseminação de informações, a privacidade dos usuários e a tomada de decisões éticas. Além disso, devem considerar possíveis vieses algorítmicos, discriminação e questões de responsabilidade.
3. Soluções Responsáveis: Além de identificar os desafios éticos, os alunos devem propor soluções responsáveis e éticas para lidar com esses dilemas. Isso pode incluir sugestões para políticas, regulamentações ou práticas de design que promovam o uso responsável da inteligência artificial. Eles devem considerar como essas soluções podem equilibrar os interesses de diferentes partes interessadas e promover valores éticos fundamentais, como transparência, justiça e privacidade.
4. Colaboração e Discussão: O trabalho deve envolver discussões em grupo e colaboração entre os alunos. Eles devem compartilhar ideias, debater diferentes pontos de vista e chegar a conclusões informadas através do diálogo e da reflexão mútua. O estudo de caso do ChatGPT pode servir como um ponto de partida para essas discussões, incentivando os alunos a aplicar conceitos éticos e legais aprendidos ao analisar um caso concreto.
5. Limite de Palavras: O trabalho terá um limite de 6 a 10 páginas teria aproximadamente entre 1500 e 3000 palavras.
6. Estruturação Adequada: O trabalho siga uma estrutura adequada, incluindo introdução, desenvolvimento e conclusão. Cada seção deve ocupar uma parte proporcional do total de páginas, com a introdução e a conclusão ocupando menos espaço do que o desenvolvimento.
7. Controle de Informações: Evitar incluir informações desnecessárias que possam aumentar o comprimento do trabalho sem contribuir significativamente para o conteúdo. Concentre-se em informações relevantes, argumentos sólidos e evidências importantes para apoiar sua análise.
8. Síntese e Clareza: O trabalho deverá ser conciso e claro em sua escrita. Evite repetições desnecessárias e redundâncias. Sintetize suas ideias e argumentos de forma eficaz para transmitir suas mensagens de maneira sucinta.
9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: hfps://bibliotecas.ufpr.br/wp- content/uploads/2022/03/template-artigo-de-periodico.docx **B - RESOLUÇÃO**

Estudo de Caso: Implicações Éticas do Uso do ChatGPT
RESUMO

Este estudo de caso investiga as implicações éticas do uso do ChatGPT em diferentes contextos e propõe soluções responsáveis para lidar com esses dilemas. A análise aborda questões éticas significativas, incluindo a disseminação de informações, privacidade dos usuários, vieses algorítmicos, discriminação e

responsabilidade. Soluções responsáveis são propostas para equilibrar os interesses das partes interessadas e promover valores fundamentais como transparência, justiça e privacidade.
Palavras-chave: Inteligência Artificial, Ética, ChatGPT, Privacidade, Vieses.

ABSTRACT This case study investigates the ethical implications of using ChatGPT in different contexts and proposes responsible solutions to address these dilemmas. The analysis addresses significant ethical issues, including information dissemination, user privacy, algorithmic biases, discrimination, and responsibility. Responsible solutions are proposed to balance stakeholder interests and promote fundamental values such as transparency, fairness, and privacy.
Keywords: Artificial Intelligence, Ethics, ChatGPT, Privacy, Biases.

1. INTRODUÇÃO

De acordo com Aruda (2024), em novembro de 2022, houve uma certa euforia e também assombro quando a empresa OpenAI tornou público e acessível a plataforma ChatGPT. Segundo a empresa OpenAI (2022), o ChatGPT é um modelo de inteligência artificial (IA) que interage de maneira conversacional, ou seja, ele foi programado para que haja um diálogo entre o usuário e a máquina, permitindo que a plataforma responda a perguntas, admita erros, conteste premissas incorretas e rejeite solicitações inadequadas.

Com a definição dada pelos próprios criadores da ferramenta, é possível identificar inúmeras lacunas relacionadas a questões éticas provenientes do processo de geração de respostas por meio de IA. Nesse sentido, alguns dilemas abordam problemáticas éticas relacionadas a: (1) privacidade e segurança dos dados; (2) desinformação e uso indevido; (3) viés discriminatório; (4) impactos nas formas de trabalho; (5) responsabilidade e transparência; (6) interações humanas; e, por último, (7) robustez e precisão das informações geradas automaticamente. A discussão acerca de cada item acima gera um cenário multifacetado, com perspectivas provenientes de diversas áreas. No âmbito da jurisprudência, o contexto torna-se ainda mais complexo, exigindo uma compreensão mais aprofundada para que as questões éticas sejam cuidadosamente examinadas.

Neste trabalho abordaremos três questões éticas, sendo elas: (1) privacidade e segurança dos dados; (2) desinformação e uso indevido e (7) robustez e precisão das informações geradas de forma automática. Quando se fala em privacidade dos dados no uso do ChatGPT, é importante ressaltar que a funcionalidade da ferramenta requer um grande volume de dados para o treinamento do modelo da IA por trás dela. Além disso, é necessário que as requisições sejam feitas pelo usuário. Desta forma, fica evidente que os dados são elementos protagonistas e participam de todas as camadas desta aplicação.

No entanto, muitos debates têm sido levantados acerca da responsabilização pela posse dos dados, sejam eles sensíveis ou não, que a plataforma possui. Uma vez que os dados são fornecidos como entrada, eles se tornam parte do sistema. Neste sentido, há abertura de brechas relacionadas ao uso e compartilhamento de dados sensíveis de forma indevida, causando danos quase que irreversíveis aos afetados. Segundo Wu et al. (2024) embora existam mecanismos de proteção de privacidade no ChatGPT, como o bloqueio de acesso a dados pessoais de indivíduos, não é garantido que não ocorra nenhum vazamento.

Aliado a isso, o modelo de IA (LLM - *Large Language Model*, tradução para o português: Modelo de Linguagem de Grande Escala) usado para a geração de informações pelo ChatGPT não é simples de entender, o que torna o processo pouco transparente e suscetível a gerar informações pouco precisas sobre determinado assunto e até informações falsas. Essa falta de transparência pode manipular a opinião pública, causando desinformação e influenciando decisões de forma negativa. Para elucidar estas questões o trabalho discutirá sobre cada um dos itens de forma mais aprofundada trazendo alguns exemplos reais dos impactos gerados pelos dilemas éticos.

2. DESENVOLVIMENTO

A revisão de literatura foca nos estudos prévios relacionados ao uso ético da IA e os desafios associados à implementação destes sistemas. Neste sentido, esta seção tem como objetivo trazer as definições necessárias presentes na literatura e os debates acerca dos impactos gerados por cada um dos dilemas éticos latentes no assunto.

2.1 Privacidade e segurança dos dados

Para Westin (1967): A privacidade na era digital é um direito fundamental que deve ser protegido com rigor. A coleta massiva de dados pessoais sem o consentimento adequado dos usuários representa uma ameaça significativa à autonomia e à dignidade individual.

A privacidade dos usuários é uma questão em evidência neste momento em que a IA está sendo inserida na sociedade e as pessoas começam a conhecer e utilizar. A partir do momento em que interagimos com o ChatGPT, ele armazena todas as informações em sua base de dados para treinamento e isso implica que esses dados se tornarão públicos, a partir do momento em que se faz uma pergunta e a ferramenta entender que aqueles dados inseridos por outra pessoa são os melhores para uma resposta a qualquer usuário. Então, surge a grande preocupação quanto a privacidade dos dados, que dados serão utilizados para a análise, que dados serão apresentados nas respostas, existe algum contato humano com esta base de dados? Não basta somente informar ao usuário de que ele deve evitar inserir dados sensíveis, e sim, o ChatGPT deve utilizar critérios éticos que aperfeiçoem a ferramenta para que ela analise esses dados e gere uma nova conclusão e resposta, considerando os critérios da ética e segurança e dando a eles o máximo de importância.

