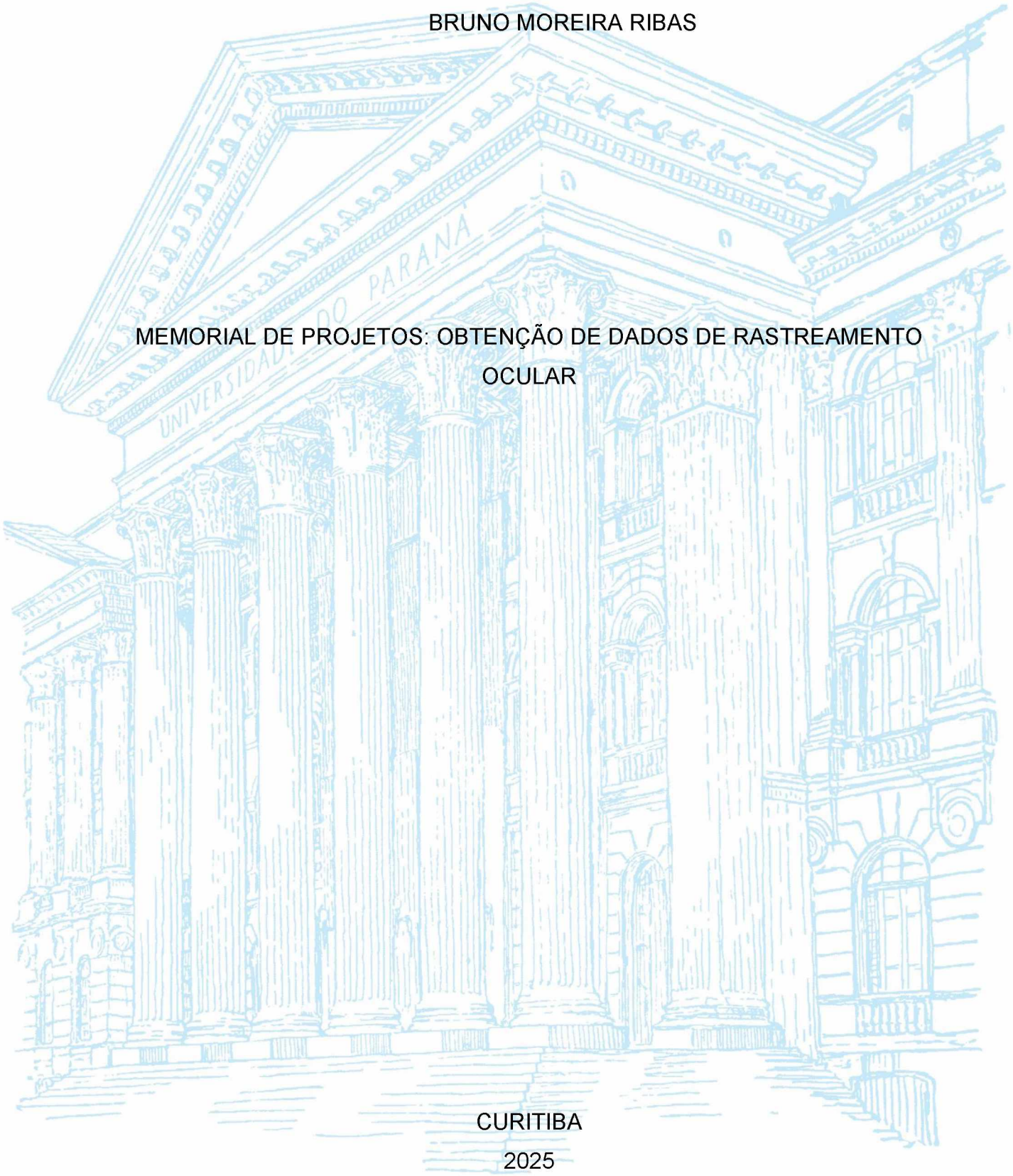
UNIVERSIDADE FEDERAL DO PARANÁ

BRUNO MOREIRA RIBAS



MEMORIAL DE PROJETOS: OBTENÇÃO DE DADOS DE RASTREAMENTO OCULAR

CURITIBA

2025

BRUNO MOREIRA RIBAS

MEMORIAL DE PROJETOS: OBTENÇÃO DE DADOS DE RASTREAMENTO OCULAR

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

Orientador: Prof. Dr Jaime Wojciechowski

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **BRUNO MOREIRA RIBAS**, intitulada: **MEMORIAL DE PROJETOS: OBTENÇÃO DE DADOS DE RASTREAMENTO OCULAR**, 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, 20 de Outubro de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

O presente trabalho de conclusão de curso da Especialização em Inteligência Artificial Aplicada visa apresentar o uso do rastreamento ocular para obtenção de dados. Iniciando pela experiência do usuário (UX), para determinar a qualidade do serviço. O propósito é fornecer suporte àqueles que utilizam a técnica de rastreamento ocular para coletar dados que auxiliem na tomada de decisões sobre a melhoria da experiência do usuário (UX) de seus serviços ou produtos. Citando diversas fontes bibliográficas, desde os primeiros autores nesta área, por volta do ano 2000, em que era comum perda de dados, até estudos mais recentes, com uso de tecnologia avançada e dispositivos que mapeiam o olhar com exatidão. Aborda métodos de rastreamento como HOG (Histograma de Gradiente Orientado) e Árvores de Regressão. Ao final, exemplo de dois casos reais de rastreamento: o primeiro, utilizado em uma situação real, com clientes; o segundo, uma coleta de dados realizada pelo autor, fazendo uso de rastreamento do olhar em tempo real.

Palavras-chave: rastreamento ocular; experiência do usuário (ux); inteligência artificial aplicada; coleta de dados; tomada de decisão.

ABSTRACT

This final Project for the Specialization in Applied Artificial Intelligence aims to present the use of eye tracking for data acquisition, starting with user experience (UX) to determine service quality. The purpose is to provide support to those who use the eye-tracking technique to collect data that assists in decision-making regarding the improvement of the user experience (ux) of their services or products. It cites various bibliographic sources, from the earliest authors in this field around the year 2000, when data loss was common, to more recent studies using advanced technology and devices that accurately map gaze. It covers tracking methods such as HOG (Oriented Gradient Histogram) and Regression Trees. Finally, it includes examples of two real-life tracking cases: the first, used in a real-life situation with customers; the second, a data collection performed by the author, using real-time eye tracking.

Keywords: eye tracking; user experience (ux); applied artificial intelligence; data collection; decision making.

SUMÁRIO

1. PARECER TÉCNICO

O presente trabalho fará menção à obtenção de dados de rastreamento ocular, utilizando Inteligência Artificial. O rastreamento ocular não é assunto novo, mas evoluiu muito ao longo dos anos, melhorando o desconforto e ampliando sua aplicabilidade.

A experiência do usuário (UX) é um aspecto crucial para determinar a qualidade de um produto ou serviço. Diversos indicadores são empregados para compreender a UX, incluindo questionários que avaliam a satisfação, perguntas sobre aspectos específicos do produto ou serviço e medidas de tempo, como o tempo necessário para realizar tarefas no sistema e o tempo de resposta do sistema ao usuário. Apesar da variedade de indicadores disponíveis, há esforços para unificá-los, como destacado por Strohl et al. (2015).

Entre esses indicadores, o rastreamento ocular tem emergido como uma ferramenta promissora na avaliação da experiência do usuário, especialmente em produtos e serviços digitais (Zardari et al. 2021). No entanto, trabalhar com dados oculares apresenta desafios devido à qualidade variável desses dados, que pode ser afetada por fatores como calibração inadequada dos equipamentos, movimentação excessiva durante a captura dos dados e variações na coloração dos olhos, resultando em perda de dados (*data-loss*) (Hasse e Bruder, 2015; Nyström et al. 2013), e tornando a captura dos dados um maior desafio.

1.1 OBTENÇÃO DE DADOS DE RASTREAMENTO OCULAR – MÉTODOS

Este trabalho é de caráter bibliográfico, e se propõe a apresentar técnicas para realizar o rastreamento ocular no contexto da experiência do usuário, com foco no processamento de dados oculares de maneira abrangente. Serão discutidos métodos para lidar com os desafios inerentes à análise de dados oculares, visando oferecer informações para aprimorar a compreensão da experiência do usuário e, consequentemente, a qualidade dos produtos e serviços.

Alguns estudos já exploraram esse tema, como o artigo de Pretorius, Biljon e Kock (2010), que demonstra as experiências de usuários experientes e inexperientes de um determinado sistema. Também temos trabalhos conduzidos com redes neurais

convolucionais por Fuhl, Rong e Kasneci (2020). Outras exploram o eye tracking em diferentes dispositivos e telas como o artigo Andersson et al. (2017) onde é abordado no contexto de dispositivos mobile, foi utilizado também o estudo Park, Aksan et al. (2020) onde é feito eye-tracking utilizando vídeo de um usuário em frente às câmeras.

Para melhorar a suavização dos dados, especialmente em relação aos problemas de mudança de direção do olhar (*Eye Gaze*), foi adotado o método descrito por Park e Zhang et al. (2018), que se baseou nas proposições anteriores de Park e Aksan et al. (2020), com a técnica de normalização conforme sugerida por Zhang, Sugano e Bulling (2018).

1.2 EXPERIÊNCIA DO USUÁRIO (UX)

A Experiência do Usuário, ou UX, pode ser descrita como um conjunto de informações que quantificam e medem o quão eficaz pode ser um produto ou serviço com base em várias interações do usuário. Segundo Morville e Sullenger (2010) a experiência do usuário também pode ser obtida por meio de sete itens: Útil, Utilizável, Localizável, Valioso, Desejável, Acessível e Confiável.

Ao ser denominado como Utilizável, o produto deve ser fácil de usar, minimizando as dificuldades do usuário em alcançar os objetivos do produto. Ao ser denominado Acessível, o produto deve ser facilmente acessível a todos os usuários, incluindo aqueles com necessidades especiais.

Um dos primeiros autores que falam sobre o UX é Nielsen (1993) em seu livro *Usability Engineering*, onde ele explora a importância de projetar sistemas que sejam fáceis de usar e proporcionem uma experiência satisfatória ao usuário, além de apresentar uma abordagem sistemática para melhorar a usabilidade de sistemas interativos, enfatizando a importância de projetar produtos que sejam fáceis de aprender, eficientes de usar e proporcionem satisfação ao usuário.

Outro livro que aborda assuntos relevantes a experiência do usuário é *The Design of Everyday Things,* onde Norman (2002) introduz conceitos fundamentais como: *Affordances e Feedback.*

Norman descreve *affordances* como as possibilidades de ação que um objeto oferece intuitivamente ao usuário. Ele argumenta que um bom design deve comunicar claramente suas possibilidades para minimizar a necessidade de instruções explícitas.

Além disso, discute a importância do *feedback* (opinião) imediato para informar aos usuários o resultado de suas ações.

Em sua obra Norman aborda também a técnica de Mapping e Design Centrado no Usuário, na qual o autor enfatiza a importância de um design centrado no usuário, onde os objetos e sistemas são projetados considerando as habilidades e expectativas naturais dos usuários. Ele critica designs que dificultam o uso ou causam confusão devido a um mau mapeamento entre a funcionalidade do objeto e as expectativas do usuário.

Norman (2002) se refere ainda ao Conceito de Design Emocional, introduzindo a ideia de que o design não apenas deve ser funcional, mas também deve evocar uma resposta emocional positiva dos usuários. Isso envolve considerações estéticas, ergonômicas e psicológicas que contribuem para uma experiência satisfatória e gratificante.

Outro tópico abordado se trata dos Erros de Design; Norman (2002) explora diferentes tipos de erros de design, desde aqueles causados por falta de *feedback* claro até problemas de visibilidade e *feedback* pobre. Ele argumenta ainda que muitos erros de uso não são culpa do usuário, mas sim do design inadequado do sistema.

1.3 RASTREAMENTO OCULAR (EYE TRACKING), EVOLUÇÃO E DESAFIOS

A técnica de rastreamento ocular é amplamente utilizada para monitorar precisamente onde os olhos estão focados durante a captura de dados, analisando a localização, movimento e tamanho da pupila para identificar áreas de interesse, segundo Hasse e Bruder (2015). A pesquisa nessa área evoluiu significativamente, resultando em algoritmos avançados para a detecção automática de movimentos oculares (Braunagel, 2016; Braunagel et. al. 2016). No entanto, desafios como a perda de dados durante o rastreamento ocular continuam sendo um problema destacado na literatura Hessels e Hooge (2019), com estudos explorando soluções potenciais para mitigar esse problema.

Outro desafio significativo é o custo associado aos equipamentos e métodos de calibração necessários para realizar esses estudos, como mencionado no Youth Study (2011). Como resultado, muitas pesquisas recentes têm adotado abordagens remotas, onde os participantes são monitorados por câmeras ou sensores, com o

apoio de técnicas de inteligência artificial para melhorar a precisão dos dados e reduzir perdas. (Park, Zhang et al., 2018; Park, Aksan et al., 2020).

Ao examinar os movimentos oculares das pessoas, podemos obter percepções sobre seus processos de atenção e aprender mais sobre o que elas consideram importante, interessante ou confuso (Bojko 2005). Existem diferentes tipos de movimentos oculares, um deles é denominado saccades, onde os olhos pulam de um estímulo para outro (Bojko 2005) e outro é a fixação, sendo ela quando os olhos ficam parados em algum lugar (Salvucci e Goldberg, 2000), existem outros movimentos oculares como no livro do *Eye Tracking Methodology: Theory and Practice,* de Duchowski (2017).

Eventos como o simpósio ETRA *(Eye Tracking Research and Application)* da ACM desempenham um papel crucial ao apresentar as mais recentes pesquisas e avanços na área de rastreamento ocular.

1.3.1 Método HOG (Histograma De Gradiente Orientado)

O método de extração de características de imagens HOG foi introduzido em 2005 pelos pesquisadores Dalal e Triggs como parte de um algoritmo de detecção de pedestres em imagens. Este método visa extrair informações referentes à orientação das arestas presentes em uma imagem, sendo essas arestas calculadas por métodos de detecção de bordas, como o de Sobel e Feldman.

O HOG divide a imagem em pequenas regiões chamadas células e calcula um histograma de gradientes orientados para cada célula. Esses histogramas são então normalizados em blocos para melhorar a invariância a mudanças de iluminação e contraste. As características extraídas são robustas a pequenas variações na forma e aparência do objeto, tornando o HOG um método eficaz para a detecção de objetos.

1.3.2 Árvores de Regressão

Árvores de regressão são um tipo de modelo de aprendizado de máquina usado para prever valores contínuos. Elas funcionam dividindo os dados de entrada em subconjuntos mais simples, com base em regras de decisão aprendidas dos dados

de treinamento. Cada nó da árvore representa uma característica do dado, e cada folha representa um valor de saída. O objetivo é criar uma árvore que minimize o erro de previsão para o conjunto de dados de treinamento.

Um exemplo clássico do uso de árvores de regressão é o trabalho de Kazemi e Sullivan (2014), que desenvolveram um método rápido e eficiente para o alinhamento facial utilizando um conjunto de árvores de regressão. Nele, eles propõem uma abordagem que alinha faces em imagens prevendo a posição de 68 pontos de referência faciais.

1.4 TRABALHOS RELACIONADOS

Utilizando como referência principal o trabalho *Eye tracking in user experience testing: How to make the most of it* (Rastreamento ocular em testes de experiência do usuário: como aproveitá-lo ao máximo), de Agnieszka Bojko, publicado em 2005, nos Anais da 14ª Conferência Anual da Usability Professionals Association (UPA).

Os trabalhos do Bojko (2005) demonstrando o uso de *eye tracking* no contexto de experiência do usuário utiliza diferentes sites para demonstrar a pesquisa como exemplo, o site de uma das lojas do Starbucks onde mostra os mapas de calor ilustrando a atividade combinada do olhar dos usuários que procuram um localizador de lojas na página inicial da Starbucks. O referido artigo nos mostra que *eye tracking* é uma ferramenta útil para nos trazer informações sobre experiência do usuário, porém, a pesquisa tem que ser feita com cuidado e ser bem planejada. (FIGURA 1)

FIGURA 1 – MAPA DE CALOR

FONTE: Bojko (2000)

A Figura 1 mostra que os usuários à esquerda (n = 13) receberam uma versão em preto e branco da página inicial, enquanto os usuários à direita (n = 12) viram sua versão colorida real. As cores nos mapas de calor representam a porcentagem de usuários fixando o olhar em qualquer área específica, com cores mais quentes indicando porcentagens mais altas (Bojko, 2000).