No estudo apresentado por Khowaja et. al (2024) são discutidas as preocupações acerca da privacidade dos dados usados para o treinamento de LLMs, em especial do ChatGPT. Os autores mencionam que os dados utilizados para o treinamento do ChatGPT são sistematicamente extraídos de mensagens, websites, artigos, livros e informações pessoais sem a devida autorização. Além disso, de acordo com a política de privacidade do ChatGPT (OpenAI, 2024b), são coletadas informações como dados de interação do usuário com o site, configurações e tipo de navegador, e endereço IP, juntamente com o tipo de conteúdo com o qual os usuários interagem no ChatGPT. Eles também coletam informações sobre atividades de navegação em diferentes sites e ao longo de um determinado período de tempo. A política de privacidade também afirma que "Além disso, de tempos em tempos, podemos analisar o comportamento geral e as características dos usuários de nossos serviços e compartilhar informações agregadas, como estatísticas gerais de usuários, com terceiros, publicar essas informações agregadas ou torná-las geralmente disponíveis. Podemos coletar informações agregadas através dos Serviços, através de cookies e por outros meios descritos nesta política de privacidade"e ainda que "em certas circunstâncias, podemos fornecer suas informações pessoais a terceiros sem aviso prévio, a menos que exigido por lei".

2.2 Desinformação e uso indevido

Segundo Wardle e Derakhshan (2017): A desinformação, especialmente em plataformas digitais, pode ter consequências graves para a sociedade. Ela pode manipular opiniões, influenciar decisões eleitorais e até mesmo causar pânico em situações de crise.

Atualmente, a internet é a principal fonte de informação. Desta forma, o desafio não está apenas em obter um conteúdo relevante, mas também em filtrar as informações incorretas da grande quantidade disponível. Antes do lançamento do ChatGPT, as pessoas utilizavam outros métodos para confirmar a veracidade das informações, avaliando o nível de conhecimento do criador do conteúdo pelo rigor da linguagem, correção do formato e a fonte do texto, sendo esses indicadores de confiabilidade. No entanto, ao utilizar o ChatGPT, é possível observar que esses indicadores são excelentes nesses aspectos, dando a falsa sensação de confiabilidade. Os usuários podem cair na armadilha de confiar cegamente no conteúdo gerado pelo ChatGPT depois de executar testes simples. Essa confiança cega pode levar a julgamentos errados devido a hábitos arraigados. Em áreas críticas, como documentos de pesquisa médica, informações básicas de experimentos em larga escala e conteúdo de políticas, fazer referência a informações errôneas do ChatGPT e tirar conclusões incorretas pode ter consequências imprevisíveis. Esses riscos não decorrem de fraude intencional, mas sim dos erros do ChatGPT, apesar de sua aparência inicialmente confiável (Wu et al., 2024).

Outro aspecto muito importante relacionado ao uso do ChatGPT é a geração de notícias falsas, que têm implicações éticas de bastante relevância social e política e já são um dos principais problemas que vêm sendo enfrentados no Brasil. Estes Modelos de Linguagem Grande (LLMs) são muito populares e podem ser facilmente acessados para a geração de informações falsas ou enganosas que podem facilmente serem disseminadas nas redes sociais e plataforma digitais, podendo ter com consequências graves, como

por exemplo a manipulação de eleições e a incitação à violências (Khowaja et. al, 2024). Até que ponto as informações geradas pelo ChatGPT são verdadeiras? Esse é um grande desafio, pelo motivo de que em muitos momentos as informações necessitam ser validadas, muitas vezes depende de decisões que ainda precisam ser formuladas e votadas pela sociedade, categorias e governos. O fato de que as informações geradas pelo ChatGPT são originadas de uma base de dados e esta base foi construída pelo próprio ChatGPT, leva a conclusão de que esta base de dados deve ser constantemente validada, se possível por um órgão neutro externo que controle a legitimidade das informações e as distribua de forma a guardar a ética e os direitos de todas as partes envolvidas. Atualmente existe a versão gratuita e a versão paga do ChatGPT, e a principal diferença entre elas é a atualização da base de dados e funcionalidades. Já foi constatado que em determinados questionamentos ao ChatGPT, a ferramenta informou erroneamente e isso é muito sério, pois, a maior fatia de utilização é pela opção gratuita, que é a com a base de dados desatualizada e que fornece grandes vulnerabilidades. Uma regulamentação que defina a forma que esses dados são validados é de fundamental importância para garantir a segurança e confiabilidade das informações.

2.3 Robustez e precisão das informações geradas

Para O'Neil (2016): A automação na tomada de decisões promete eficiência e objetividade, mas também apresenta riscos consideráveis. Algoritmos podem perpetuar vieses existentes nos dados de treinamento, levando a decisões injustas ou discriminatórias.

Outro desafio é a tomada de decisões automatizadas, que dá poder para a ferramenta decidir o que fazer com base nas regras que foram criadas por seus programadores. Um grande exemplo são as IAs de recrutamento e seleção, que fazem a triagem com base em critérios definidos e podem muitas vezes serem injustas. Com relação ao ChatGPT, eu posso desde solicitar a criação de um código em determinada linguagem que diante de um perfil pré-selecionado busque a opção ideal em sua base, posso pedir para que ele gere uma imagem como por exemplo pedir uma imagem de uma pessoa bonita, e ele irá criar uma imagem de uma pessoa branca, loira, de olhos claros, ou seja, seguindo critérios já definidos pelos programadores ou seguindo uma tendência das imagens buscadas na internet. Deixar que a IA faça a escolha sozinha oferece riscos, pois ela foi configurada para buscar a melhor solução em sua base de dados e ela pode estar repleta de vieses.

3. CONCLUSÃO

Através de discussões em grupo e colaboração entre os membros da equipe, chegamos a algumas propostas para lidar com as implicações éticas do uso do ChatGPT. Nossa análise destacou a importância de criar regulamentações claras para garantir que o uso da IA respeite a privacidade dos usuários e evite a disseminação de informações falsas. Essas regulamentações devem definir padrões rigorosos para a coleta, armazenamento e uso dos dados, assegurando a proteção das informações pessoais.

Além disso, identificamos a necessidade de promover a transparência nos processos de decisão do ChatGPT. Isso pode ser alcançado através da implementação de mecanismos de auditoria e monitoramento contínuo, permitindo verificar e corrigir vieses algorítmicos, reduzindo o risco de discriminação.

Durante nossas discussões, também enfatizamos a importância de práticas de design ético na criação e atualização dos modelos de IA. Envolver equipes multidisciplinares e incluir feedback de usuários e especialistas externos pode contribuir para a criação de um sistema mais robusto e justo.

Finalmente, destacamos a necessidade de educar os usuários sobre os potenciais riscos e limitações do ChatGPT. Programas de conscientização e treinamentos específicos podem capacitar os usuários a utilizarem a IA de forma crítica e responsável, reconhecendo a importância da verificação das informações fornecidas pela ferramenta.

Em resumo, através da colaboração entre os membros da equipe e discussões aprofundadas, conseguimos desenvolver soluções práticas para lidar com as implicações éticas do uso do ChatGPT. As soluções propostas incluem regulamentações claras, transparência nos processos de decisão, práticas de design ético e educação do usuário, promovendo um uso mais seguro e justo dessa tecnologia.

4. REFERÊNCIAS

Aruda, E. P. Inteligência Artificial Generativa No Contexto Da Transformação do Trabalho Docente. 2024. Disponível em: <https://doi.org/10.1590/0102-469848078>.

Deng, J.; Lin, Y. The benefits and challenges of ChatGPT: An overview. Frontiers in Computing and Intelligent Systems, v. 2, n. 2, p. 81-83, 2022.
Khowaja, S. A.; Khuwaja, P.; Dev, K.; Wang, W.; Nkenyereye, L. Chatgpt needs spade (sustainability, privacy, digital divide, and ethics) evaluation: A review. Cognitive Computation, p. 1-23, 2024.
O'Neil, C. Armas de Destruição Matemática: Como o Big Data Aumenta a Desigualdade e Ameaça a Democracia. Nova York: Crown Publishing Group, 2016.
OpenAI [internet]. Introducing ChatGPT. Disponível em: <https://openai.com/index/chatgpt/>. Acessado em 01 de Julho de 2024a.
OpenAi [internet]. Privacy policy. Disponível em: <https://openai.com/policies/privacy-policy/>. Acesso em 8 de Julho de 2024b.
Wardle, C.; Derakhshan, H. Desordem da Informação: Rumo a uma Estrutura Interdisciplinar para Pesquisa e Formulação de Políticas. Conselho da Europa, 2017.
Westin, A. F. Privacidade e Liberdade. Nova York: Atheneum, 1967. Wu, X.; Ran, D.; Jianbing, N. Unveiling security, privacy, and ethical concerns of ChatGPT. Journal of Information and Intelligence, v. 2, n. 2, p. 102-115, 2024.

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A - ENUNCIADO

1 Objetivo

Individualmente, ler e resumir – seguindo o template fornecido – um dos artigos abaixo:
AHMAD, L.; ABDELRAZEK, M.; ARORA, C.; BANO, M; GRUNDY, J. Requirements practices and gaps when engineering human-centered Artificial Intelligence systems. Applied Soft Computing. 143. 2023. DOI https://doi.org/10.1016/j.asoc.2023.110421

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

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

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

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

2 Orientações adicionais

Escolha o artigo que for mais interessante para você. Utilize tradutores e o Chat GPT para entender o conteúdo dos artigos – caso precise, mas escreva o resumo em língua portuguesa e nas suas palavras. Não esqueça de preencher, no trabalho, os campos relativos ao seu nome e ao artigo escolhido. No template, você deverá responder às seguintes questões:

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

Responda cada questão utilizando o espaço fornecido no template, sem alteração do tamanho da fonte (Times New Roman, 10), nem alteração do espaçamento entre linhas (1.0). Não altere as questões do template. Utilize o editor de textos de sua preferência para preencher as respostas, mas entregue o trabalho em PDF.

B - RESOLUÇÃO

Artigo: *Requirements practices and gaps when engineering human-centered Artificial Intelligence systems*

Qual o objetivo do estudo descrito pelo artigo?

O objetivo do estudo é entender como os profissionais estão desenvolvendo sistemas de Inteligência Artificial que são centrados no ser humano. O artigo busca identificar quais práticas de Engenharia de Requisitos estão sendo usadas na indústria para esses sistemas de IA e quais aspectos centrados no ser humano estão sendo considerados. Além disso, pretende descobrir as lacunas entre

as diretrizes existentes, fornecidas por empresas como Google, Microsoft e Apple, e as práticas reais adotadas.

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

O estudo foi realizado porque existe uma necessidade de compreender melhor como os profissionais estão desenvolvendo sistemas de Inteligência Artificial que sejam centrados nas pessoas, especialmente na fase de Engenharia de Requisitos. Embora grandes empresas como Google, Microsoft e Apple tenham criado diretrizes para auxiliar nesse processo, ainda há pouca informação sobre como essas práticas estão sendo aplicadas na indústria. Além disso, há lacunas entre o que a pesquisa acadêmica propõe e o que é realmente feito nas empresas.

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

Os autores utilizaram uma metodologia em duas etapas. Primeiro, mapearam as diretrizes industriais existentes para o desenvolvimento de IA centrada no ser humano e as compararam com estudos acadêmicos sobre Engenharia de Requisitos para IA. Em seguida, realizaram uma pesquisa com profissionais para entender as práticas atuais na indústria em relação à Engenharia de Requisitos para IA e aos aspectos centrados no ser humano considerados.

Quais os principais resultados obtidos pelo estudo?

Os principais resultados obtidos pelo estudo foram que a maioria dos profissionais pesquisados concordou que todos os aspectos centrados no ser humano devem ser considerados durante a fase de Engenharia de Requisitos ao desenvolver sistemas de Inteligência Artificial. Além disso, identificou que muitas das ferramentas e métodos atualmente utilizados, como UML e Microsoft Office, não são adequados para gerenciar requisitos de software baseado em IA. Foram encontradas divergências entre a pesquisa acadêmica e as práticas da indústria na Engenharia de Requisitos para IA. Com base nesses resultados, os autores propuseram cinco recomendações para aprimorar as práticas e ferramentas na Engenharia de Requisitos de sistemas de IA centrados no ser humano.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A - ENUNCIADO

1 Classificação (RNA)

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

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

Informações:

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

```
1    data = tf.keras.datasets.fashion_mnist
2    (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.

```
1
2  url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine
       -quality/winequality-red.csv"
3  data = pd.read_csv(url, delimiter=';')
4
5  Dica 1. Para facilitar o trabalho, renomeie o nome das colunas para
       português, dessa forma:
6
7  data.columns = [
8      'acidez_fixa',             # fixed acidity
9      'acidez_volatil',          # volatile acidity
10     'acido_citrico',           # citric acid
11     'acucar_residual',         # residual sugar
12     'cloretos',                # chlorides
13     'dioxido_de_enxofre_livre', # free sulfur dioxide
14     'dioxido_de_enxofre_total', # total sulfur dioxide
15     'densidade',               # density
16     'pH',                      # pH
17     'sulfatos',                # sulphates
18     'alcool',                  # alcohol
19     'score_qualidade_vinho'                # quality
20 ]
21
22 Dica 2. Separe os dados (x e y) de tal forma que a última coluna (í
       ndice -1), chamada score_qualidade_vinho, seja a variável target (
       y)
```

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

- Imagem onírica obtida por Main Loop;

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

Informações:

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

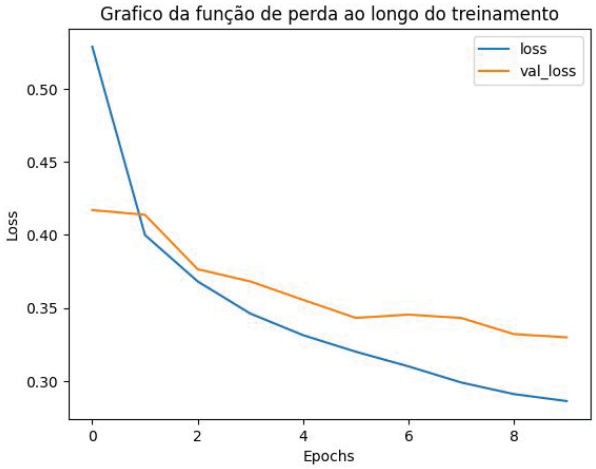
```

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

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

B - RESOLUÇÃO

1 Classificação (RNA)

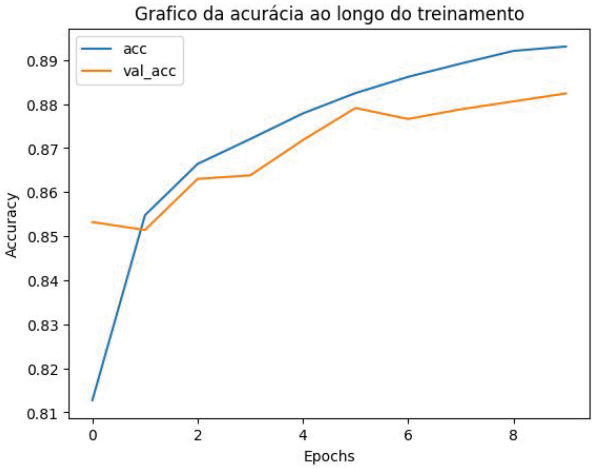
```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

(X_train, Y_train), (X_test, Y_test) = tf.keras.datasets.
    fashion_mnist.load_data()

x_train, x_test = X_train/255.0, X_test/255.0

i = tf.keras.layers.Input(shape=(28,28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation='relu')(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation='softmax')(x)
model = tf.keras.Model(i, x)

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

r = model.fit(x_train, Y_train, validation_data=(x_test, Y_test),
    epochs=10)

plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label='val_loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafico da função de perda ao longo do treinamento')
plt.legend()
```