1.5 RASTREAMENTO DO OLHAR EM TEMPO REAL

O rastreamento teve como base a publicação *Real-time camera-based eye gaze tracking using convolutional neural network: a case study on social media website* (Rastreamento do olhar baseado em câmera em tempo real usando rede neural convolucional: um estudo de caso em um site de mídia social) tendo como desenvolvedores Nandini Modi e Jaiteg Singh, em 2022.

Esse estudo mostra o estudo do olhar e mapas de calor no contexto de redes sociais, onde ele faz a detecção da face, acha a região dos olhos e utilizando CNN mapeia o olho direito e o olho esquerdo e então as classifica.

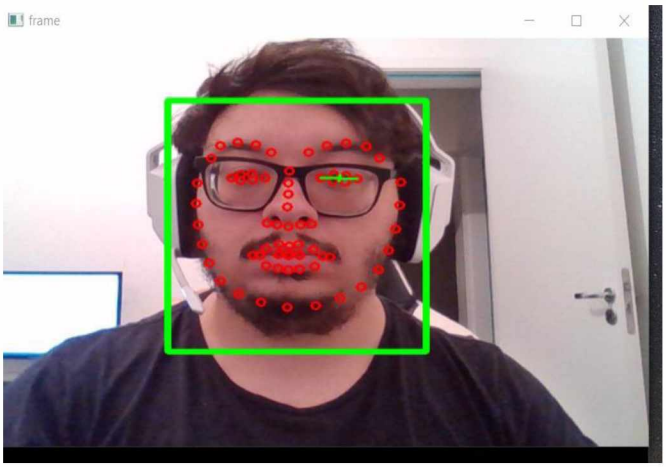
1.5.1 Materiais e métodos

Foi utilizada a base shape predictor 68 face landmarks (Kazemi e Sullivan, 2014). Foi utilizado o algoritmo HOG (Dalal e Triggs, 2005) em Python, utilizando Dblib3 OpenCV Development Team Year, OpenCV4 e NumPy Community Year Numpy.

1.5.2 Experimentos E Resultados

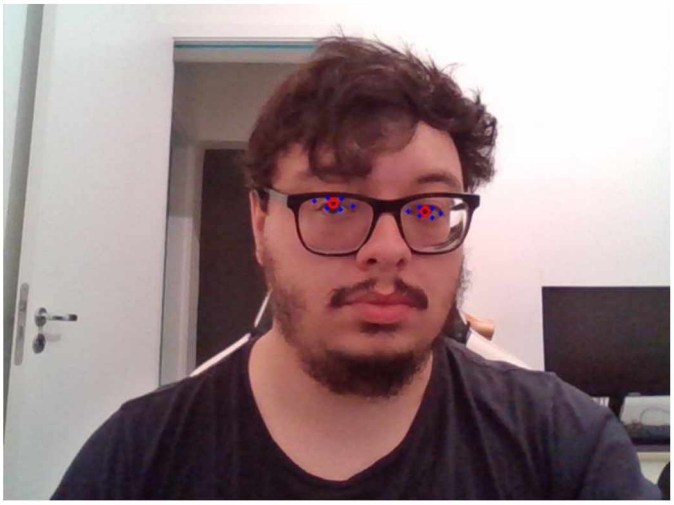
O experimento feito captura em tempo real por dados de vídeo as delimitações no rosto do usuário (Hooge et al., 2019) utilizando HOG (Dalal e Triggs, 2005). Após isso buscam-se os olhos do usuário para maior precisão de rastreamento ocular como mostrado na imagem 2 e 3. Assim tendo dados oculares em tempo real do usuário em frente a câmera. (FIGURAS 2 e 3)

FIGURA 2 – Código fazendo tracking da face e do olho



FONTE: O autor (2025)

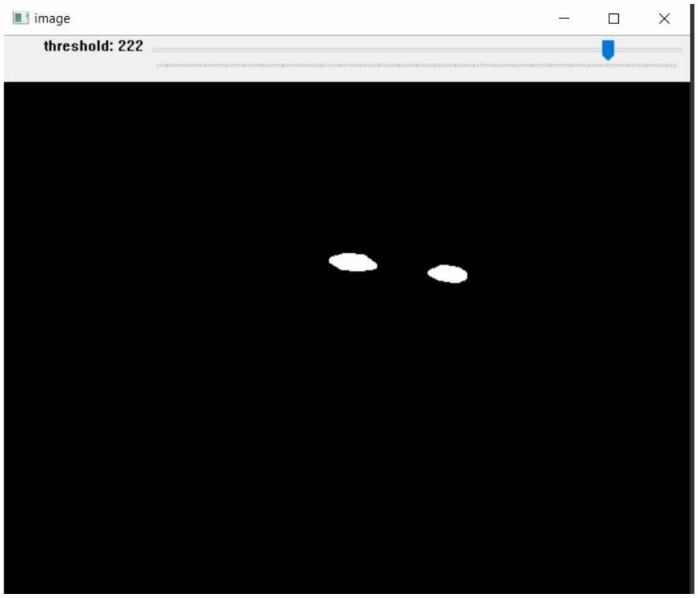
FIGURA 3 – Olhos marcados pelo eye tracking



FONTE: O autor (2025)

A câmera analisa os movimentos e padrões do olhar da pessoa, registrando a fixação do foco visual, dilatação da pupila e outras atividades oculares.

FIGURA 4 – Olhos marcados somente o olho com threshold máximo



FONTE: O autor (2025)

1.6 CONSIDERAÇÕES FINAIS

Esse trabalho teve como intuito tentar fazer o rastreamento ocular do usuário em tempo real como forma de obtenção de dados para entender a experiência do usuário em relação a um produto e serviço. Conseguiu-se chegar no objetivo da obtenção de dados oculares, porém outros trabalhos conseguiram chegar mais longe retirando problemas de *saccades* e outros que geram *data loss.*

Existem algoritmos melhores para obtenção de dados oculares, como mostrado por Park e Aksan (2020), e outras que continuam sendo discutidos no ETRA e na ECCV.

REFERÊNCIAS

ANDERSSON, R.; LARSSON, L.; HOLMQVIST, K.; STRIDH, M.; NYSTRÖM, M. **One algorithm to rule them all?** An evaluation and discussion of ten eye movement event-detection algorithms. Behavior Research Methods, v. 49, p. 616–637, 2017. DOI: 10.3758/s13428-016-0738-9.

BOJKO, A. **Eye tracking in user experience testing:** How to make the most of it. In: Proceedings of UPA '05. Montréal, Canada, 2005.

BRAUNAGEL, C. et al. **Automatic detection of eye movements**. Journal of Vision, 2016.

BRAUNAGEL, C.; GEISLER, D.; STOLZMANN, W.; ROSENSTIEL, W.; KASNECI, E. **On the necessity of adaptive eye movement classification in conditionally automated driving scenarios.** In: Proceedings of the Eye Tracking Research and Applications Symposium (ETRA). [S. l.: s. n.], 2016. v. 14, p. 19–26. DOI: 10.1145/2857491.2857529.

DALAL, N.; TRIGGS, B. **Histograms of Oriented Gradients for Human Detection**. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). San Diego, CA, USA: IEEE Computer Society, 2005. v. 1, p. 886–893. DOI: 10.1109/CVPR.2005.177.

DUCHOWSKI, A. T. **Eye Tracking Methodology: Theory and Practice**. 3. ed. Cham, Switzerland: Springer International Publishing AG, 2017.

FUHL, W.; RONG, Y.; KASNECI, E. **Fully Convolutional Neural Networks for Raw Eye Tracking Data Segmentation, Generation, and Reconstruction**. In: Proceedings–International Conference on Pattern Recognition. [S. l.: s. n.], 2020. p. 142–149. DOI: 10.1109/ICPR48806.2021.9413268.

HASSE, C.; BRUDER, C. **Eye-tracking measurements and their link to a normative model of monitoring behaviour.** Ergonomics, v. 58, n. 3, p. 355–367, 2015. DOI: 10.1080/00140139.2014.967310.

HASSE, C.; BRUDER, G. **Eye tracking:** A comprehensive guide. Journal of Eye Tracking Research, 2015.

HESSELS, R. S.; HOOGE, I. T. C. **Eye tracking in developmental cognitive neuroscience** – The good, the bad and the ugly. Developmental Cognitive Neuroscience, v. 40, p. 100710, 2019. DOI: 10.1016/j.dcn.2019.100710. Disponível em: https://www.sciencedirect.com/science/article/pii/S187892931930297X.

HESSELS, R. S.; HOOGE, I. T. C. **Loss of data in eye movement research:** Causes and solutions. Behavior Research Methods, 2019.

HOOGE, I. T. C.; HOLLEMAN, G. A.; HAUKES, N. C.; HESSELS, R. S. **Gaze tracking accuracy in humans:** One eye is sometimes better than two. Behavior Research

Methods, v. 51, p. 2712–2721, 2019. DOI: 10.3758/s13428-018-1135-3.

KAZEMI, V.; SULLIVAN, J. **One millisecond face alignment with an ensemble of regression trees.** In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [S. l.: s. n.], 2014. p. 1867–1874. DOI: 10.1109/CVPR.2014.241.

MODI, N.; SINGH, J. **Real-time camera-based eye gaze tracking using convolutional neural network:** a case study on social media website. Virtual Reality, v. 26, p. 1489–1506, 2022. DOI: 10.1007/s10055-022-00642-6. Disponível em: https://doi.org/10.1007/s10055-022-00642-6.

MORVILLE, P.; SULLENGER, P. **Ambient findability:** Libraries, serials, and the internet of things. The Serials Librarian, 2010. DOI: 10.1080/03615261003622999.

NIELSEN, J. **Usability Engineering.** San Francisco, CA: Morgan Kaufmann, 1993.

NORMAN, D. A. **The Design of Everyday Things.** New York, NY: Basic Books, 2002.

NUMPY COMMUNITY. **NumPy Documentation.** [S. l.]: NumPy, [s.d.]. Disponível em: https://numpy.org/doc/.

NYSTRÖM, M.; ANDERSSON, R.; HOLMQVIST, K.; VAN DE WEIJER, J. **The influence of calibration method and eye physiology on eyetracking data quality.** Behavior Research Methods, v. 45, n. 1, p. 272–288, 2013. DOI: 10.3758/s13428-012-0247-4.

OPENCV DEVELOPMENT TEAM. **OpenCV Documentation**. [S. l.]: OpenCV, [s.d.]. Disponível em: https://docs.opencv.org/.

PARK, J.; AKSAN, N. **Remote eye tracking:** Enhancing data quality with AI. Journal of Artificial Intelligence Research, 2020.

PARK, S.; AKSAN, E.; ZHANG, X.; HILLIGES, O. **Towards End-to-end Video-based Eye-Tracking.** In: European Conference on Computer Vision (ECCV). [S. l.: s. n.], 2020.

PARK, S.; ZHANG, X.; BULLING, A.; HILLIGES, O. **Learning to Find Eye Region Landmarks for Remote Gaze Estimation in Unconstrained Settings.** In: ACM Symposium on Eye Tracking Research and Applications (ETRA). New York, NY, USA: ACM, 2018.

PERCEPTUAL USER INTERFACES. **Data normalization for gaze estimation.** 2018. Vídeo. Disponível em: https://youtu.be/D3gcMEzLl5s?si=atNbbb5zddYPVlBk. Acesso em: 21 jun. 2018.

PRETORIUS, M. C.; BILJON, J. V.; KOCK, E. D. **Added Value of Eye Tracking in Usability Studies:** Expert and Non-expert Participants. In: FORBRIG, P.; PATERNÓ, F.; PEJTERSEN, A. M. (eds.). Human-Computer Interaction. Berlin, Heidelberg: Springer, 2010. p. 115–125. (IFIP Advances in Information and Communication

Technology, v. 33

SALVUCCI, D. and GOLDBERG, J. H. **Identifying fixations and saccades in eye-tracking protocols.** Proceedings of the Eye Tracking Research and Applications Symposium, Association for Computing Machinery (ACM), 2000.

STROHL, J.; GONZALEZ, C.; SAUSER, J.; MONTAZERI, S. and GRIEPENTROG. **Creating Forms and Disclosures that Work:** Using Eye Tracking to Improve the User Experience. UAHCI 2015.

YOUTH STUDY GROUP. **Eye Tracking Study on Youth Behavior.** Academic Press, 2011.

ZARDARI, B. A.; HUSSAIN, Z.; ARAIN, A. A.; RIZVI, W. H. and VIGHIO, M. S. **Applying heuristic evaluation, usability testing and eye tracking.** Universal Access in the Information Society, 2021.

ZHANG, X.; SUGANO, Y. and BULLING, A. **Eye tracking in developmental cognitive neuroscience** – The good, the bad and the ugly. Developmental Cognitive Neuroscience, V. 40, 2019. Disponível em: https://doi.org/10.1016/j.dcn.2019.100710

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

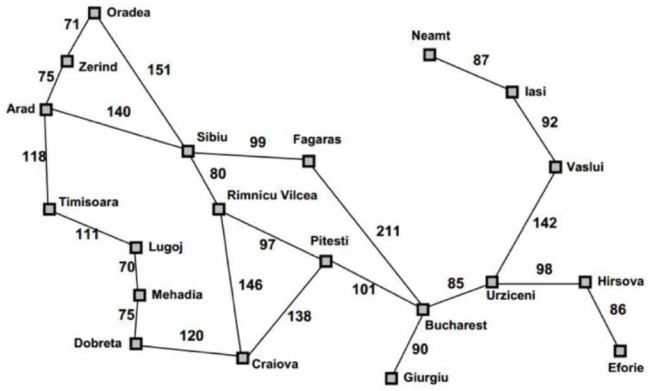
1 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

2 Lógica

Verificar se o argumento lógico é válido.

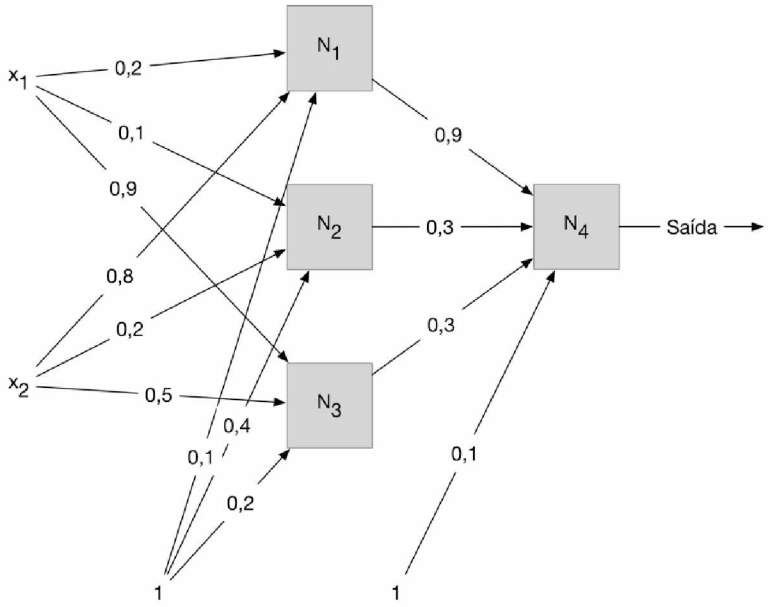
Se as uvas caem, então a raposa as come
Se a raposa as come, então estão maduras
As uvas estão verdes ou caem

Logo

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

3 Redes Neurais Artificiais

Seja a RNA da figura abaixo.

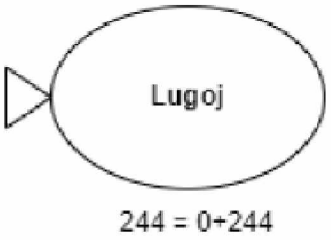


Dada a entrada x1=-3, x2=1, dê os valores de saída de todos os neurônios e indique qual é a saída da rede.