Figura 13.1 – EXERCÍCIO 1: GRÁFICO DE PERDA



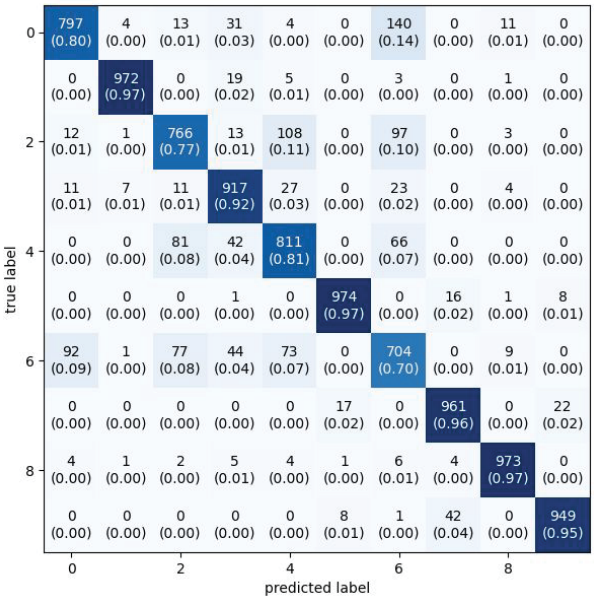

```
plt.plot(r.history['accuracy'], label='acc')
plt.plot(r.history['val_accuracy'], label='val_acc')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Grafico da acurácia ao longo do treinamento')
plt.legend()
```

Figura 13.2 – EXERCÍCIO 1: GRÁFICO DE ACURÁCIA



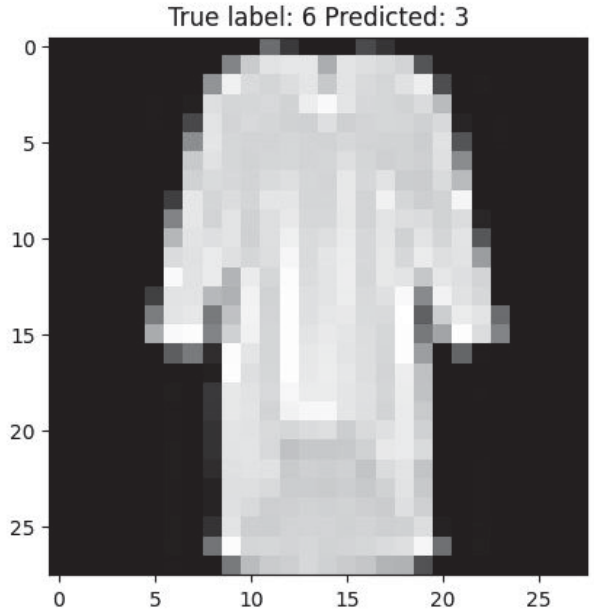

```
y_pred = model.predict(x_test).argmax(axis=1)
cm = confusion_matrix(Y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7,7), show_normed=True)
```

Figura 13.3 – EXERCÍCIO 1: MATRIZ DE CONFUSÃO




```
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" % (Y_test[i], y_pred[i]));
```

Figura 13.4 – EXERCÍCIO 1: APRESENTANDO CLASSIFICAÇÃO INCORRETA



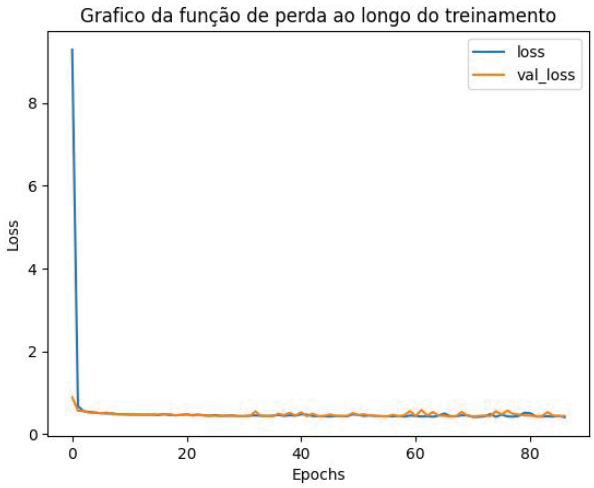
2 Regressão (RNA)

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

```
5  from tensorflow.python.keras import backend
6  from sklearn.model_selection import train_test_split
7  from sklearn.metrics import r2_score, mean_squared_error
8  from math import sqrt
9
10 url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine
       -quality/winequality-red.csv"
11
12 data = pd.read_csv(url, delimiter=';')
13
14 data.columns = [
15 'acidez_fixa', # fixed acidity
16 'acidez_volatil', # volatile acidity
17 'acido_citrico', # citric acid
18 'acucar_residual', # residual sugar
19 'cloretos', # chlorides
20 'dioxido_de_enxofre_livre', # free sulfur dioxide
21 'dioxido_de_enxofre_total', # total sulfur dioxide
22 'densidade', # density
23 'pH', # pH
24 'sulfatos', # sulphates
25 'alcool', # alcohol
26 'score_qualidade_vinho' # quality
27 ]
28
29 data = data.values
30
31 # Dica 2. Separe os dados (x e y) de tal forma que a última coluna (í
       ndice -1), chamada score_qualidade_vinho
32
33 X = data[:, 0:-1].astype(float)
34 Y = data[:, -1].astype(float)
35
36 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size
       =0.25)
37
38 i = tf.keras.layers.Input(shape=(11,))
39 x = tf.keras.layers.Dense(100, activation='relu')(i)
40 x = tf.keras.layers.Dense(1)(x)
41 model = tf.keras.Model(i, x)
42
43 # Criação de funções para as métricas r2 e RMSE serem inseridas no
       modelo
44 def rmse(y_true, y_pred):
45     return backend.sqrt(backend.mean(backend.square(y_true- y_pred),
       axis=-1))
46
47 def r2(y_true, y_pred):
48     media = backend.mean(y_true)
49     num = backend.sum(backend.square(y_true - y_pred))
50     den = backend.sum(backend.square(y_true - media))
```

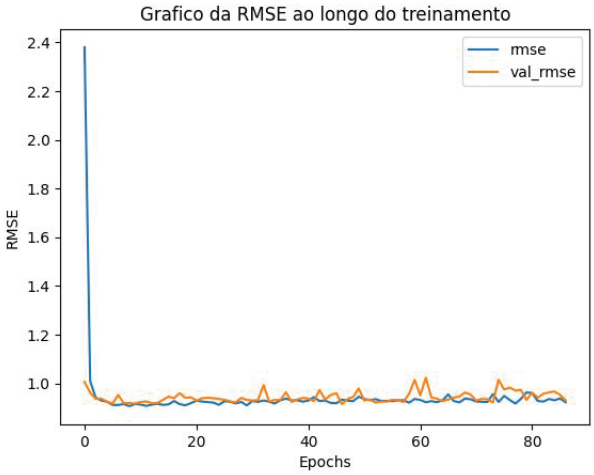
```
    return (1.0 - num/den)

# Compilação
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='mse', metrics=[r2, rmse])

# 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, validation_data=(X_test, Y_test),
    epochs=1500, callbacks=[early_stop])

plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label='val_loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafico da função de perda ao longo do treinamento')
plt.legend()
```

Figura 13.5 – EXERCÍCIO 2: GRÁFICO DE PERDA



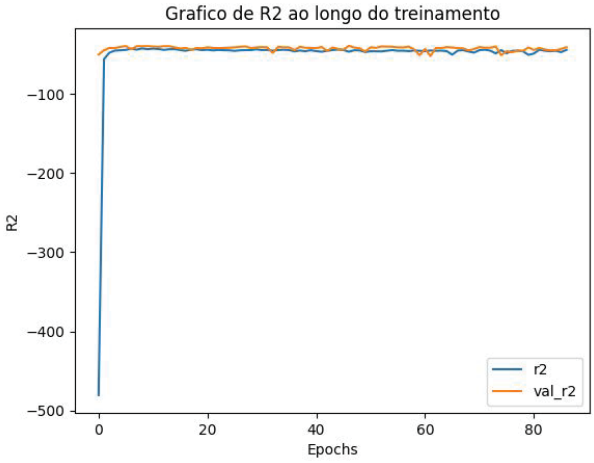

```
plt.plot(r.history['rmse'], label='rmse')
plt.plot(r.history['val_rmse'], label='val_rmse')
plt.xlabel('Epochs')
plt.ylabel('RMSE')
plt.title('Grafico da RMSE ao longo do treinamento')
plt.legend()
```

Figura 13.6 – EXERCÍCIO 2: GRÁFICO DE RMSE




```
plt.plot(r.history['r2'], label='r2')
plt.plot(r.history['val_r2'], label='val_r2')
plt.xlabel('Epochs')
plt.ylabel('R2')
plt.title('Grafico de R2 ao longo do treinamento')
plt.legend()
```

Figura 13.7 – EXERCÍCIO 2: GRÁFICO DE R2




```
y_pred = model.predict(X_test).flatten()
mse = mean_squared_error(Y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(Y_test, y_pred)

print("mse:", mse)
print("rmse:", rmse)
print("r2:", r2)
```

mse: 0.4207298387142231
rmse: 0.6486369082269549

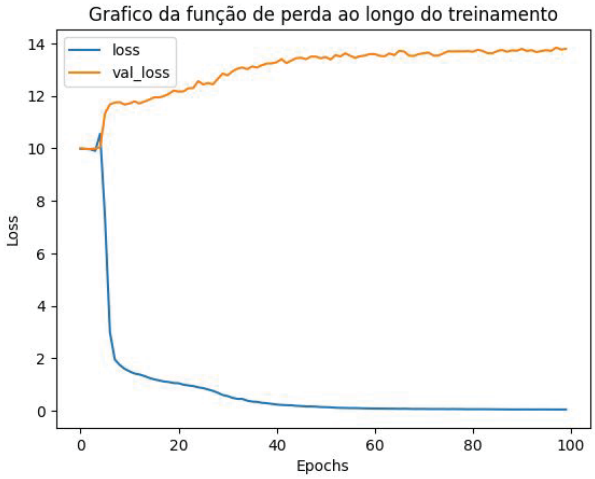
r2: 0.38145588854025325

3 Sistemas de Recomendação

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Dense,
    Embedding, Flatten, Concatenate
from tensorflow.keras.optimizers import SGD, Adam
from sklearn.utils import shuffle

df = pd.read_csv('Base_livros(in).csv', sep=',', on_bad_lines='skip')
df['Notas;;;;;;'] = df['Notas;;;;;;'].str.replace(r'[^0-9]', '',
    regex=True).astype(float)
df.rename(columns={'Notas;;;;;;': 'Notas'}, inplace=True)
df.dropna(inplace=True)

categorical_columns = ['ID_usuario', 'Titulo', 'Autor', 'Editora']
for cat_column in categorical_columns:
    df[cat_column] = pd.Categorical(df[cat_column])
    df['new_' + cat_column] = df[cat_column].cat.codes

N = len(set(df['ID_usuario']))
M = len(set(df['new_Titulo']))

# dimensão do embedding
K = 10

# 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
# filme
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)

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

user_ids, movie_ids, ratings = shuffle(df['new_ID_usuario'], df['
    new_Titulo'], df['Notas'])
Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%
train_user = user_ids[:Ntrain]
train_movie = movie_ids[:Ntrain]
```

```
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_movie = movie_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

epochs = 100
r = model.fit(
x=[train_user, train_movie],
y=train_ratings,
epochs=epochs,
batch_size=1024,
verbose=2, # não imprime o progresso
validation_data=([test_user, test_movie], test_ratings)
)

plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label='val_loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Grafico da função de perda ao longo do treinamento')
plt.legend()
```

Figura 13.8 – EXERCÍCIO 3: GRÁFICO DE PERDA




```
input_usuario = np.repeat(a=8263, repeats=M)
film = np.array(list(set(movie_ids)))
preds = model.predict([input_usuario, film])

# descentraliza as predições
rat = preds.flatten() + avg_rating
# índice da maior nota
```

```
idx = np.argmax(rat)
print("Recomendação: Filme - ", film[idx], " / ", rat[idx] , "*")
```

Recomendação: Filme - 66366 / 19.101202 *

4 Deepdream

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

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)
```

Figura 13.9 – EXERCÍCIO 3: IMAGEM BAIXADA DA URL FORNECIDA

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

# 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)

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)

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
```

```
50
51          # Na subida gradiente, a "perda" é maximizada.
52          # Você pode atualizar a imagem adicionando diretamente os
        gradientes (porque eles têm o mesmo f
53          img = img + gradients*step_size
54          img = tf.clip_by_value(img, -1, 1)
55
56          return loss, img
57
58  deepdream = DeepDream(dream_model)
59
60  def run_deep_dream_simple(img, steps=100, step_size=0.01):
61      img = tf.keras.applications.inception_v3.preprocess_input(img)
62      img = tf.convert_to_tensor(img)
63      step_size = tf.convert_to_tensor(step_size)
64      steps_remaining = steps
65      step = 0
66      while steps_remaining:
67          if steps_remaining>100:
68              run_steps = tf.constant(100)
69          else:
70              run_steps = tf.constant(steps_remaining)
71
72          steps_remaining -= run_steps
73          step += run_steps
74
75          loss, img = deepdream(img, run_steps, tf.constant(step_size))
76
77          display.clear_output(wait=True)
78          show(deprocess(img))
79          print ("Step {}, loss {}".format(step, loss))
80
81      result = deprocess(img)
82      display.clear_output(wait=True)
83      show(result)
84
85      return result
86
87  dream_img = run_deep_dream_simple(img=original_img, steps=100,
        step_size=0.01)
```

Figura 13.10 – EXERCÍCIO 3: RESULTADO OBTIDO NO PROCESSO DE DEEP DREAM

```
import time
start = time.time()

OCTAVE_SCALE = 1.30
img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[:-1]
float_base_shape = tf.cast(base_shape, tf.float32)
for n in range(-2, 3):
    new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)
    img = tf.image.resize(img, new_shape).numpy()
    img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

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

Figura 13.11 – EXERCÍCIO 3: RESULTADO OBTIDO NO PROCESSO DE DEEP DREAM À OITAVA

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A - ENUNCIADO

Escolha um conjunto de dados brutos (ou uma visualização de dados que você acredite que possa ser melhorada) e faça uma visualização desses dados (de acordo com os dados escolhidos e com a ferramenta de sua escolha) Desenvolva uma narrativa/storytelling para essa visualização de dados considerando os conceitos e informações que foram discutidas nesta disciplina. Não esqueça de deixar claro para seu possível público alvo qual **o objetivo dessa visualização de dados, o que esses dados significam, quais possíveis ações podem ser feitas com base neles.**

Entregue em um PDF:

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

B - RESOLUÇÃO

Contexto e Público-Alvo

Os dados de pesquisa pelo termo "Passagem de avião" mostram um aumento expressivo em dezembro e janeiro. Esses picos coincidem com as férias no Brasil, indicando que muitos consumidores deixam para comprar passagens de última hora, em vez de se planejarem com antecedência. O público alvo serão gestores e analistas de marketing de companhias aéreas e agências de viagens, que podem ajustar campanhas e promoções.