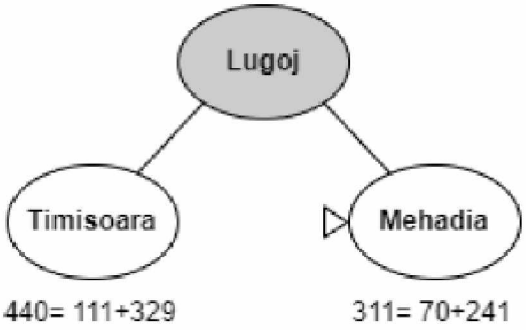
Os neurônios N1, N2 e N3 possuem função de ativação linear. Já N4 possui função de ativação tangente hiperbólica (pesquise a fórmula e aplique).

a) **B - RESOLUÇÃO**

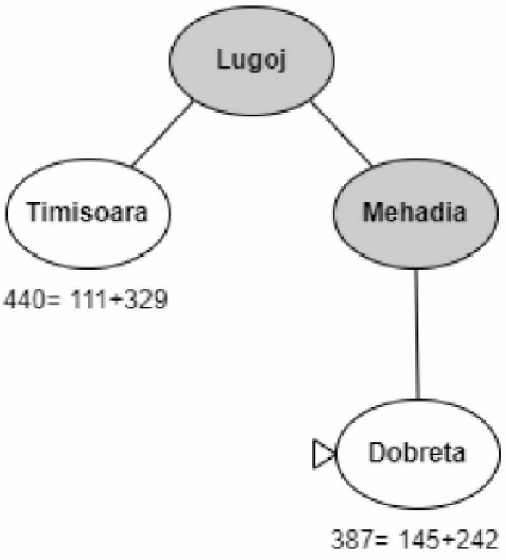
(a) Estado inicial



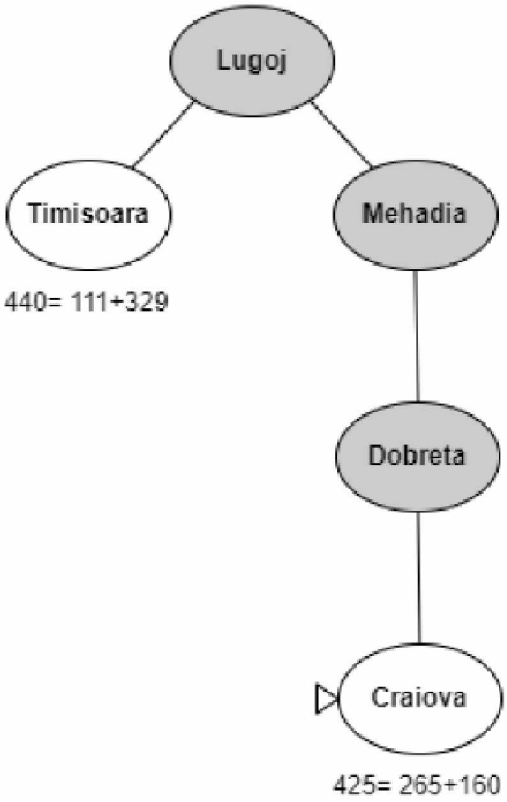
(b) Após expandir Lugoj



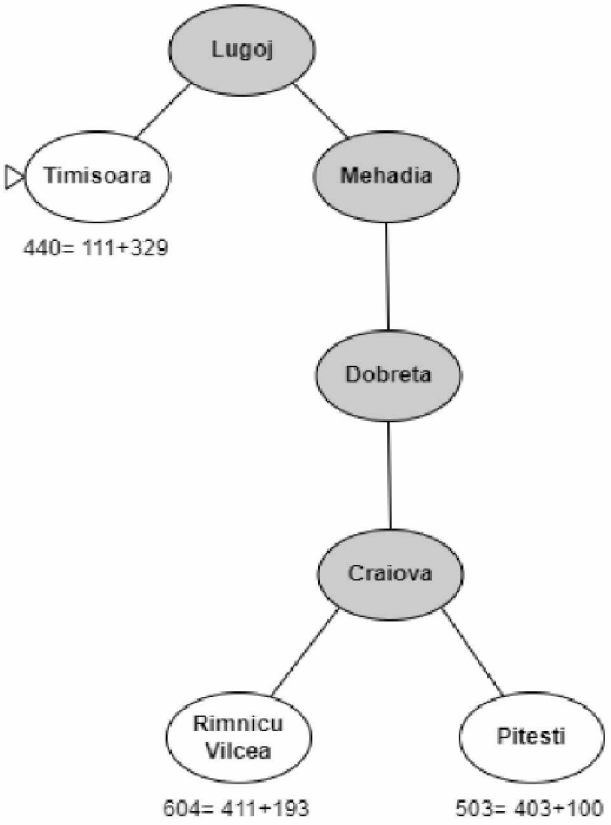
(c) Após expandir Mehadia



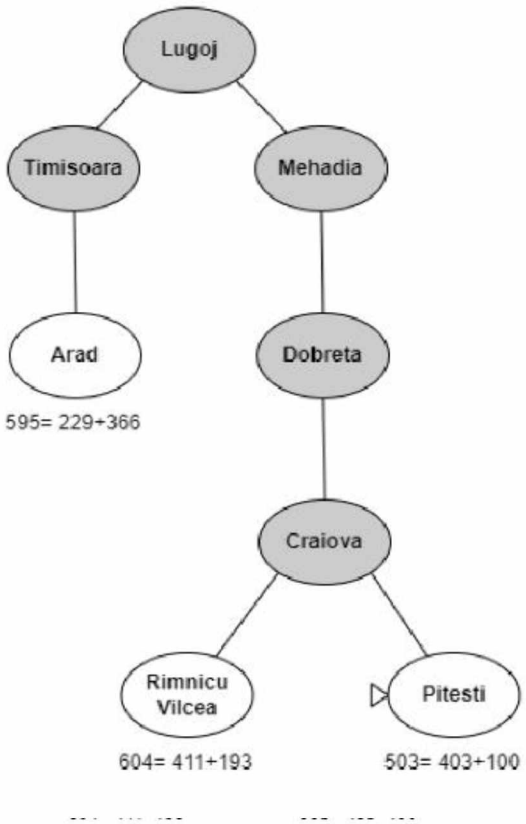
(d) Após expandir Dobreta



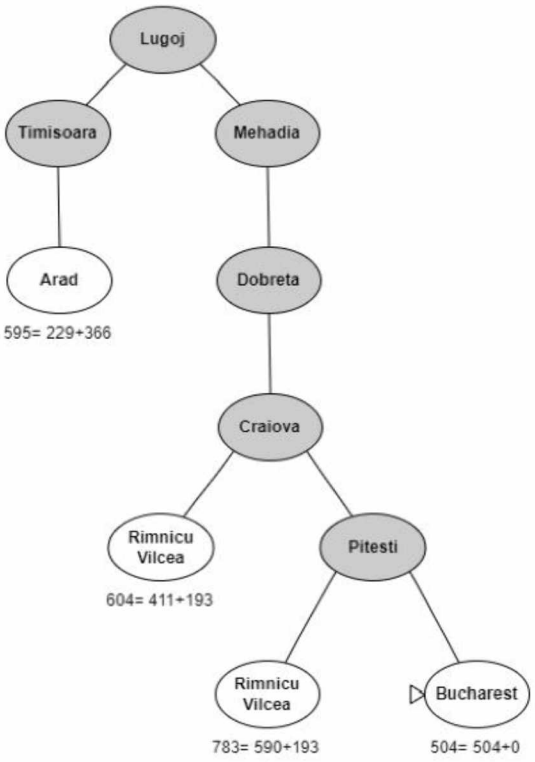
(e) Após expandir Craiova



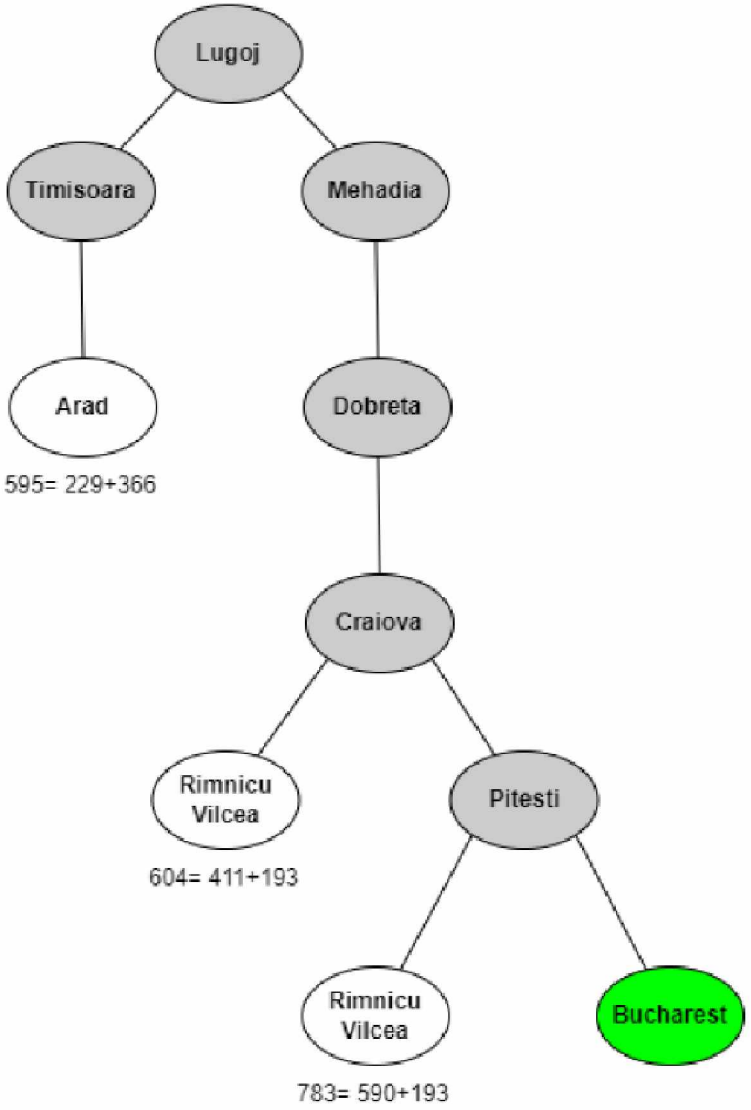
(f) Após expandir Timisoara



(g) Após expandir Pitesti



(h) Após expandir Bucharest



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

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem

q: a raposa come as uvas

r: as uvas estão maduras

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

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

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

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

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

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

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

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

Resposta:

Dado BC

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

Quer-se provar que

- $q \leftrightarrow p$

(a) Aplica-se equivalência Implicação em $R3$

$R4$: $r \rightarrow p$

Dado BC

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

Quer-se provar que

- $q \leftrightarrow p$

(b) Aplica-se Silogismo Hipotético em $R2$ e $R4$

$R5$: $q \rightarrow p$

Dado BC

- $R1$: $p \rightarrow q$
- $R2$: $q \rightarrow r$
- $R3$: $\neg r \vee p$
- $R4$: $r \rightarrow p$
- $R5$: $q \rightarrow p$

Quer-se provar que

- $q \leftrightarrow p$

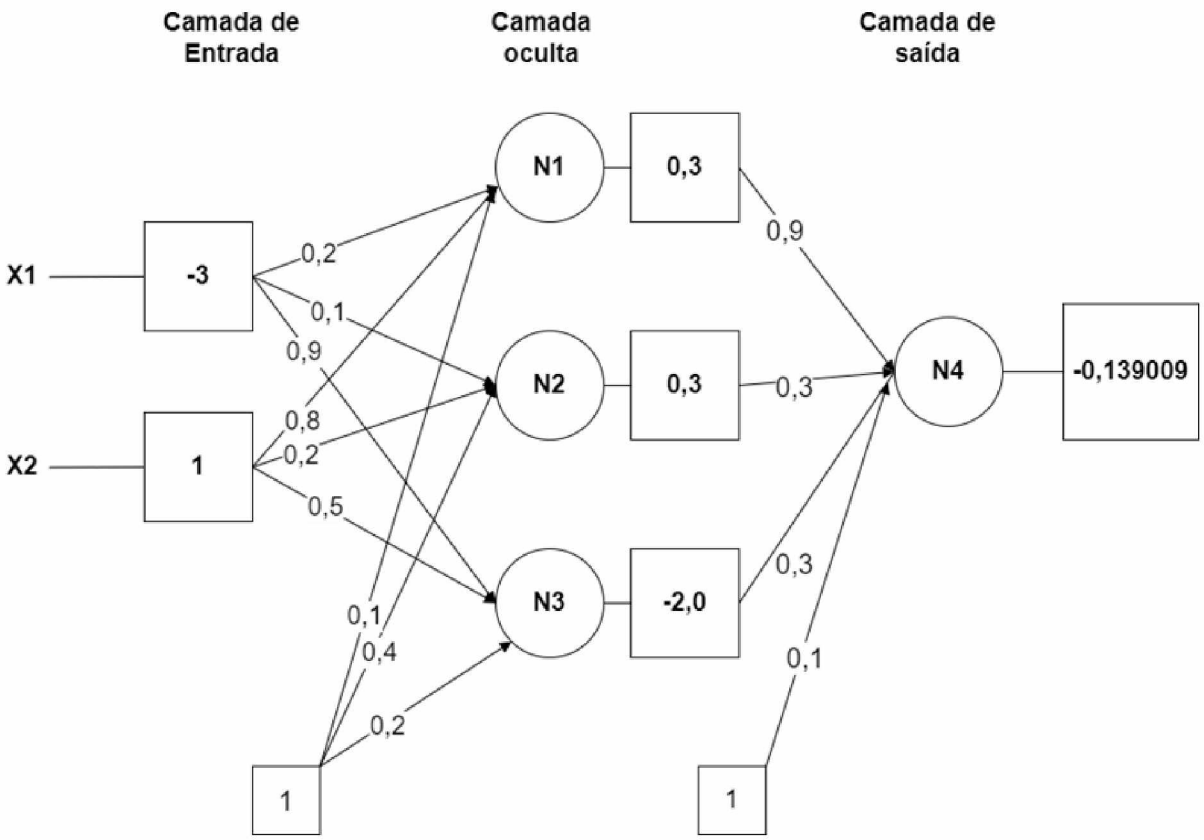
(c) Aplica-se Equivalência Bicondicional em $R5$ e $R1$

$R6$: $q \leftrightarrow p$

Dado BC

- $R1$: $p \rightarrow q$
- $R2$: $q \rightarrow r$
- $R3$: $\neg r \vee p$
- $R4$: $r \rightarrow p$
- $R5$: $q \rightarrow p$
- $R6$: $q \leftrightarrow p$

b)

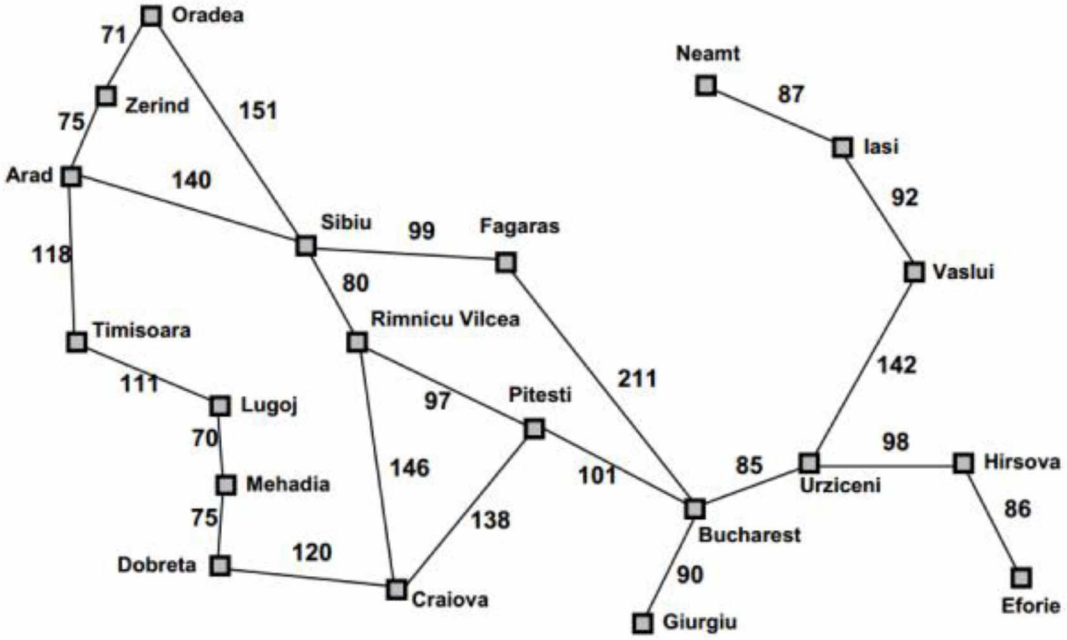


APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

Exercício de implementação do Algoritmo A*.

a. Implemente a função `a_star` do arquivo a_star.py presente no repositório.
b. O destino será sempre **Bucharest**.
c. O arquivo dists.py descreve estruturas de dados que representam as seguintes informações:



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

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

B - RESOLUÇÃO

```
class Dists:
    def __init__(self):
        self.straight_line_dists_from_bucharest = {
            'Arad': 366,
            'Bucharest': 0,
            'Craiova': 160,
            'Drobeta': 242,
            'Eforie': 161,
            'Fagaras': 176,
            'Giurgiu': 77,
            'Hirsova': 151,
            'Iasi': 226,
            'Lugoj': 244,
            'Mehadia': 241,
            'Neamt': 234,
            'Oradea': 380,
            'Pitesti': 100,
            'Rimnicu Vilcea': 193,
            'Sibiu': 253,
            'Timisoara': 329,
            'Urziceni': 80,
            'Vaslui': 199,
            'Zerind': 374
        }
        self.dists = {
            'Bucharest': [
                ('Urziceni',85),
                ('Giurgiu',90),
                ('Pitesti',101),
                ('Fagaras',211)
            ],
            'Giurgiu': [
                ('Bucharest',90)
            ],
            'Urziceni': [
                ('Bucharest',85),
                ('Hirsova',98),
                ('Vaslui',142)
            ],
            'Hirsova': [
                ('Urziceni',98),
                ('Eforie',86)
            ],
            'Eforie': [
                ('Hirsova',86)
            ],
            'Vaslui': [
                ('Urziceni',142),
                ('Iasi',92)
            ],
            'Iasi': [
                ('Vaslui', 92),
                ('Neamt',87)
            ],
            'Neamt': [
```

```
            ('Iasi', 87)
        ],
        'Fagaras': [
            ('Bucharest',211),
            ('Sibiu',99)
        ],
        'Pitesti': [
            ('Bucharest',101),
            ('Rimnicu Vilcea',97),
            ('Craiova',138)
        ],
        'Craiova':[
            ('Pitesti',138),
            ('Rimnicu Vilcea',146),
            ('Drobeta',120)
        ],
        'Rimnicu Vilcea':[
            ('Craiova',146),
            ('Pitesti',97),
            ('Sibiu',80)
        ],
        'Sibiu':[
            ('Rimnicu Vilcea',80),
            ('Fagaras',99),
            ('Oradea',151),
            ('Arad',140)
        ],
        'Oradea':[
            ('Sibiu',151),
            ('Zerind',71)
        ],
        'Zerind':[
            ('Oradea',71),
            ('Arad',75)
        ],
        'Arad':[
            ('Zerind',75),
            ('Timisoara',118),
            ('Sibiu',140)
        ],
        'Timisoara':[
            ('Arad',118),
            ('Lugoj',111)
        ],
        'Lugoj':[
            ('Timisoara',111),
            ('Mehadia',70)
        ],
        'Mehadia':[
            ('Lugoj',70),
            ('Drobeta',75)
        ],
        'Drobeta':[
            ('Mehadia',75),
            ('Craiova',120)
        ],
    }
```

```
dists = Dists()

class Node:
    """
    Representa um nó (cidade) na árvore de busca.
    """
    def __init__(self, cidade, dist_linha_reta, origem, conexoes):
        self.cidade = cidade
        self.dist_linha_reta = dist_linha_reta
        self.origem = origem
        self.conexoes = conexoes

class AStar:
    """
    Implementa o algoritmo de busca A* para encontrar o caminho mais
curto.
    """
    def __init__(self):
        # Inicializa os atributos da classe AStar
        self.straight_line_list =
dists.straight_line_dists_from_bucharest
        self.connections_list = dists.dists
        self.great_path = []
        self.explored = []
        self.walked_distance = 0
        self.node = None
        self.border = []

    def add_city_on_border(self, tup_city):
        """
        Adiciona uma cidade na borda de exploração, considerando seu
custo f(n) = g(n) + h(n).
        Evita adicionar cidades já exploradas ou com custo maior se já
estiverem na borda.
        """
        city, distance_from_current = tup_city
        validation = True

        # Verifica se a cidade já foi explorada
        for visited_city in self.explored:
            if visited_city == city:
                validation = False
                break # Sai do loop se a cidade já foi explorada

        # Verifica se a cidade já está na borda com um custo menor
        if validation:
            current_g_n = self.walked_distance + distance_from_current
            current_f_n = current_g_n + self.straight_line_list[city]

            for border_city_data in self.border:
                name_border_city = border_city_data.get('cidade')
                if name_border_city == city:
                    if border_city_data.get('custo') < current_f_n:
                        validation = False
```

```
                        break # Sai do loop se já existe na borda com
custo menor

        # Se a cidade é válida para ser adicionada ou atualizada na
borda
        if validation:
            cidade = city
            origem = self.node.cidade # A cidade de onde viemos
            # Calcula o custo f(n) = g(n) + h(n)
            custo = (self.walked_distance + distance_from_current) +
self.straight_line_list[city]

            # Remove a cidade da borda se ela já existe e esta nova
rota é melhor
            self.remove_city_on_border(city) # Remove para evitar
duplicatas com custo pior

            self.border.append({"cidade": cidade, "origem": origem,
"custo": custo})
            # Mantém a borda ordenada pelo custo (f(n)) para sempre
pegar o menor
            self.border.sort(key=lambda x: x['custo'])

    def remove_city_on_border(self, city):
        """
        Remove uma cidade da borda. Usado quando uma cidade é explorada
        ou quando uma rota melhor para ela é encontrada.
        """
        # Cria uma nova lista sem a cidade a ser removida
        self.border = [b_city for b_city in self.border if
b_city.get('cidade') != city]

    def move_to_next_city(self):
        """
        Muda o nó atual para a cidade na borda com o menor custo f(n).
        Atualiza a distância percorrida (g(n)).
        """
        if not self.border:
            # Não há mais cidades para explorar, caminho não encontrado
ou erro.
            # Em um algoritmo A* completo, isso indicaria falha.
            return False

        # Pega a cidade com menor custo da borda (já ordenada)
        next_city_data = self.border.pop(0) # Remove e retorna o
primeiro elemento

        city = next_city_data.get('cidade')
        origin = next_city_data.get('origem')

        # Atualiza a distância percorrida (g(n)) para o novo nó
        # Encontra a distância do nó de origem para a cidade atual
        distance_from_origin_to_current = 0
        for connection in self.connections_list.get(origin, []):
            con_city, dist = connection
            if con_city == city:
                distance_from_origin_to_current = dist
```

```
                break

        # Se o nó atual é o de partida, walked_distance é 0
        if origin == '': # É o nó inicial
            self.walked_distance = 0
        else:
            self.walked_distance = next_city_data.get('custo') -
self.straight_line_list[city]


        # Atualiza o nó atual
        self.node = Node(city, self.straight_line_list[city], origin,
self.connections_list[city])

        # Adiciona a cidade atual ao caminho (e remove duplicatas, se
houver)
        if city not in self.great_path:
            self.great_path.append(city)

        return True # Indica que a movimentação foi bem-sucedida

    def search_best_path(self, start, goal='Bucharest'):
        """
        Retorna uma lista com o caminho de 'start' até 'goal'
        segundo o algoritmo A*.
        """

        self.great_path = []
        self.explored = []
        self.walked_distance = 0

        # Inicializa o nó de partida
        self.node = Node(start, self.straight_line_list[start], '',
self.connections_list[start])

        # Adiciona o nó inicial à borda
        self.border = [{"cidade": start, "origem": '', "custo":
self.straight_line_list[start]}]

        # Adiciona o nó inicial ao caminho
        self.great_path.append(self.node.cidade)

        # Se a cidade de partida já é o objetivo
        if self.node.cidade == goal:
            return self.great_path

        # Loop principal do algoritmo A*
        while self.node.cidade != goal:
            if not self.border:
                # Se a borda está vazia e o objetivo não foi
encontrado,
                # significa que não há caminho.
                print(f"Não foi possível encontrar um caminho de
{start} para {goal}.")
                return []
```

```
            self.explored.append(self.node.cidade) # Adiciona o nó
atual aos explorados

            # Adiciona os vizinhos do nó atual à borda
            for city_connection in self.node.conexoes:
                self.add_city_on_border(city_connection)

            # Move para o próximo nó com menor custo na borda
            if not self.move_to_next_city():
                print(f"Não foi possível encontrar um caminho de
{start} para {goal}.")
                return []

        # Quando o objetivo é alcançado, retorna o caminho
        return self.great_path

    def get_best_path_string(self, start):
        """
        Retorna na tela uma lista com o caminho de 'start' até 'goal'
        segundo o algoritmo A*.
        """
        print("O melhor caminho encontra-se na sequencia de cidades:")
        path = self.search_best_path(start)
        if path:
            print(" -> ".join(path))
        else:
            print("Caminho não encontrado.")

# Bloco principal para execução do código
if __name__ == '__main__':
    a_star = AStar()

    # Exemplo de uso:
    nome_da_cidade = input("Digite o nome da cidade: ")
    caminho = a_star.get_best_path_string(nome_da_cidade)
    # a_star.get_best_path_string('Arad')
    # a_star.get_best_path_string('Sibiu')
```

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 * dap^2 * Ht

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

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

Tarefas

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

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

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

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

    - Coeficiente de determinação: R^2

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

- **Exercício 1 - Satélites**

Carregamento e visualização inicial do dataset:

```
library("mlbench")
data(Satellite)
dataset <- Satellite
head(dataset)
```

```
  x.1 x.2 x.3 x.4 x.5 x.6 x.7 x.8 x.9 x.10 x.11 x.12 x.13 x.14 x.15 x.16
x.17 x.18 x.19 x.20 x.21 x.22 x.23 x.24 x.25 x.26 x.27 x.28 x.29 x.30 x.31
x.32 x.33 x.34 x.35 x.36 classes
1  92 115 120  94  84 102 106  79  84 102 102  83 101 126 133 103  92 112
118  85  84 103 104  81 102 126 128 100  84 107 113  87  84 104  79 grey
soil
2  84 102 106  89  84 102 102  83  80 102  79  92 112 104  81  84 103 104
78  88 121 104  81  84 107  84  99 104  78  88 121 104  81  84 107  grey
soil
3  84 102 102  79  84 102  79  84  94 102  79  84 103  84  94  98  76  84
99  84  94  98  76  84  99  84  94  98  76  84  99  84  94  98  76  84  99
4  84 102 102  79  84 102  79  84  94 102  79  84 103  84  94  98  76  84
99  84  94  98  76  84  99  84  94  98  76  84  99  84  94  98  76  84  99
```

```
tail(dataset)
```

```
     x.1 x.2 x.3 x.4 x.5 x.6 x.7 x.8 x.9 x.10 x.11 x.12 x.13 x.14 x.15 x.16
x.17 x.18 x.19 x.20 x.21 x.22 x.23 x.24 x.25 x.26 x.27 x.28 x.29 x.30 x.31
x.32 x.33 x.34 x.35 x.36 classes
6430 71 100 85 74 83 104 92 78 96 112 96 red soil
6431 91 104 82 66 87 108 89 63 83 104 85 red soil
6432 83 100 85 66 83 102 83 63 83 100 81 red soil
6433 83 100 81 59 87  96 81 63 83  92 74 vegetation stubble
```

```
6434 83  96 74 59 83  92 74 59 83  92 70 vegetation stubble
6435 83  92 74 59 83  92 70 63 79 108 92 vegetation stubble
```

- **Random Forest**

```
indices <- createDataPartition(dataset$classes, p=0.8, list=F)
treino <- dataset[indices,]
teste <- dataset[-indices,]

set.seed(1)
rf <- train(classes ~ x.17 + x.18 + x.19 + x.20, data=treino, method="rf")
predicao.rf <- predict(rf, teste)
confusionMatrix(predicao.rf, teste$classes)
```

```
Confusion Matrix and Statistics
              Reference
Prediction     red soil cotton crop grey soil damp grey soil vegetation
stubble very damp grey soil
red soil             300           2          2              7
0
cotton crop            0         133          0              4
0
grey soil              0           0        253             28
3
damp grey soil         0          12         60              0
40
vegetation stubble     1           1          1            115
5
very damp grey soil    0           3         34             15
253

Overall Statistics
Accuracy : 0.8676
95% CI : (0.8478, 0.8857)
No Information Rate : 0.2383
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.8361

Statistics by class:
Sensitivity, Specificity, Pos Pred Value, Neg Pred Value, Prevalence,
Detection Rate, Balanced Accuracy
(red soil: 0.9084, cotton crop: 0.9500, grey soil: 0.9336, damp grey soil:
0.4800, vegetation stubble: 0.8816)
```

Avaliação do resultado: a precisão do modelo foi de 86,76%, o que é considerado positivo, pois indica que o modelo não está sofrendo de overfitting. No entanto, o modelo apresentou dificuldade em reconhecer a classe de solo cinza úmido.

- **SVM**

```
svm <- train(classes ~ x.17 + x.18 + x.19 + x.20, data=treino,
method="svmRadial")
predicao.svm <- predict(svm, teste)
confusionMatrix(predicao.svm, teste$classes)
```

```
Confusion Matrix and Statistics
Overall Statistics
Accuracy : 0.8707
95% CI : (0.8511, 0.8886)
No Information Rate : 0.2383
P-Value [Acc > NIR] : < 2.2e-16
Kappa : 0.8398
```

Avaliação do resultado: o modelo obteve acurácia de 87,07%, o que é positivo, pois indica que o modelo não está sofrendo de overfitting. O modelo apresentou menor dificuldade em reconhecer a classe damp grey soil.

- **RNA**

```
set.seed(1)
rna <- train(classes ~ x.17 + x.18 + x.19 + x.20, data=treino,
method="nnet")
predicao.rna <- predict(rna, teste)
confusionMatrix(predicao.rna, teste$classes)
```

```
Confusion Matrix and Statistics
Accuracy : 0.8217
95% CI : (0.7996, 0.8422)
Kappa : 0.7761
```

Avaliação do resultado: o modelo obteve acurácia de 82,17%, o que é positivo, pois indica que o modelo não está sofrendo de overfitting. O modelo apresentou maior dificuldade em reconhecer a classe damp grey soil.

Resultado: o melhor modelo apresentado foi o SVM, devido à sua maior acurácia.

Exercício 2 - Árvores

Carregamento de pacotes e leitura do conjunto de dados de volumes:

```
library("caret")
# Carregando pacotes exigidos: ggplot2, lattice
# Warning message:
# package 'caret' was built under R version 4.2.3

df <- read.csv("http://www.razer.net.br/datasets/volumes.csv", sep=";",
dec=".")
df$NR <- NULL
head(df)
```

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

```
tail(df)
    DAP    HT   HP      VOL
95  31.5 23.50 2.50 0.9221653
```

```
96  33.1 25.75 2.65 1.0966956
97  31.0 25.70 2.60 1.0514350
98  43.0 27.90 2.70 2.0090605
99  40.0 24.20 1.10 1.7411209
100 38.0 27.65 2.45 1.5336724
```

```
indices <- createDataPartition(df$VOL, p=0.8, list=F)
treino <- df[indices,]
teste <- df[-indices,]
```

1 Random Forest

```
set.seed(1)
rf <- train(VOL ~ ., data=treino, method="rf")
```

```
predicao.rf <- predict(rf, teste)
summary(predicao.rf)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
# 0.8508  1.0936  1.2212  1.3690  1.6106  2.3515
```

2 SVM

```
set.seed(1)
svm <- train(VOL ~ ., data=treino, method="svmRadial")
predicao.svm <- predict(svm, teste)
```

3 Redes neurais

```
set.seed(1)
# install.packages("neuralnet")  # pacote instalado durante a execução
original
library(neuralnet)
rna <- neuralnet(VOL ~ ., data=treino, hidden=c(2,1), linear.output=FALSE,
threshold=0.01)
predicao.rna <- predict(rna, teste)
```

4 Modelo alométrico de SPURR

```
set.seed(1)
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, data=treino, start=list(b0=0.5,
b1=0.5))
predicao.alom <- predict(alom, teste)
```

5 Criação de funções de métricas

```
r2 <- function(valor_observado, valor_predito){
  return(1 - (sum((valor_observado-valor_predito)^2)/sum((valor_observado-
mean(valor_observado))^2)))
}

syx_porc <- function(valor_observado, valor_predito){
  return(sqrt(sum((valor_observado-
valor_predito)^2)/mean(valor_observado)*100))
}

syx <- function(valor_observado, valor_predito){
  return(sqrt(sum((valor_observado-
valor_predito)^2/(length(valor_observado)-2))))
}
```

6 Resultados das métricas

R^2

```
r2(teste$VOL, predicao.rf)
# [1] 0.8879647

r2(teste$VOL, predicao.svm)
# [1] 0.7985581

r2(teste$VOL, predicao.rna)
# [1] -0.8725625

r2(teste$VOL, predicao.alom)
# [1] 0.8654416
```

Avaliação: O modelo Random Forest se saiu melhor que os demais por apresentar R^2 mais próximo de 1. Observação: a RNA apresentou valor negativo de R^2, indicando mau ajuste no cenário avaliado.