Storytelling

A visualização traça a evolução das buscas pelo termo "passagem de avião" ao longo do tempo e revela um padrão sazonal marcante. Ao longo do ano, os níveis de interesse permanecem relativamente constantes, mas observam-se picos significativos em dezembro e janeiro. Esses aumentos coincidem com as férias de fim de ano e o início das férias escolares, períodos em que os consumidores tendem a adiar o planejamento da viagem e optam por comprar passagens em cima da hora.

A narrativa constrói a ideia de que essa concentração de buscas reflete um comportamento de compra impulsivo, em que a ausência de um planejamento prévio leva os viajantes a buscarem alternativas somente quando a viagem se aproxima. Esse insight sugere oportunidades para que companhias aéreas e agências de viagens desenvolvam estratégias de marketing mais eficazes, incentivando a compra antecipada por meio de promoções e ofertas exclusivas.

Ao explorar essa visualização, o público-alvo poderá compreender não apenas o padrão sazonal das buscas, mas também as implicações desse comportamento para o setor de turismo, possibilitando a adoção de estratégias que contribuam para uma distribuição mais equilibrada da demanda ao longo do ano.

Figura 14.1 – ANÁLISE DE TENDÊNCIAS DE BUSCA POR PASSAGEM DE AVIÃO

APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A - ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java. Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida). Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels. O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

a) 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.

b) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).

c) Utilize no mínimo uma população com 100 indivíduos;

d) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;

e) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);

f) 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 import math
2 import random
3 import matplotlib.pyplot as plt
4
5 # Definindo parâmetros do problema
```

```
6  NUM_CIDADES = 100
7  TAMANHO_POPULACAO = 100
8  NUM_GERACOES = 5000
9
10 # Função para calcular a distancia total de uma rota (caminho fechado
       que passa
11 def calcular_distancia_total(rota, coordenadas):
12     """
13     Calcula a distância total percorrida seguindo a sequência de
       cidades em 'rota'.
14     A distância é a soma das distâncias euclidianas entre cada par
       consecutivo.
15     *Importante: A rota começa e termina na cidade de origem (rota
       [0]).
16     """
17     distancia = 0.0
18     # Soma as distâncias entre cada cidade e a próxima na rota
19     for i in range(len(rota) - 1):
20         cidade_atual = rota[i]
21         proxima_cidade = rota[i+1]
22         # Obtém as coordenadas das duas cidades
23         x1, y1 = coordenadas[cidade_atual]
24         x2, y2 = coordenadas[proxima_cidade]
25         # Calcula a distancia Euclidiana entre cidade atual e proxima
       cidade
26         distancia += math.hypot(x2-x1, y2-y1)
27     return distancia
28
29 # Função de seleção por torneio para escolher um individuo (rota)
       como pai
30 def selecionar_pai_por_torneio(populacao, distancias, k=3):
31     """
32     Seleciona e retorna um individuo da população usando seleção por
       torneio.
33     Aleatoriamente escolhe 'k' indivíduos e retorna o que tiver a
       menor distância.
34     """
35     melhor_indice = -1
36     melhor_distancia = float('inf')
37     # Realiza o torneio entre individuos aleatórios
38     for i in range(k):
39         indice = random.randrange(len(populacao)) # escolhe um indice
       aleatorio
40         # Verifica se este individuo é o melhor do torneio até agora
41         if distancias[indice] < melhor_distancia:
42             melhor_distancia = distancias[indice]
43             melhor_indice = indice
44     # Retorna uma cópia da rota vencedora (melhor distância)
45     return list(populacao[melhor_indice])
46
47 def crossover_OX(pai1, pai2):
```

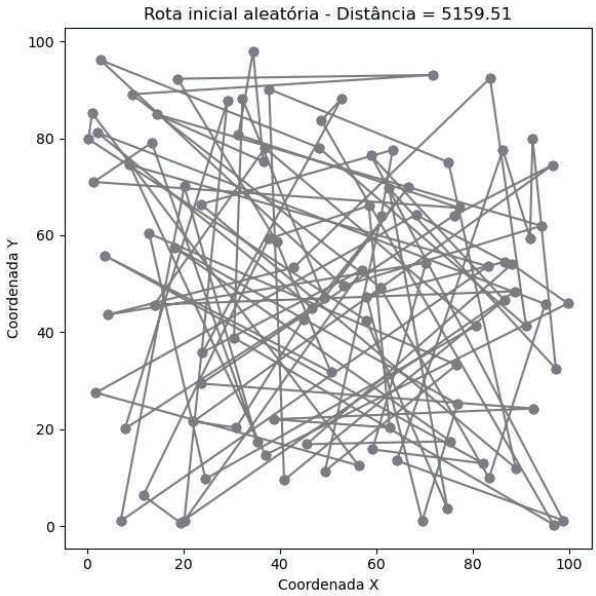
```
48      """
49      Realiza o crossover OX (Order Crossover) entre dois cromossomos (
        pais).
50      Retorna um cromossomo filho válido, com cidade inicial fixa.
51      """
52      tamanho = len(pai1)
53      # Inicializa o filho garantindo cidade inicial e final fixas
54      filho = [None] * tamanho
55      filho[0] = pai1[0]
56      filho[-1] = pai1[-1]
57
58      # Escolhe dois pontos de corte válidos e diferentes
59      corte1, corte2 = sorted(random.sample(range(1, tamanho - 1), 2))
60
61      # Copia segmento do pai1 diretamente para o filho
62      segmento_pai1 = pai1[corte1:corte2 + 1]
63      filho[corte1:corte2+1] = segmento_pai1
64
65      # Guarda as cidades ja inseridas no filho
66      cidades_no_filho = set(segmento_pai1)
67
68      # Preenche o restante das posições com cidades de pai2 (na ordem
        correta)
69      pos_filho = (corte2 + 1) % (tamanho - 1)
70      pos_pai2 = (corte2 + 1) % (tamanho - 1)
71
72      # Repete até o filho ser totalmente preenchido (sem None)
73      while None in filho[1:-1]:
74          cidade = pai2[pos_pai2]
75          if cidade not in cidades_no_filho:
76              filho[pos_filho] = cidade
77              cidades_no_filho.add(cidade)
78              pos_filho = pos_filho + 1 if pos_filho + 1 < tamanho - 1
        else 1
79          pos_pai2 = pos_pai2 + 1 if pos_pai2 + 1 < tamanho - 1 else 1
80
81      return filho
82
83  # Função de mutação que troca duas cidades de lugar na rota (mutação
        por troca)
84  def mutacao(rota):
85      """
86      Realiza uma mutação simples em uma rota, trocando de posição duas
        cidades aleatórias.
87      A cidade de origem (posição 0 e última posição) não é alterada.
88      Retorna uma nova rota mutada.
89      """
90      # Cria uma cópia da rota para mutação (para não alterar a rota
        original diretamente)
91      rota_mutada = list(rota)
92
```