Erro padrão da estimativa (syx)

```
syx(teste$VOL, predicao.rf)
# [1] 0.1491092

syx(teste$VOL, predicao.svm)
# [1] 0.199941

syx(teste$VOL, predicao.rna)
# [1] 0.6096002

syx(teste$VOL, predicao.alom)
# [1] 0.1634114
```

Avaliação: O Random Forest apresentou melhor desempenho em comparação aos demais, com menor erro padrão da estimativa (mais próximo de 0). A RNA obteve o pior resultado.

Erro padrão da estimativa em % (syx_porc)

```
syx_porc(teste$VOL, predicao.rf)
# [1] 10.69305

syx_porc(teste$VOL, predicao.svm)
# [1] 14.33835

syx_porc(teste$VOL, predicao.rna)
# [1] 43.7162

syx_porc(teste$VOL, predicao.alom)
# [1] 11.71871
```

Após avaliar os resultados, foi escolhido o modelo Random Forest por apresentar as melhores métricas (R^2 mais alto e menores syx/syx_porc).

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

Com a base de dados “realestateiaa” obter os seguintes resultados com o auxílio do “R”

a) Elaborar o histograma e o boxplot das variáveis “parea e tarea”.

b) Elaborar a tabela de distribuição de frequências da variável “price” (preço dos imóveis);

c) Para a variável “price” calcular os seguintes indicadores: média; mediana; moda; variância; desvio padrão; CV–Coeficiente de Variação; Quartis; distância interquartílica; percentis.

d) Estimar o intervalo de confiança para a média da variável “price” com 95% de confiança

e) Fazer o teste de diferença entre médias para as variáveis “parea” e “tarea”.

f) Fazer o teste de diferença entre variâncias para as variáveis “parea” e “tarea”.

g) Fazer o Teste de Wilcoxon-Mann-Whitney para amostras independentes para as variáveis “parea” e “tarea”.

h) Fazer 2 testes de normalidade (a sua escolha) para a variável “price”.

Apresentar esses resultados em um documento pdf. Não é para postar a rotina, mas sim a saída (resultados), fazer a interpretação dos resultados. Fazer upload do documento pdf no “ufprvirtual”, na tarefa aberta no Tópico 1.

Com a base de dados “imoveiscwbav” obter os seguintes resultados com o auxílio do “R”

a) Estimar um modelo preliminar e apresentar os resultados;

b) Testar as variáveis para formulação do modelo;

c) Verifique a presença de outliers;

d) Teste a especificação do modelo e altere se necessário;

e) Teste a presença de multicolinearidade e exclua variáveis se necessário;

f) Selecione um modelo pela técnica de stepwise;

g) Faça o teste de homocedasticidade e faça correção da heterocedasticidade se necessário;

h) Obtenha os indicadores de desempenho do modelo;

i) Estime os intervalos de confiança para os parâmetros do modelo;

j) Faça predição de um imóvel hipotético: apresente seus parâmetros de simulação e o resultado.

Apresentar esses resultados em um documento pdf. Não é para postar a rotina, mas sim a saída (resultados), fazer a interpretação dos resultados. Fazer upload do documento pdf no "ufprvirtual", na tarefa aberta no Tópico 2.

- **Estatística I - Primeira Lista de Exercício**

- **a) Histograma e Boxplot das Variáveis "parea" e "tarea"**

O boxplot e o histograma foram elaborados para as variáveis "parea" e "tarea".

- **Boxplot Parea:** Nos gráficos de frequência, nota-se que existem muitos *outliers* na variável parea. * **Histograma Parea:** Nos histogramas, é visível que existem muitos valores de parea entre 100 e 200. * **Histograma Tarea:**

- **b) Tabela de Distribuição de Frequências da Variável "price"**

A tabela de distribuição de frequências para a variável "price" (preço dos imóveis) foi elaborada.

class limits	fr	rf	rf (%)	cf	cf (%)
[157410, 591062.5)	279	0.29	28.56	279	28.56
[591062.5, 1024715)	288	0.29	29.48	567	58.03
[1024715, 1458368)	162	0.17	16.58	729	74.62
[1458368, 1892020)	117	0.12	11.98	846	86.59
[1892020, 2325673)	53	0.05	5.42	899	92.02
[2325673, 2759325)	28	0.03	2.87	927	94.88

[2759325, 3192978)	17	0.02	1.74	944	96.62
[3192978, 3626630)	8	0.01	0.82	952	97.44
[3626630, 4060283)	14	0.01	1.43	966	98.87
[4060283, 4493935)	8	0.01	0.82	974	99.69
[4493935, 4927588)	3	0.00	0.31	977	100.00

Exportar para as Planilhas

- A maior frequência é entre **591062.5** e **1024715.**

- **c) Indicadores para a Variável "price"**

Os seguintes indicadores foram calculados para a variável "price": média, mediana, moda, variância, desvio padrão, Coeficiente de Variação (CV), quartis, distância interquartílica e percentis.

- **1. Medidas de Tendência Central**

Indicador	Código R	Saída
Média	`> mean(realestateiaasprice)`	`[1] 1140123`
Mediana	`> median(realestateiaašprice)`	`[1] 950000`
Moda	`> subset (table (realestateiaašprice), table(realestateiaašprice) == max(table (realestateiaašprice)))`	`850000 16`

Exportar para as Planilhas

- **2. Medidas de Dispersão**

Indicador	Código R	Saída

Variância	> var (realestateiaasprice)	[1] 670486776668
Desvio Padrão	> sd (realestateiaasprice)	[1] 818832.6
CV	> (sd(realestateiaašprice) / mean(realestateiaašprice)) * 100	[1] 71.81967

Exportar para as Planilhas

- **3. Medidas Separatrizes**

Indicador	Código R	Saída
Q1 (25%)	> quantile(realestateiaasprice, probs = 0.25)	550000
Q2 (50%)	> quantile(realestateiaašprice, probs = 0.50)	950000
Q3 (75%)	> quantile(realestateiaasprice, probs = 0.75)	1485000
Distância Interquartílica	> IQR(realestateiaašprice)	[1] 935000

Exportar para as Planilhas

- **Percentis (10% a 90%)**

R

```
> quantile(realestateiaašprice, c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9))
```

Percentil	Saída
10%	353000.0

20%
490000.0

30%
607056.8

40%
799400.0

50%
950000.0

60%
1090000.
0

70%
1366000.
0

80%
1678800.
0

90%
2123200.
0

- **d) Intervalo de Confiança (95%) para a Média de "price"**

O intervalo de confiança para a média de "price" foi estimado com 95% de confiança, usando um teste Z *one-sample*.

```
R
one-sample z-Test
data: realestateiaasprice
$z=43.522$ p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
1088778 1191468
sample estimates:
mean of x
1140123
```

- O intervalo de confiança com 95% está entre **1088778** e **1191468.**

- **e) Teste de Diferença Entre Médias ("parea" e "tarea")**

O teste de diferença entre médias para as variáveis "parea" e "tarea" foi realizado, usando um teste Z *Two-sample*.

```
R
Two-sample z-Test
data: realestateiaaSparea and realestateiaastarea
$z=-14.151$, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-94.09729 -71.20261
sample estimates:
mean of x mean of y
152.9867 235.6366
```

- A diferença entre médias ("parea" e "tarea") encontra-se entre **152.9867** e **235.6366** (Nota: esta afirmação no documento parece citar as médias amostrais, não o IC ou a diferença de médias).

- **f) Teste de Diferença Entre Variâncias ("parea" e "tarea")**

O teste F foi utilizado para comparar as variâncias das variáveis "parea" e "tarea".

```
R
F test to compare two variances
data: realestateiaaSparea and realestateiaastarea
$F=0.43855$ , num df $=976$, denom df $=976$, p-value $<2.2e-16$
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
0.3868074 0.4972143
sample estimates:
ratio of variances
0.4385501
```

- A diferença entre variâncias (razão das variâncias) é de **0.4385501.**

- **g) Teste de Wilcoxon-Mann-Whitney ("parea" e "tarea")**

Foi realizado o teste de Wilcoxon-Mann-Whitney para amostras independentes.

```
R
wilcoxon rank sum test with continuity correction
data: realestateiaasparea and realestateiaastarea
```

```
W 287419, p-value < 0.00000000000000022
alternative hypothesis: true location shift is not equal to 0
```

- O teste atesta que as amostras são independentes.

- **h) 2 Testes de Normalidade ("price")**

Dois testes de normalidade foram realizados para a variável "price".

- **1. Teste de Kolmogorov-Smirnov (Lilliefors)**

```
R
Lilliefors (Kolmogorov-Smirnov) normality test
data: price
$D=0.14486$ , p-value < 0.00000000000000022
```

- No teste, o resultado mostra que **não se deve rejeitar a normalidade** da amostra.

- **2. Teste de Anderson-Darling**

Hipóteses:

- H0: Os dados seguem a distribuição normal
- H1: Os dados não seguem a distribuição normal

```
R
Anderson-Darling normality test
data: realestateiaaSprice
$A=35.862$, p-value < 0.00000000000000022
```

- Como p-value=0.00000000000000022<0.05, há evidências para rejeitar a hipótese H0, logo, os dados **não seguem a distribuição normal.**

- **Estatística I: Segunda Lista de Exercício (Modelagem de Regressão)**

- **a) Estimar um Modelo Preliminar e Apresentar os Resultados**

O modelo de regressão linear múltipla (OLS) preliminar foi ajustado usando todas as variáveis explicativas disponíveis (age até categ) para prever price.

- **Código**

```
R
> load("C:/Users/bruno/Documents/Facul/lista2/imoveiscwbav.RData")
```

```
> resultados <- lm(price ~ age + parea + tarea + bath + ensuit + garag +
plaz + park +
  trans + kidca + school + health + bike + barb + balc + elev +
  fitg + party + categ, data = imoveiscwbav)
> summary (resultados)
```

- **Saída: Coeficientes e Resumo do Modelo Preliminar**

| Coeficiente | Estimate | Std. Error | t value | Pr(>\|t\|) | |:---:|:---:|:---:|:---:|:---:| | **(Intercept)** | -420453.5 | 130052.5 | -3.233 | 0.0013 * | | **age** | -7839.1 | 1025.3 | -7.645 | 1.01e-13 *** | | **parea** | 2592.2 | 624.0 | 4.154 | 3.82e-05 *** | | **tarea** | 1975.8 | 333.9 | 5.918 | 5.91e-09 *** | | **bath** | 13452.6 | 14832.9 | 0.907 | 0.3649 | | **ensuit** | 125949.6 | 18560.7 | 6.786 | 3.15e-11 *** | | **garag** | 169687.5 | 21756.1 | 7.800 | 3.41e-14 *** | | **plaz** | 224393.0 | 94219.1 | 2.382 | 0.0176 * | | **park** | -63439.6 | 27154.0 | -2.336 | 0.0199 * | | **trans** | 26642.3 | 22718.5 | 1.173 | 0.2414 | | **kidca** | 10452.8 | 34899.8 | 0.300 | 0.7647 | | **school** | -7975.8 | 56635.7 | -0.141 | 0.8881 | | **health** | 1217.4 | 56216.5 | 0.022 | 0.9827 | | **bike** | -85864.4 | 56073.0 | -1.531 | 0.1263 | | **barb** | -43925.7 | 22602.3 | -1.943 | 0.0525 . | | **balc** | 65144.8 | 25242.3 | 2.581 | 0.0101 * | | **elev** | -111743.4 | 25295.0 | -4.418 | 1.21e-05 *** | | **fitg** | 123052.7 | 28456.0 | 4.324 | 1.83e-05 *** | | **party** | 36463.1 | 28481.1 | 1.280 | 0.2010 | | **categ** | 283061.5 | 55653.0 | 5.086 | 5.11e-07 *** |

Métrica	Valor
Residual standard error	229400 on 521 degrees of freedom
Multiple R-squared	0.8099
Adjusted R-squared	0.803
F-statistic	116.8 on 19 and 521 DF, p-value: < 2.2e-16

- **b) Testar as Variáveis para Formulação do Modelo (Teste Pan-Jenes)**

O teste de **Pan-Jenes** (fform) é realizado para avaliar se transformações não-lineares de variáveis melhoram o modelo (critério AIC/BIC). A variável com o menor AIC/BIC é selecionada.

- **Teste Age**

R

```
> formBase<-formula(price ~ parea + tarea + bath + ensuit + garag + plaz +
park +
  trans + kidca + school + health + bike + barb + balc + elev +
  fitg + party + categ)
> PanJenage<-fform(imoveiscwbav, "age", formBase)
```

Transformação	AIC	BIC	Ranking (BIC)
smoothing	14879.74	14977.41	1
x^2	14897.14	14987.30	2
log(x)	14901.06	14991.22	3
X	14912.25	15002.42	4
x+x^2	14943.68	15033.85	5
sqr(x)	14944.35	15034.51	6
1/x	14957.80	15047.96	7
base	14967.78	15053.65	8

Decisão: Incluir Age (Transformação selecionada: age2 é o próximo após *smoothing* que é semi-paramétrico, mas o código final usa age2).

- **Teste Parea**

```
R
# formBase é atualizado para incluir age
formBase<-formula(price ~ age + tarea + bath + ensuit + garag + plaz + park
+
  trans + kidca + school + health + bike + barb + balc + elev +
  fitg + party + categ)
> PanJenparea<-fform(imoveiscwbav, "parea", formBase)
```

Transformação	AIC	BIC	Ranking (BIC)
sqr(x)	14904.77	14994.94	1
x+x^2	14904.80	14994.96	2
X	14912.25	15002.42	3
smoothing	14881.96	15002.43	4

x^2	14916.70	15006.87	5
log(x)	14921.03	15011.19	6
base	14927.88	15013.75	7
1/x	14929.78	15019.94	8

Exportar para as Planilhas

Decisão: Incluir Parea (Transformação selecionada: parea (*sqr(x)*)).

- **Teste Tarea**

```
R
# formBase é atualizado para incluir age e parea
formBase<-formula(price ~ age + parea + bath + ensuit + garag + plaz + park
+
  trans + kidca + school + health + bike + barb + balc + elev +
  fitg + party + categ)
> PanJentarea<-fform(imoveiscwbav, "tarea", formBase)
```

Transformação	AIC	BIC	Ranking (BIC)
smoothing	14863.46	14963.98	1
sqr(x)	14886.91	14977.07	2
x+x^2	14886.96	14977.12	3
X	14912.25	15002.42	4
x^2	14925.17	15015.33	5
log(x)	14935.80	15025.96	6
base	14945.45	15031.32	7
1/x	14947.12	15037.28	8

Exportar para as Planilhas

Decisão: Incluir Tarea (Transformação selecionada: tarea (*sqr(x)*), é a segunda melhor após *smoothing*).

- **Teste Plaz, Park, Trans, Kidca, School, Health, Bike**

  - **Teste Plaz:** log(x) (AIC 14910.84) vs. base (AIC 14916.11). **Decisão:** Não incluir Plaz (o melhor critério não supera o modelo base em termos de BIC no ranking).
  - **Teste Park:** 1/x (AIC 14905.25) vs. base (AIC 14915.89). **Decisão:** Incluir Park (a melhor transformação é 1/x).
  - **Teste Trans:** 1/x (AIC 14906.86) vs. base (AIC 14911.68). **Decisão:** Não incluir Trans (o melhor critério não supera o modelo base em termos de BIC no ranking).
  - **Teste Kidca:** base (AIC 14910.35) vs. log(x) (AIC 14912.07). **Decisão:** Não incluir Kidca (o melhor critério é a transformação base).
  - **Teste School:** base (AIC 14910.28) vs. 1/x (AIC 14908.59). **Decisão:** Não incluir School (o melhor critério não supera o modelo base em termos de BIC no ranking).
  - **Teste Health:** base (AIC 14910.26) vs. 1/x (AIC 14911.26). **Decisão:** Não incluir Heath (o melhor critério é a transformação base).
  - **Teste Bike:** log(x) (AIC 14910.52) vs. base (AIC 14912.68). **Decisão:** Não incluir Bike (o melhor critério não supera o modelo base em termos de BIC no ranking).

Nota: As transformações escolhidas e aplicadas no código final são: age2 , parea , tarea , log(plaz) , 1/park e 1/trans.

- **c) e f) Modelo Final Após Transformações e Stepwise**

As transformações foram aplicadas. O modelo é então reajustado e o **stepwise** é realizado para a seleção final de variáveis, culminando no **Melhor Modelo.**

- **Código de Transformação e Ajuste do Modelo**

```
R
# Aplicação das transformações selecionadas
> imoveiscwbav$age <- (imoveiscwbav$age^2)
> imoveiscwbav$parea <- sqrt(imoveiscwbav$parea)
> imoveiscwbav$tarea <- sqrt(imoveiscwbav$tarea)
> imoveiscwbav$plaz <- log(imoveiscwbav$plaz)
> imoveiscwbav$park <- 1/(imoveiscwbav$park)
> imoveiscwbav$trans <- 1/(imoveiscwbav$trans)

# Modelo após transformações e antes do Stepwise
```

```
resultados <- lm(price ~ age + parea + tarea + bath + ensuit + garag + plaz
+ park +
  trans + kidca + school + health + bike + barb + balc + elev +
  fitg + party + categ, data = imoveiscwbav)

# Stepwise (Direction: backward/forward, Criterion: AIC)
> step <- stepwise (resultados, direction= 'backward/forward', criterion =
'AIC')

# Melhor Modelo Selecionado pelo Stepwise (baseado em AIC mínimo)
> resultados <-lm(price ~ age + parea + tarea + ensuit + garag + plaz +
  park + trans + bike + barb + balc + elev + fitg + categ, data =
imoveiscwbav)

> summary(resultados)
```

- **Saída: Melhor Modelo (Final)**

O modelo final exclui bath, kidca, school, health, e party.

| Coeficiente | Estimate | Std. Error | t value | Pr(>\|t\|) | |:---:|:---:|:---:|:---:|:---:| | **(Intercept)** | -1.046e+06 | 1.252e+05 | -8.352 | 5.98e-16 *** | | **age** | -1.302e+02 | 2.476e+01 | -5.258 | 2.12e-07 *** | | **parea** | 4.591e+04 | 1.420e+04 | 3.234 | 0.001297 ** | | **tarea** | 5.525e+04 | 9.650e+03 | 5.725 | 1.74e-08 *** | | **ensuit** | 1.543e+05 | 1.580e+04 | 9.763 | < 2e-16 *** | | **garag** | 1.894e+05 | 2.200e+04 | 8.610 | < 2e-16 *** | | **plaz** | 4.858e+04 | 1.681e+04 | 2.889 | 0.004024 ** | | **park** | 1.480e+05 | 3.955e+04 | 3.743 | 0.000202 *** | | **trans** | -8.332e+04 | 3.999e+04 | -2.084 | 0.037684 * | | **bike** | -8.097e+04 | 5.513e+04 | -1.469 | 0.142511 | | **barb** | -3.314e+04 | 2.316e+04 | -1.431 | 0.153088 | | **balc** | 8.671e+04 | 2.495e+04 | 3.475 | 0.000553 *** | | **elev** | -9.766e+04 | 2.443e+04 | -3.998 | 7.30e-05 *** | | **fitg** | 1.414e+05 | 2.640e+04 | 5.354 | 1.29e-07 *** | | **categ** | 2.699e+05 | 5.435e+04 | 4.966 | 9.26e-07 *** |

Métrica	Valor
Residual standard error	236800 on 526 degrees of freedom
Multiple R-squared	0.7955
Adjusted R-squared	0.7901

- **d) Teste de Especificação do Modelo (RESET Test)**

Avalia a inclusão de termos não lineares (y^2 e y^3) no modelo final.

- **Código e Saída**

R

```
> resettest (price ~ age + parea + tarea + bath + ensuit + garag + plaz +
park +
  trans + kidca + school + health + bike + barb + balc + elev +
  fitg + party + categ, power = 2:3, type = "regressor", data =
imoveiscwbav)

RESET test
data: price ~ age + parea + tarea + bath + ensuit + garag + plaz + park +
trans + kidca + school + health + bike + barb + balc + elev + fitg + party
+ categ
RESET F = 5.2815, df1 = 38, df2 = 483, p-value < 2.2e-16

> qf (0.95, df1 = 38, df2 = 483)
[1] 1.429987
```

- **F Calculado (5.2815)** > **F Tabelado (1.429987),** e p-value<0.05.
- **Conclusão:** Existe erro de especificação do modelo.

- **e) Teste de Multicolinearidade (VIF)**

Verifica a correlação entre as variáveis explicativas.

- **Código e Saída**

R

```
> library(car)
> vif(lm(price ~ age + parea + tarea + bath + ensuit + garag + plaz + park
+ trans + kidca + school + health + bike, data = imoveiscwbav), type =
"high-order")
```

- **Conclusão:** A matriz de correlação mostra que parea e tarea são correlacionadas.
- O **Valor de Inflação de Variância (VIF)** para parea e tarea é alto (próximo de 4), indicando correlação considerável. A sugestão inicial é excluir tarea, mas a decisão é postergada para após o *stepwise*.

- **g) Teste de Homocedasticidade (Breusch-Pagan)**

Avalia se a variância dos resíduos é constante.

- **Código e Saída**

R

```
# Modelo após Stepwise (o modelo final selecionado)
> bptest(price ~ age + parea + tarea + ensuit + garag + plaz + park + trans
+ bike + barb + balc + elev + fitg + categ, studentize = FALSE, data =
imoveiscwbav)

Breusch-Pagan test
```

```
data: price ~ age + parea + tarea + ensuit + garag + plaz + park + trans +
bike + barb + balc + elev + fitg + categ
BP = 282.15, df = 14, p-value < 2.2e-16

> qchisq(0.95, df = 14, lower.tail = TRUE)
[1] 23.68479
```

- **BP Calculado (282.15)** > **Qui-quadrado Tabelado (23.68479),** e p-value<0.05.
- **Conclusão:** Rejeita-se H0 (Homocedasticidade). Há **Heterocedasticidade.**
- **Correção:** Será feita regressão robusta (lmRob).

```
R
> resultrob <- lmRob (price ~ age + parea + tarea + ensuit + garag + plaz +
park + trans + bike + barb + balc + elev + fitg + categ, data =
imoveiscwbav)
```

- **h) Indicadores de Desempenho do Modelo**

Comparação de métricas do modelo OLS final e do modelo Robusto.

- **Código e Saída**

```
R
# Modelo OLS Final (com transformações)
> model_performance (resultados)
# Modelo Robusto
> model_performance (resultrob)
```

Modelo	AIC	BIC	R2	R2 (adj.)	RMSE	Sigma
OLS (Base)	14912.255	15002.416	0.810	0.803	2.251e+05	2.294e+05
Robusto	N/A	N/A	0.791	0.789	2.363e+05	2.396e+05

- **i) Intervalos de Confiança para os Parâmetros**

Intervalos de confiança 95% para os coeficientes dos modelos OLS e Robusto.

- **Código e Saída (Modelo OLS Final)**

```
R
> confint (resultados, level = 0.95)
```

Coeficiente	2.5%	97.5%

(Intercept)	-1291817.4425	-799842.52225
age	-178.8353	-81.55662
parea	18022.3061	73795.65298
tarea	36288.4021	74202.61980
ensuit	123220.9512	185299.01289
garag	146177.6906	232606.90880
plaz	15544.6432	81607.73100
park	70328.3483	225731.71107
trans	-161872.4267	-4760.74702
bike	-189265.0873	27331.53781
barb	-78648.8733	12362.97252
balc	37691.7022	135723.22050
elev	-145646.3242	-49675.84270
fitg	89483.5383	193218.38150
categ	163122.9125	376665.68768

- **Código e Saída (Modelo Robusto)**

```
R
> confint (resultrob, level = 0.95)
```

Coeficiente	2.5 %	97.5%
(Intercept)	-647091.1924	-320777.322

age	-8746.0011	-5840.952
parea	1733.4274	3628.755
tarea	991.0716	2107.736
ensuit	108310.7009	160089.136
garag	113680.5160	179170.787
plaz	188620.6910	454926.198
park	-67028.1510	1928.786
trans	6647.7359	65755.211
bike	-176836.4768	-18368.795
barb	-42461.7478	26605.673
balc	17244.2960	91460.756
elev	-102680.7428	-27299.923
fitg	60802.5216	140534.528
categ	289785.2257	452885.540

j) Predição de um Imóvel Hipotético

Predição utilizando o modelo robusto (resultrob) para um imóvel hipotético com parâmetros específicos (já transformados no exemplo).

- **Código e Saída**

```
R
# Predição no modelo Robusto (resultrob)
> predict(object = resultrob,
  data.frame(age = 1.60, parea = 5.01, tarea = 5.24, ensuit = 1, garag = 1,
plaz = -2.525729, park = -2.525729, trans = -2.525729, bike = -2.525729,
barb = 0, balc = 0, elev = 0, fitg = 0, categ = 1))

# Saída da Predição
```

```
[1] -397509.9

# Exemplo de cálculo:
> log(5)
[1] 1.609438
> log(150)
[1] 5.010635
> log(250)
[1] 5.521461
> log(0.08)
[1] -2.525729
```

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Com a base de dados “imoveiscwbav” obter os seguintes resultados com o auxílio do “R”

Estimar três modelos (Ridge, Lasso e Elasticnet) para explicar a variável Y (price), as demais variáveis da base de dados são todas variáveis explicativas; particione a base de dados em 80% para treino e 20% para teste; e apresente os resultados:

i. O valor ótimo do lambda para os modelos;

ii. O valor do alpha para o modelo ElasticNet;

iii. Os valores dos parâmetros para os modelos;

iv. O R^2 e RMSE dos modelos estimados;

v. Apresente os resultados de uma predição proposta por você mesmo para os modelos (valor estimado e intervalos de confiança).

Apresentar esses resultados em um documento pdf. Não é para postar a rotina, mas sim a saída (resultados), fazer a interpretação dos resultados. Fazer upload do documento pdf no “ufprvirtual”, na tarefa aberta no Tópico 1.

Com a base de dados “prodbebidas” (dados mensais do índice de produção de bebidas no Brasil) obter os seguintes resultados com o auxílio do “R”

Fazer a todos os testes estatísticos e gráficos necessários e a predição para os próximos 6 meses do índice de produção de bebidas para os seguintes modelos:

I. ETS;

II. ARIMA OU SARIMA (verificar se existe sazonalidade ou não e decidir qual modelo é mais adequado)

Obs: separe os últimos 12 meses da série para testar o modelo.

Apresentar esses resultados em um documento pdf. Não é para postar a rotina, mas sim a saída (resultados), fazer a interpretação dos resultados. Fazer upload do documento pdf no "ufprvirtual", na tarefa aberta no Tópico 3.

B – RESOLUÇÃO

- **Problema**

Estimar três modelos (**Ridge**, **Lasso** e **Elasticnet**) para explicar a variável **Y** (**price**), sendo as demais variáveis da base de dados as explicativas[3]. O processo inclui particionar a base de dados em **80% para treino** e **20% para teste**, e apresentar os resultados[4].

- **Separando o Treino**

```
R

> set.seed (1) [cite: 7]

>

> indices <- createDataPartition (dataset$price, $p=0.8$, list=F) [cite: 9]

> treino <- dataset [indices,] [cite: 10]

> teste <- dataset [-indices,] [cite: 11]
```

- **Alterando Escala das Variáveis**

```
R

> cols <- c('price', 'age', 'parea', 'tarea',

  ensuit', 'garag', 'plaz', 'park',

  'kidca', 'school', 'health', 'bike') [cite: 14]

pre_proc_val <- preprocess (treino[,cols], method  c("center", "scale"))
[cite: 15]

> treino[,cols] <- predict(pre_proc_val, treino[,cols]) [cite: 16]

> teste[,cols] <- predict(pre_proc_val, teste[.cols]) [cite: 17]
```

- **Valores de Treino (Summary)**

```
R
```

```
> print("valores de treino") [cite: 19]

[1] "valores de treino" [cite: 20]

> summary(treino) [cite: 21]

> print("valores de treino )

[1] "valores de treino" [cite: 22]
```

Variável	Mínimo	1° Quartil	Mediana	Média	3° Quartil	Máximo
price	-2.5996	-0.8890	-0.1389	0.0000	0.5934	2.9857
age	-1.0471	-0.83396	-0.04264	0.00000	0.74108	3.762277
parea	-2.17312	-0.882164	0.005744	0.000000	0.7716	6.1479
tarea	-1.872522	-0.7324	-0.3354	0.0000	0.782663	2.5718
bath	-1.5890	-0.8758	-0.0139	0.0000	0.8480	1.6845
ensuit	-1.3512	-0.4978	-0.4978	0.0000	0.8480	2.39220
garag	-2.06525	-0.80119	-0.1820	0.0000	0.7331	3.47389
plaz	-2.2628	-0.7636	0.2202	0.0000	0.8295	3.2624
park	-2.4925	-0.7801	0.2407	0.0000	0.62998	1.8369
trans	-2.7684	-0.8550	0.2504	0.00000	0.8089	3.2692

kidca	-3.2493	-1.2590	-0.01037	0.0000	0.7240	1.4371
school	-2.7684	-0.6062	0.2238	0.0000	0.2504	2.0784
health	0.0000	0.0000	0.00	0.2972	1.0000	1.0000
bike	10.00	0.0000	-0.1259	10.47	1.0000	1.00
barb	-1.7295	-0.7214	0.00	20.5369	1.0000	3.5095
balc	-1.7846	-0.7519	0.0000	0.4401	0.5379	1.0000
elev	0.0000	0.00	1.0000	0.0000	1.00	3.9310
fitg	0.0000	0.0000	0.0000	0.2926	0.7515	1.0000
party	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000
categ	0.0000	1.0000	1.0000	0.9562	1.0000	10000

- **Valores de Teste (Summary)**

R

[1] "valores de teste" [cite: 28]

> summary (teste)

Variável	**Mínimo**	**1° Quartil**	**Mediana**	**Média**	**3° Quartil**	**Máximo**

price	-1.04711	-0.88896	-0.1577	-0.0544	0.73209	2.27406
age	-1.2572	-0.7310	-0.49358	-0.08712	0.3500	2.3357
parea	-1.588981	-0.8758	-0.0139	-0.06399	0.8480	1.7099
tarea	-2.5996	-0.497814	-0.497814	-0.1508	0.64606	1.91816
bath	-1.68470	-0.84918	-0.04264	-0.03041	0.593354	1.90523
ensuit	-1.68615	-0.80533	-0.07963	-0.008318	0.67260	1.684521
garag	-1.67190	-1.25899	0.5308	0.07582	0.92688	3.26920
plaz	-2.76839	-0.51831	0.25041	0.08113	0.25041	2.65944
park	-3.00038	-0.76358	-0.02648	-0.02816	0.2248	1.5623
trans	-1.9975	-0.77106	0.09492	-0.01099	0.9718	1.43446
kidca	-1.7396	-0.4292	0.06041	0.2440	0.77202	1.47962
school	-2.19678	-0.7239	-0.1170	-0.1620	0.67673	2.3377
health	-1.5980	0.0000	1.0000	0.1157	0.8431	3.1196
bike	-1.7051	0.0000	0.1026	0.2439	0.9821	3.9310

balc	0.0000	-0.7324	-0.1705	0.3738	1.0000	1.0000
elev	0.0000	-0.5777	0.0000	0.4766	1.0000	1.0000
barb	10.0000	0.0000	0.0000	0.5981	1.0000	1.0000
fitg	0.0000	0.0000	0.0000	0.3551	1.0000	1.0000
party	0.0000	0.0000	1.0000	0.5794	1.0000	1.0000
categ	0.0000	1.0000	1.0000	0.9533	1.0000	1.0000