```
93      # Seleciona aleatoriamente duas posições diferentes na parte
    intermediaria
94      i = random.randint(1, len(rota_mutada) - 2)
95      j = random.randint(1, len(rota_mutada) - 2)
96
97      while i == j:
98          j = random.randint(1, len(rota_mutada) - 2)
99
100     # Troca as cidades das posições i e j
101     rota_mutada[i], rota_mutada[j] = rota_mutada[j], rota_mutada[i]
102     return rota_mutada
103
104 # Gera coordenadas aleatórias para cada cidade dentro de um espaço
    100x100
105 coordenadas = [] # Lista para armazenar tuples (x, y) de coordenadas
    de cada cidade
106 for i in range(NUM_CIDADES):
107     x = random.random() * 100 # coordenada x aleatoria entre 0 e 100
108     y = random.random() * 100 # coordenada y aleatoria entre 0 e 100
109     coordenadas.append((x, y))
110 print("Coordenadas:", coordenadas)
111
112 # Gera a população inicial de rotas aleatórias
113 populacao = []
114 distancias = []
115 cidade_origem = 0 # definindo a cidade de indice 0 como cidade de
    origem
116
117 # Inicializa as variaveis para acompanhar a melhor solução encontrada
118 melhor_rota = None
119 melhor_distancia = float('inf')
120
121 for i in range(TAMANHO_POPULACAO):
122     # Cria uma ordem aleatória das cidades (exceto a cidade de origem
    )
123     outras_cidades = list(range(1, NUM_CIDADES))
124     random.shuffle(outras_cidades)
125     # Monta a rota completa: cidade de origem no início, seguida das
    outras cidades
126     rota = [cidade_origem] + outras_cidades + [cidade_origem]
127     # Adiciona a rota gerada à população
128     populacao.append(rota)
129     # Calcula a distância dessa rota e guarda
130     distancias.append(calcular_distancia_total(rota, coordenadas))
131
132 # Seleciona uma rota inicial aleatória (por exemplo, a primeira da
    população)
133 rota_inicial = populacao[0]
134 distancia_inicial = distancias[0]
135 print("Distancia inicial:", distancia_inicial)
136
```

```
137 # Identifica o melhor indivíduo da população inicial (com menor
    distancia)
138 for i, rota in enumerate(populacao):
139     d = distancias[i]
140     if d < melhor_distancia:
141         melhor_distancia = d
142         melhor_rota = rota
143 
144 # Loop principal do Algoritmo Genético
145 for geracao in range(NUM_GERACOES):
146     nova_populacao = []
147     nova_distancias = []
148 
149     # Elitismo: preserva a melhor rota da geração atual na próxima
150     nova_populacao.append(melhor_rota)
151     nova_distancias.append(melhor_distancia)
152 
153     # Determina quantos novos indivíduos serão gerados por crossover
    e por mutação
154     num_por_crossover = int(TAMANHO_POPULACAO * 0.90)
155     num_por_mutacao = int(TAMANHO_POPULACAO * 0.09)
156     if num_por_mutacao < 1:
157         num_por_mutacao = 1
158 
159     total_novos = 1 + num_por_crossover + num_por_mutacao # 1
    elitismo + novos
160     if total_novos > TAMANHO_POPULACAO:
161         num_por_crossover -= (total_novos - TAMANHO_POPULACAO)
162     elif total_novos < TAMANHO_POPULACAO:
163         num_por_crossover += (TAMANHO_POPULACAO - total_novos)
164 
165     # Gera novos individuos por crossover
166     for _ in range(num_por_crossover):
167         pai1 = selecionar_pai_por_torneio(populacao, distancias, k=3)
168         pai2 = selecionar_pai_por_torneio(populacao, distancias, k=3)
169         filho = crossover_OX(pai1, pai2)
170         nova_populacao.append(filho)
171         nova_distancias.append(calcular_distancia_total(filho,
    coordenadas))
172 
173     # Gera novos individuos por mutação
174     for _ in range(num_por_mutacao):
175         base = selecionar_pai_por_torneio(populacao, distancias, k=3)
176         individuo_mutado = mutacao(base)
177         nova_populacao.append(individuo_mutado)
178         nova_distancias.append(calcular_distancia_total(
    individuo_mutado, coordenadas))
179 
180     # Se por algum motivo excedemos o tamanho da população, cortamos
    o excedente
181     if len(nova_populacao) > TAMANHO_POPULACAO:
```

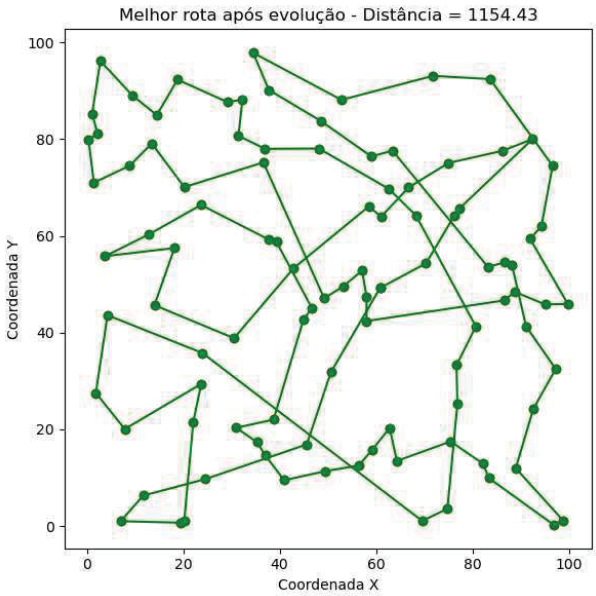
```
182         nova_populacao = nova_populacao[:TAMANHO_POPULACAO]
183         nova_distancias = nova_distancias[:TAMANHO_POPULACAO]
184 
185     # Atualiza a população para a proxima geração
186     populacao = nova_populacao
187     distancias = nova_distancias
188 
189     # Atualiza o melhor individuo
190     for i, d in enumerate(distancias):
191         if d < melhor_distancia:
192             melhor_distancia = d
193             melhor_rota = populacao[i]
194 
195     # A cada 100 gerações, imprime a evolução
196     if (geracao + 1) % 100 == 0 or geracao == 0:
197         print(f"Geração {geracao + 1}: melhor distância encontrada {
    melhor_distancia}")
198 
199 # Imprime as distâncias da rota inicial e da melhor rota final
    encontrada
200 print(f"\nDistancia total da rota inicial: {distancia_inicial:.2f}")
201 print(f"Distância total da melhor rota final: {melhor_distancia:.2f}\
    n")
202 
203 # Gera o gráfico da rota inicial aleatória
204 plt.figure(figsize=(6,6))
205 xs = [coordenadas[cidade][0] for cidade in rota_inicial]
206 ys = [coordenadas[cidade][1] for cidade in rota_inicial]
207 plt.plot(xs, ys, marker='o', color='gray')
208 plt.scatter(xs, ys, color="blue")
209 plt.title(f"Rota inicial aleatória - Distancia = {distancia_inicial
    :.2f}")
210 plt.xlabel("Coordenada X")
211 plt.ylabel("Coordenada Y")
212 plt.tight_layout()
```

Figura 15.1 – EXERCÍCIO 1: ROTA INICIAL ALEATÓRIA, DISTÂNCIA = 5159,51




```
# Gera o gráfico da melhor rota final apos a evolução
plt.figure(figsize=(6,6))
xs_best = [coordenadas[cidade][0] for cidade in melhor_rota]
ys_best = [coordenadas[cidade][1] for cidade in melhor_rota]
plt.plot(xs_best, ys_best, marker='o', color='green')
plt.scatter(xs_best, ys_best, color='red')
plt.title(f"Melhor rota após evolução - Distância = {melhor_distancia
    :.2f}")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")
plt.tight_layout()