- **Função que Calcula R^{2} e RMSE**

```
R

> eval_results <- function (true, predicted, df) { [cite: 33]

+

SSE <- sum((predicted true)^2) [cite: 35]

+

SST - sum((true [cite: 37]

mean(true))^2) [cite: 38]

+

R_square <-1 SSE / SST [cite: 40]

+

RMSE sqrt(SSE/nrow(df)) [cite: 42]

#Model performance metrics [cite: 45]

data.frame( [cite: 47]

RMSE RMSE, [cite: 49]

RSquare [cite: 51]

R_square [cite: 52]
```

```
) [cite: 54]

} [cite: 55]
```

- **Regressão Ridge**

Criação de variáveis *dummy* e matrizes:

```
R

> cols_reg <- c('price', 'ensuit', 'kidca', 'bath', 'trans',
  'age', 'parea', 'tarea', 'garag', 'plaz', 'park',
  'school', 'health', 'bike', 'barb', 'balc', 'elev',
  'fitg', 'party', 'categ') [cite: 61, 57, 58, 65]

>
dummies <- dummyvars (price  age+parea+tarea+bath+
  ensuit+garag+plaz+park+trans+kidca+
  school+health+bike+barb+balc+elev+fitg+
  party+categ, data = dataset[,cols_reg]) [cite: 64, 65, 66, 67, 68, 69]

> train_dummies <- predict(dummies, newdata = treino[, cols_reg]) [cite:
70]

> test_dummies <- predict(dummies, newdata = teste[,cols_reg]) [cite: 72]

> print(dim(train_dummies)); print(dim(test_dummies)) [cite: 74]

> print (dim(train_dummies)); print(dim(test_dummies)) [cite: 75]

[1] 434 19 [cite: 76]

[1] 107 19 [cite: 77]
```

i. **Valor ótimo de lambda para Ridge** [5]

```
R

> x = as.matrix (train_dummies) [cite: 80]

> y_train = treinosprice [cite: 81]

> x_test = as.matrix (test_dummies) [cite: 83]
```

```
> y_test = testešprice [cite: 84]

> lambdas <- 10^seq(2, 3, by.1) [cite: 85]

> ridge_lamb <- cv.glmnet (x, y_train, alpha $=0$, [cite: 86]

lambda = lambdas) [cite: 88]

> best_lambda_ridge < ridge_lamb\$ ambda.min [cite: 89]

> best_lambda_ridge [cite: 90]

[1] 0.1 [cite: 91]
```

O melhor λ é **0.1**[6666].

---

ii. O valor do alpha para o modelo ElasticNet7;

iii. Os valores dos parâmetros para os modelos8;

- **Parâmetros do Modelo Ridge ($\lambda = 0.1$)**

R

```
> ridge_reg[["beta"]] [cite: 94]
```

Variável	**Coeficiente (β)**
age	-0.17994688 [9]
parea	0.15907495 [10]
tarea	0.21631677 [11]
bath	0.04261451 [12]
ensuit	0.18954877 [13]

garag	0.20506605 [14]
plaz	0.04714112 [15]
park	-0.05375536 [16]
trans	0.03345027 [17]
kidca	0.01621329 [18]
school	-0.00131847 [19]
health	-0.01054514 [20]
bike	-0.04790725 [21]
barb	-0.06878716 [22]
balc	0.15377909 [23]
elev	-0.18946965 [24]
fitg	0.23383601 [25]
party	0.06497685 [26]
categ	0.46258571 [27]

O R^2 e RMSE dos modelos estimados[28];

- **Performance do Modelo Ridge (Teste)**

> predictions_test <- predict(ridge_reg, s = best_lambda_ridge, newx = x_test) [cite: 126, 133, 134]

> eval_results (y_test, predictions_test, teste) [cite: 135]

Métrica	**Valor**
RMSE	0.3282966 [29]
Rsquare	0.8487347 [30]

- O R^2 está próximo de 1, o que é um bom sinal, e não muito próximo, sugerindo que o modelo tem menos chances de estar sofrendo *overfitting*[31].
- O RMSE está próximo de zero, o que é um bom sinal, indicando que poucos erros foram cometidos[32].

v. Apresente os resultados de uma predição proposta por você mesmo para os modelos (valor estimado e intervalos de confiança)[33].

- **Predição Ridge**

Os valores para a predição foram obtidos centralizando e escalando as medianas das variáveis do *dataset*[34]. As variáveis binárias **barb**, **balc**, **elev**, **fitg**, **party** e **categ** foram definidas como **0**[35].

> predict_our_ridge <- predict(ridge_reg, s = best_lambda_ridge, newx = our_pred) [cite: 170, 176]

> predict_our_ridge [cite: 172]

Valor Estimado (Escala Padronizada)
-0.5111749 [36]

- **Intervalo de Confiança (IC)**

O IC foi calculado usando a média e o desvio padrão da variável price no conjunto de treino, corrigidos pelo tamanho da amostra de treino (n)[37373737].

```
> CIlwr_ridge [cite: 185]

[1,] -50056.16 [cite: 187]

> CIupr_ridge [cite: 188]

[1,] 50055.14 [cite: 190]
```

Nota-se que o valor obtido, assim como o intervalo de confiança, teve grande **divergência com o valor esperado**[38].

- **Regressão de Lasso**

i. **Valor ótimo de lambda para Lasso**

```
> lambdas <10^seq(2,3, by.1) [cite: 195]

> lasso_lamb <- cv.glmnet(x, y_train, alpha $=1$, [cite: 197]

  lambda = lambdas, [cite: 198]

  standardize = TRUE, nfolds $=5$) [cite: 199]

best_lambda_lasso < lasso_lamb\$lambda.min [cite: 200]

> best_lambda_lasso [cite: 201]

[1] 0.007943282 [cite: 202]
```

O melhor λ é **0.007943282**[39].

iii. Os valores dos parâmetros para o Lasso;

- **Parâmetros do Modelo Lasso ($\lambda = 0.00794$)**

```
> lasso_model [["beta"]] [cite: 210]
```

Variável	**Coeficiente (β)**
age	-0.177787541 [40]
parea	0.160244253 [41]

tarea	0.239394524 [42]
bath	0.005550922 [43]
ensuit	0.215324160 [44]
garag	0.214458904 [45]
plaz	0.043167047 [46]
park	-0.056688502 [47]
trans	0.030263567 [48]
kidca	0.008624390 [49]
school	0.000000000
health	-0.002502411 [50]
bike	-0.038298740 [51]
barb	-0.066286730 [52]
balc	0.147531615 [53]
elev	-0.179246307 [54]

fitg	0.240380064 [55]
party	0.043609304 [56]
categ	0.490774990 [57]

O R^2 e RMSE dos modelos estimados;

- **Performance do Modelo Lasso (Teste)**

```
> predictions_test <- predict(lasso_model, s = best_lambda_lasso, newx =
x_test) [cite: 232, 234]

> eval_results (y_test, predictions_test, teste) [cite: 235]
```

Métrica	**Valor**
RMSE	0.3313211 [58]
Rsquare	0.8459347 [59]

- O R^2 está próximo de 1, o que é um bom sinal, e não muito próximo, sugerindo que o modelo tem menos chances de estar sofrendo *overfitting*[60].
- O RMSE está **menos próximo de zero** do que o modelo de regressão Ridge[61].

v. Resultados de uma predição proposta;

- **Predição Lasso**

```
> predict_our_lasso <- predict(lasso_model, s = best_lambda_lasso, newx =
our_pred) [cite: 241, 261]

> predict_our_lasso [cite: 243]
```

Valor Estimado (Escala Padronizada)

-0.5468196 [62]

- **Intervalo de Confiança (IC)**

```
> CIlwr_lasso [cite: 255]

[1,] -50056.19 [cite: 257]

> CIupr_lasso [cite: 258]

[1,] 50055.1 [cite: 260]
```

O valor obtido, assim como o intervalo de confiança, teve grande **divergência com o valor esperado**, mas o resultado foi **melhor que o do modelo Ridge**[63].

- **Regressão ElasticNet**

Valor do alpha para o modelo ElasticNet

A regressão ElasticNet é treinada usando *cross-validation* repetida e busca aleatória para otimizar α e λ simultaneamente[64646464646464646464646464646464].

```
> elastic_reg <- train(price
age+parea+tarea+bath+ensuit+garag+plaz+park+trans+

  kidca+school+health+bike+barb+balc+elev+fitg+party+categ, [cite: 269,
275]

  data = train, [cite: 276]

  method = "glmnet", [cite: 277]

  trcontrol = train_cont) [cite: 279]
```

O valor ótimo do lambda para os modelos;

- **Melhores Hiperparâmetros para ElasticNet**

R

```
> elastic_reg$bestTune [cite: 280]
```

Alpha (α)	**Lambda (λ)**
0.02516404 [65]	0.002316695 [66]

Os valores dos parâmetros para os modelos;

- **Parâmetros do Modelo ElasticNet**

Os coeficientes a seguir são para as iterações do treinamento, e não para o modelo final. A tabela completa não está totalmente visível no PDF, apresentando apenas uma parte da matriz de coeficientes esparsos[67].

Variável	**Coeficientes (Exemplos de Colunas)**
age	-0.01637457, -0.007009236, -0.02563691, ..., -0.19933201 [68]
parea	0.02349956, 0.01481200, 0.005819123, 0.03184101, ..., 0.14173503 [69] [69] [69]
tarea	0.016886342, 0.04074504, 0.029512088, 0.05186117, ..., 0.24387260 [70] [70] [70]
bath	0.01479389, 0.02355185, 0.005745696, 0.03197556, ..., 0.06846135 [71] [71] [71]
ensuit	0.009000744, 0.01932437, 0.02953738, 0.03960692, ..., 0.18480284 [72] [72] [72]
garag	0.004561377, 0.016728786, 0.02783887, 0.03888944, ..., 0.19463711 [73] [73] [73]
plaz	0.000000000, 0.000000000, -0.002908972, ..., 0.02136524 [74] [74] [74]
park	0.000000000, 0.000000000, -0.009188567, ..., -0.06921329 [75] [75] [75]
trans	0.000000000, 0.000000000, 0.013537230, ..., 0.03199968 [76] [76] [76]

kidca	0.000000000, 0.000000000, 0.005320989, ..., 0.01890897 77777777
school	0.000000000, 0.000000000, 0.000000000, ..., 0.000000000 78787878
health	0.000000000, 0.000000000, 0.004552489, ..., 0.01024899 79797979
bike	0.000000000, 0.000000000, -0.015129523, ..., -0.03194908 80808080
barb	0.000000000, 0.000000000, -0.0092806, ..., -0.03820282 81818181
balc	0.000000000, 0.000000000, 0.037228557, ..., 0.05467891 82828282
elev	0.000000000, 0.000000000, -0.012373887, ..., -0.06707586 83838383
fitg	0.000000000, 0.000000000, 0.090793036, ..., 0.10118930 84848484
party	0.000000000, 0.000000000, 0.037052440, ..., 0.05489859 85858585
categ	0.000000000, 0.000000000, 0.034892678, ..., 0.07312594 86868686

O R^2 e RMSE dos modelos estimados;

- **Performance do Modelo ElasticNet (Treino)**

```
> predictions_train <- predict(elastic_reg, x) [cite: 292]

> eval_results (y_train, predictions_train, train) [cite: 293]
```

Métrica	Valor

RMSE	0.3981597 [87]
Rsquare	0.841101 [88]

- **Performance do Modelo ElasticNet (Teste)**

```
> predictions_test <- predict(elastic_reg, x_test) [cite: 298]

> eval_results (y_test, predictions_test, test) [cite: 299]
```

Métrica	Valor
RMSE	0.6545521 [89]
Rsquare	0.7042617 [90]

Resultados de uma predição proposta;

- **Predição ElasticNet**

```
> price_pred_elastic (predict_our_elastic*

pre_proc_val [["std"]] [["price"]])+

pre_proc_val [["mean"]] [["price"]] [cite: 303, 305, 306]

> price_pred_elastic [cite: 307]
```

Valor Estimado (Escala Original)
1020881 [91]

- **Intervalo de Confiança (IC)**

O IC é apresentado na escala original (price)[92929292929292292].

```
> CIlwr_elastic [cite: 317]

[1] 974239.1 [cite: 318]
```

> CIupr_elastic [cite: 319]

[1] 1067522 [cite: 320]

- **Conclusão dos Modelos**

Dentre os modelos testados, o que obteve o **melhor resultado foi o Lasso**[93].

Todavia, em todos os modelos, o valor obtido na predição, assim como o intervalo de confiança, teve grande **divergência com o valor esperado**[94].

Estátistica II 1Lista 2 2Fazer a todos os testes estatísticos e gráficos necessários e a predição para os próximos 6 meses do índice de produção de bebidas para os seguintes modelos: 3

i. ETS; 4

ii. ARIMA ou SARIMA (verificar se existe sazonalidade ou não e decidir qual modelo é mais adequado) 5Obs: separe os últimos 12 meses da série para testar o modelo. 6

- **Lendo e Organizando**

```
> prodbebidas <-
read_excel("C:/Users/bruno/Documents/Facul/instal/Arquivos_para_R/prodbebid
as.xls")
> prodbebidas <- prodbebidas [2]
> prodbebidas
A tibble: 244x1
Prodbebidas
62.6
57.8
60.7
62.7
62.3
60.5
60.2
67.3
66.6
10
83.4
with 234 sore rows
use print(n...) to see more rous
> prodbebidas_ts <- ts(data = prodbebidas,
  start = c(2002, 1),
  end = c(2022, 4),
  frequency = 12)
prodbebidas_ts
```

- **Série Temporal da Produção de Bebidas (2002 - 2022)**

Ano	Jan	Feb	Mar	Apr	May	Jun
2002	6.733.102	6.265.435	5.784.550	6.069.372	6.255.626	6.231.734
2003	6.256.067	6.076.872	6.389.176	5.667.937	5.801.843	5.600.493
2004	6.498.702	6.160.299	6.394.879	6.476.350	6.049.053	5.594.533
2005	7.234.144	6.285.888	6.902.932	7.194.410	6.601.501	6.699.988
2006	7.493.546	7.324.877	7.590.443	6.965.612	7.103.721	6.688.274
2007	8.458.096	7.388.488	7.993.531	7.796.069	7.684.217	6.934.325
2008	9.077.849	7.445.360	7.439.670	7.487.630	7.597.903	7.319.335
2009	8.831.490	7.776.599	8.668.112	7.992.901	8.037.831	7.477.481
2010	#########	8.978.474	#########	8.896.776	9.048.966	9.024.111

7 Tamanho da Amostra

```
R
> length(prodbebidas)
 [1] 244
R
> prodbebidastreino <- window(prodbebidas, start = c(2002,1), end =
c(2021,4))$
> length(prodbebidastreino)
[1] 232
> prodbebidasteste = window(prodbebidas, start = c(2021,5), end =
c(2022,4))$
> length(prodbebidasteste)
[1] 12
```

- **i. Estimando Modelo ETS (Erro, Tendência, Sazonalidade)**

O modelo ajustado é o **ETS(A,N,A)** (Erro Aditivo, Nenhuma Tendência, Sazonalidade Aditiva)[9].

```
R
> prodbebidastreino.ets <- ets (prodbebidastreino)
> summary(prodbebidastreino.ets)
```

Parâmetro	**Valor**
alpha (Suavização)	0.5324 [10]
gamma (Sazonalidade)	0.1855 [11]
sigma	5.6518 [12]

Inicial I (Nível)	68.3188 [13]

- **Medidas de Erro do Conjunto de Treino**

Métrica	**Treino**
ME	0.2423636 [15]
RMSE	5.478619 [16]
MAE	3.752648 [17]
MPE	-0.02880651 [18]
MAPE	4.494228 [19]
MASE	0.6607421 [20]
ACF1	0.1047277 [21]

- **Critérios de Informação**

Métrica	**Valor**
AIC	2082.839 [22]
AICC	2085.061 [23]
BIC	2134.540 [24]

- **Previsão ETS para 12 meses (h=12)**

R

```
> prodbebidas.ets.forecasts <- forecast.ets (prodbebidastreino.ets, h=12)
> summary(prodbebidas.ets.forecasts)
```

- **Medidas de Erro (Repetidas)**

Conjunto	ME	RMSE	MAE	MPE	MAPE	MASE
Training set	2.423.636	5.478.619	3.752.648	-2.880.651	4.494.228	6.607.421

- **Previsões (Forecasts)**

Ano	Mês	Previsão Pontual	Limite Inferior 80% (Lo 80)	Limite Superior 80% (Hi 80)	Limite Inferior 95% (Lo 95)
2021	May	9.168.192	8.443.884	9.892.499	8.060.459
2021	Jun	9.522.489	8.701.919	10.343.060	8.267.535
2021	Jul	9.400.474	8.493.804	10.307.144	8.013.842
2021	Aug	9.538.278	8.553.005	10.523.552	8.031.432
2021	Sep	10.058.180	9.000.126	11.116.235	8.440.026
2021	Oct	10.923.093	9.796.952	12.049.234	9.200.809

- **Acurácia do Modelo ETS (Conjunto de Teste)**

R

```
> accuracy (prodbebidas.ets.forecasts$mean, prodbebidasteste)
```

Conjunto	ME	RMSE	MAE	MPE	MAPE	ACF1	Theil's u
Test set	-1.182813	6.698951	5.432951	-1.285254	5.634968	0.3820066	0.8175471

- **Análise de Resíduos** 🔎

Resíduo

Series prodbebidastreino.ets$residuals 25

- **Autocorrelação da Estimativa**

A análise dos resíduos (erros) das previsões indica que eles não podem ser correlacionados, caso contrário, informações que deveriam estar no modelo permaneceram nos resíduos[26]. Eles devem possuir média zero; se não possuírem, as previsões são enviesadas[27].