# Mostra os gráficos gerados
plt.show()
```

Figura 15.2 – EXERCÍCIO 1: ROTA APÓS EVOLUÇÃO, DISTÂNCIA = 1154,43

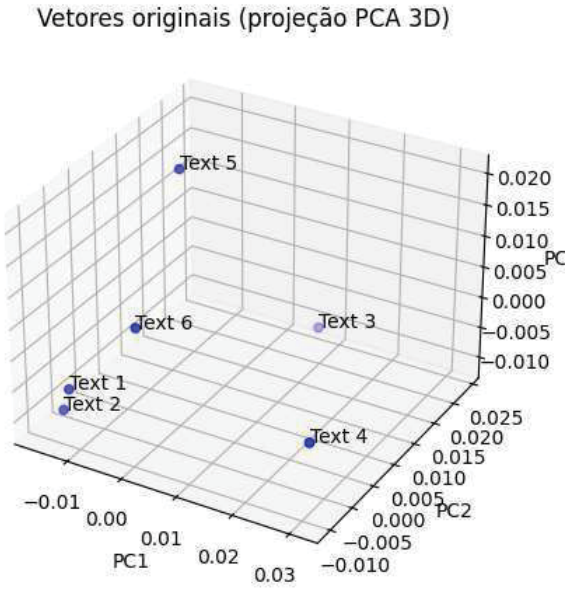


2) Compare a representação de dois modelos vetoriais

```
# Comandos para manipulação de pacotes
!pip uninstall numpy -y
!pip install numpy==1.25.2
!pip uninstall gensim -y
!pip install --no-binary :all: gensim

import numpy as np
import re
from gensim.models import Word2Vec
from sklearn.decomposition import PCA
from scipy.spatial.distance import pdist, squareform
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

# Textos curtos de exemplo
textos = [
    "O gato está dormindo no sofá.",        # Texto 1
    "O cachorro está dormindo no tapete.", # Texto 2 (similar to Texto 1)
    "A tecnologia esta mudando nossas vidas.", # Texto 3
    "Estamos vivendo na era da tecnologia.",   # Texto 4 (similar to Texto 3)
    "O dia está ensolarado e claro.",          # Texto 5
    "O sol está brilhando no céu azul."       # Texto 6 (similar to Texto 5)
]

# Pre-processamento: impor pontuação e criar lista de tokens por texto
sentencas_tokens = []
for txt in textos:
```

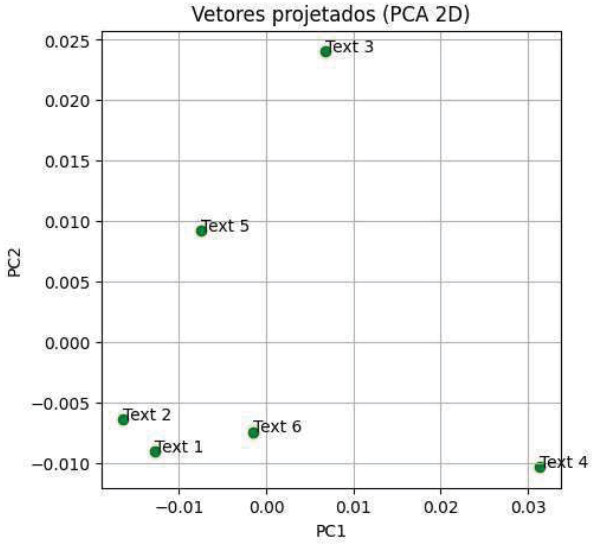
```
    # Converter para minúsculas e remover pontuação
    texto_limpo = re.sub(r'[^\w\s]', '', txt.lower())
    tokens = texto_limpo.split()
    if tokens:
        sentencas_tokens.append(tokens)

# Treinar modelo Word2Vec nos tokens
model = Word2Vec(sentencas_tokens, vector_size=50, window=5,
   min_count=1, workers=4)

# Obter vetor de cada sentenca fazendo média dos vetores de palavras
vetores_texto = []
for tokens in sentencas_tokens:
    vetores_palavras = [model.wv[word] for word in tokens if word in
   model.wv.key_to_index]
    if len(vetores_palavras) > 0:
        # Média dos vetores de palavra para representar a sentença
        vetor_sentenca = np.mean(vetores_palavras, axis=0)
    else:
        # Caso sentença sem palavras conhecidas
        vetor_sentenca = np.zeros(model.vector_size)
    vetores_texto.append(vetor_sentenca)
vetores_texto = np.array(vetores_texto)

# Calcular vetor medio e centralizar os vetores de texto
vetor_medio = np.mean(vetores_texto, axis=0)
vetores_centralizados = vetores_texto - vetor_medio

# Aplicar PCA para reduzir dimensão dos vetores
pca_3d = PCA(n_components=3)
pca_2d = PCA(n_components=2)
vetores_3d = pca_3d.fit_transform(vetores_centralizados)
vetores_2d = pca_2d.fit_transform(vetores_centralizados)

# Visualização dos vetores em 3D e 2D
labels = [f"Text {i+1}" for i in range(len(textos))]

# Plot 3D dos vetores originais (após PCA 3D)
fig = plt.figure(figsize=(6,5))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(vetores_3d[:,0], vetores_3d[:,1], vetores_3d[:,2], c='blue
   ', marker='o')
for i, label in enumerate(labels):
    ax.text(vetores_3d[i,0], vetores_3d[i,1], vetores_3d[i,2], label)
ax.set_title('Vetores originais (projeção PCA 3D)')
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
plt.show()
```

Figura 15.3 – EXERCÍCIO 2: PROJEÇÃO PCA 3D




```
# Plot 2D dos vetores projetados (PCA 2D)
plt.figure(figsize=(5,5))
plt.scatter(vetores_2d[:,0], vetores_2d[:,1], c='green', marker='o')
for i, label in enumerate(labels):
    plt.annotate(label, (vetores_2d[i,0], vetores_2d[i,1]))
plt.title('Vetores projetados (PCA 2D)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.grid(True)
plt.show()
```

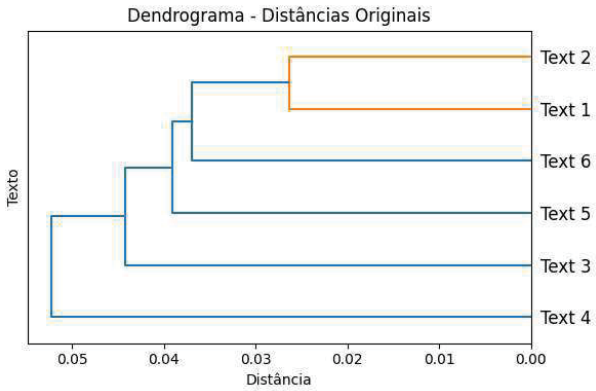
Figura 15.4 – EXERCÍCIO 2: PROJEÇÃO PCA 2D




```
# Calcular matrizes de distancia entre vetores
```

```
dist_original = squareform(pdist(vetores_centralizados, metric='
    euclidean'))
dist_projetado = squareform(pdist(vetores_2d, metric='euclidean'))

# Gerar dendrogramas de agrupamento hierárquico
# Dendrograma para distâncias originais
Z_orig = linkage(vetores_centralizados, method='ward', metric='
    euclidean')
plt.figure(figsize=(6,4))
dendrogram(Z_orig, labels=labels, orientation='left')
plt.title('Dendrograma - Distâncias Originais')
plt.xlabel('Distância')
plt.ylabel('Texto')
plt.tight_layout()
plt.show()
```

Figura 15.5 – EXERCÍCIO 2: DENDOGRAMA - DISTÂNCIAS ORIGINAIS




```
# Dendrograma para distâncias após projeção PCA 2D
Z_proj = linkage(vetores_2d, method='ward', metric='euclidean')
plt.figure(figsize=(6,4))
dendrogram(Z_proj, labels=labels, orientation='left')
plt.title('Dendrograma - Distâncias PCA 2D')
plt.xlabel('Distância')
plt.ylabel('Texto')
plt.tight_layout()
plt.show()
```

Figura 15.6 – EXERCÍCIO 2: DENDOGRAMA - DISTÂNCIAS PCA 2D