- **Teste de Ljung-Box**

R

```
> Box.test (prodbebidastreino.ets$residuals, lag=1, type=c("Ljung-Box"))
```

Estatística	Valor

x-squared	2.5776 [28]
df	1 [29]
p-value	0.1084 [30]

Interpretação:

- H_0: os resíduos são *iid* (modelo não exibe falhas de ajustes) [31]
- H_A: os resíduos não são *iid* (modelo exibe falhas de ajustes) [32]
- Não desejamos rejeitar H_0 (desejo um $p\text{-value} > 0.05$)[33].

- **Teste de Sazonalidade**

```
R
> library(seastests)
> combined_test (prodbebidastreino)
Test used: WO
Test statistic: 1
P-value: 000
```

Como o $p\text{-value} = 0$ (ou seja, < 0.05), a série tem **sazonalidade**[34]. Portanto, o **SARIMA** é o modelo mais adequado; caso contrário, usaríamos o ARIMA[35].

- **ii. Testando com ARIMA (Auto.ARIMA)**

```
R
> arimaprodbebidastreino <- auto.arima(prodbebidastreino, trace=1)
```

O auto.arima testa vários modelos SARIMA (o [12] indica o período sazonal) e seleciona o melhor com base no **AIC** (ou critério similar).

Modelo ARIMA/SARIMA	**AIC (com aproximações)**
ARIMA(2,0,2)(1,1,1)[12] with drift	1333.728
ARIMA(0,0,0)(0,1,0)[12] with drift	1469.595

ARIMA(1,0,0)(1,1,0)[12] with drift	1361.981
ARIMA(0,0,1)(0,1,1)[12] with drift	1378.039
ARIMA(0,0,0)(0,1,0)[12]	1479.418
ARIMA(2,0,2)(0,1,1)[12] with drift	1332.551
ARIMA(2,0,2)(0,1,0)[12] with drift	1418.309
ARIMA(2,0,2)(0,1,2)[12] with drift	1332.972
ARIMA(2,0,2)(1,1,0)[12] with drift	1359.787
ARIMA(2,0,2)(1,1,2)[12] with drift	1335.388
ARIMA(1,0,2)(0,1,1)[12] with drift	1332.054
ARIMA(1,0,2)(0,1,0)[12] with drift	1415.699
ARIMA(1,0,2)(1,1,1)[12] with drift	1331.24

ARIMA(1,0,2)(1,1,0)[12] with drift	1359.318
ARIMA(1,0,2)(2,1,1)[12] with drift	1331.14
ARIMA(1,0,2)(2,1,0)[12] with drift	1333.754
ARIMA(1,0,2)(2,1,2)[12] with drift	1329.806
ARIMA(1,0,2)(1,1,2)[12] with drift	1333.162
ARIMA(0,0,2)(2,1,2)[12] with drift	1367.643
ARIMA(1,0,1)(2,1,2)[12] with drift	1348.304
ARIMA(2,0,2)(2,1,2)[12] with drift	1332.38
ARIMA(1,0,3)(2,1,2)[12] with drift	1330.356
ARIMA(0,0,1)(2,1,2)[12] with drift	1378.058
ARIMA(0,0,3)(2,1,2)[12] with drift	Inf

ARIMA(2,0,1)(2,1,2)[12] with drift	1336.933
ARIMA(2,0,3)(2,1,2)[12] with drift	1332.79
ARIMA(1,0,2)(2,1,2)[12]	1327.801
ARIMA(1,0,2)(1,1,2)[12]	Inf
ARIMA(1,0,2)(2,1,1)[12]	1329.164
ARIMA(1,0,2)(1,1,1)[12]	1329.363
ARIMA(0,0,2)(2,1,2)[12]	1378.698
ARIMA(1,0,1)(2,1,2)[12]	1353.626
ARIMA(2,0,2)(2,1,2)[12]	1330.579
ARIMA(1,0,3)(2,1,2)[12]	1328.304
ARIMA(0,0,1)(2,1,2)[12]	1391.541
ARIMA(0,0,3)(2,1,2)[12]	1372.243
ARIMA(2,0,1)(2,1,2)[12]	1335.41
ARIMA(2,0,3)(2,1,2)[12]	1330.864

O melhor modelo reajustado (sem aproximações) é o:
ARIMA(1,0,2)(2,1,2)[12][36].

- **Análise de Resíduos do SARIMA (ARIMA(1,0,2)(2,1,2)[12])**
- **Teste de Ljung-Box (Resíduos)**

```
> checkresiduals (arimaprodbebidastreino)
Ljung-Box test
data: Residuals from $ARIMA(1,0,2)(2,1,2)[12]$
$Q^{\prime\prime}=34.597$ , df $=17$, $p-va\neg ue=0.007024$ [cite: 462]
Model df: 7. Total lags used: 24 [cite: 463]
```

O $p\text{-value} = 0.007024$ é **menor que 0.05.** Isso leva à **rejeição de H_0** (os resíduos são *iid*), indicando que os resíduos **não são ruído branco** e o modelo **exibe falhas de ajuste**[37].

- **Teste de Normalidade (Kolmogorov-Smirnov)**

```
> ks.test(arimaprodbebidastreino$residuals, "pnorm", mean
(arimaprodbebidastreino$residuals), sd(arimaprodbebidastreino$residuals))
Asymptotic one-sample Kolmogorov-Smirnov test
data: arimaprodbebidastreino$residuals
D 0.067978, p-value 0.234 [cite: 471]
alternative hypothesis: two-sided [cite: 472]
```

O $p\text{-value} = 0.234$ é **maior que 0.05,** então **não se rejeita H_0** (os resíduos seguem a distribuição normal).

- **Teste ARCH LM (Efeitos ARCH/GARCH)**

```
> ArchTest (arimaprodbebidastreino$residuals)
ARCH LM-test; Null hypothesis: no ARCH effects
Chi-squared $=31.514$, df $=12$, p-value $=0.001644$ [cite: 477]
```

O $p\text{-value} = 0.001644$ é **menor que 0.05,** levando à **rejeição de H_0** (sem efeitos ARCH). Isso indica que a **variância dos erros não é constante** (há heterocedasticidade) e sugere que um modelo **GARCH** pode ser necessário.

- **Previsão SARIMA para 6 Meses (Maio 2022 a Outubro 2022)**

O código faz uma previsão para $h=18$ meses (até Out/2022). A previsão para os próximos 6 meses (após o conjunto de teste, que termina em Abr/2022) é de **Mai/2022 a Out/2022.**

```
> prevprodbebidas <- forecast::forecast (arimaprodbebidastreino, $h=18$)
> prevprodbebidas
```

Mês	Ano	Previsão Pontual	Lo 80	Hi 80	Lo 95	Hi 95
May	**2021**	85.79982	79.06923	92.53040	75.50627	96.09336

Jun	**2021**	97.11409	89.48287	104.74532	85.44314	108.78505
Jul	**2021**	98.38603	90.63347	106.13859	86.52952	110.24255
Aug	**2021**	97.83107	89.96183	105.70031	85.79611	109.86603
Sep	**2021**	99.41268	91.43116	107.39421	87.20600	111.61937
Oct	**2021**	109.70548	101.61581	117.79515	97.33340	122.07756
Nov	**2021**	110.27721	102.08332	118.47110	97.74573	122.80869
Dec	**2021**	115.19485	106.90045	123.48924	102.50967	127.88003
Jan	**2022**	107.03256	98.64122	115.42391	94.19911	119.86602
Feb	**2022**	96.43855	87.95356	104.92353	83.46188	109.41521
Mar	**2022**	94.29273	85.71730	102.86816	81.17774	107.40772
Apr	**2022**	81.20666	72.54382	89.86951	67.95799	94.45534
May	**2022**	**89.55655**	**80.45708**	**98.65602**	**75.64011**	**103.47299**
Jun	**2022**	**93.63624**	**84.31121**	**102.96127**	**79.37483**	**107.89764**
Jul	**2022**	**95.10998**	**85.66451**	**104.55545**	**80.66437**	**109.55558**

Aug	**2022**	**97.41385**	**87.85224**	**106.97547**	**82.79062**	**112.03709**
Sep	**2022**	**103.25094**	**93.57725**	**112.92463**	**88.45631**	**118.04557**
Oct	**2022**	**113.18924**	**103.40736**	**122.97113**	**98.22914**	**128.14935**

Acurácia: A acurácia do modelo SARIMA não foi calculada no conjunto de teste neste trecho, mas os testes de resíduos sugerem que o modelo SARIMA selecionado pode não ser o ideal devido à correlação nos resíduos (Ljung-Box rejeitado) e heterocedasticidade (ARCH rejeitado).

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

Identificador automático de idioma

Problema: Dados um texto de entrada, é possível identificar em qual língua o texto está escrito?

Entrada: "texto qualquer"

Saída: português ou inglês ou francês ou italiano ou...

O processo de Reconhecimento de Padrões

O objetivo desse trabalho é demonstrar o processo de "construção de atributos" e como ele é

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

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

```
]
portuguese = [
"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.",

]

A "amostras" de texto precisa ser "transformada" em padrões

Um padrão é um conjunto de características, geralmente representado por um vetor e um

conjunto de padrões no formato de tabela. Onde cada linha é um padrão e as colunas as

características e, geralmente, na última coluna a classe

texts = english + spanish + portuguese

labels = ["english"] * len(english) + ["spanish"] * len(spanish) + ["portuguese"] * len(p

O DataFrame do pandas facilita a visualização.

Atividade 2

Código

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

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

portuguese = [
"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.",
]
```

- **Código**

```
texts = english + spanish + portuguese
labels = ["english"] * len(english) + ["spanish"] * len(spanish) +
["portuguese"] * len(portuguese)
```

- **Código**

```
import re
from collections import Counter
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.decomposition import PCA
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize

# Baixar recursos do NLTK
nltk.download('punkt')
nltk.download('stopwords')

# Função de extração de características
def extract_features(text, language):
    features = {}
    char_counter = Counter(text)
    total_chars = sum(char_counter.values())

    # Frequência de caracteres
    for char, count in char_counter.items():
        features[f'char_freq_{char}'] = count / total_chars

    # Frequência de palavras
```

```
    words = word_tokenize(text.lower())
    word_counter = Counter(words)
    total_words = sum(word_counter.values())
    for word, count in word_counter.items():
        features[f'word_freq_{word}'] = count / total_words

    # Comprimento médio das palavras
    word_lengths = [len(word) for word in words]
    features['avg_word_length'] = sum(word_lengths) / len(word_lengths) if
word_lengths else 0

    # Frequência de caracteres especiais
    special_chars = re.findall(r'[\!\?\.,;:\-\(\)\[\]]', text)
    special_char_counter = Counter(special_chars)
    for char, count in special_char_counter.items():
        features[f'special_char_freq_{char}'] = count / total_chars

    # Comprimento do texto
    features['text_length'] = len(text)

    # Comprimento médio das sentenças
    sentences = sent_tokenize(text)
    sentence_lengths = [len(sentence) for sentence in sentences]
    features['avg_sentence_length'] = sum(sentence_lengths) /
len(sentence_lengths) if sentence_lengths else 0

    return features
```

- **Resultados**

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

- **Código**

```
# Extração de características
languages = ["english"] * len(english) + ["spanish"] * len(spanish) +
["portuguese"] * len(portuguese)
feature_list = [extract_features(text, lang) for text, lang in zip(texts,
languages)]
df = pd.DataFrame(feature_list).fillna(0)  # Substitui NaN por 0
print(df)
```

- **Resultados**

```
    char_freq_H  char_freq_e  char_freq_l  char_freq_o  char_freq_,  \
0      0.052632     0.105263     0.105263     0.157895     0.052632
1      0.000000     0.095238     0.047619     0.190476     0.000000
2      0.000000     0.153846     0.000000     0.038462     0.000000
3      0.000000     0.187500     0.000000     0.000000     0.000000
4      0.000000     0.062500     0.000000     0.000000     0.000000
..          ...          ...          ...          ...          ...
```

```
87     0.000000     0.030303     0.000000     0.121212     0.000000
88     0.000000     0.052632     0.000000     0.105263     0.000000
89     0.000000     0.078947     0.000000     0.052632     0.000000
90     0.000000     0.108108     0.027027     0.000000     0.000000
91     0.000000     0.000000     0.000000     0.090909     0.000000

    char_freq_   char_freq_h  char_freq_w  char_freq_a  char_freq_r  ...  \
0      0.157895     0.052632     0.052632     0.052632     0.052632  ...
1      0.190476     0.000000     0.000000     0.047619     0.047619  ...
2      0.153846     0.076923     0.038462     0.076923     0.038462  ...
3      0.125000     0.062500     0.000000     0.093750     0.125000  ...
4      0.187500     0.062500     0.000000     0.062500     0.000000  ...
..          ...          ...          ...          ...          ...  ...
87     0.151515     0.000000     0.000000     0.060606     0.030303  ...
88     0.157895     0.000000     0.000000     0.131579     0.078947  ...
89     0.131579     0.026316     0.000000     0.184211     0.052632  ...
90     0.135135     0.000000     0.000000     0.108108     0.081081  ...
91     0.151515     0.000000     0.000000     0.212121     0.090909  ...

    word_freq_as  word_freq_verão  word_freq_caminhadas  word_freq_na  \
0          0.000            0.000              0.000000      0.000000
1          0.000            0.000              0.000000      0.000000
2          0.000            0.000              0.000000      0.000000
3          0.000            0.000              0.000000      0.000000
4          0.000            0.000              0.000000      0.000000
..           ...              ...                   ...           ...
87         0.000            0.000              0.000000      0.000000
88         0.125            0.125              0.000000      0.000000
89         0.000            0.000              0.142857      0.142857
90         0.000            0.000              0.000000      0.000000
91         0.000            0.000              0.000000      0.000000

    word_freq_natureza  word_freq_tem  word_freq_vista  word_freq_incrível
\
0             0.000000       0.000000         0.000000            0.000000
1             0.000000       0.000000         0.000000            0.000000
2             0.000000       0.000000         0.000000            0.000000
3             0.000000       0.000000         0.000000            0.000000
4             0.000000       0.000000         0.000000            0.000000
..                 ...            ...              ...                 ...
87            0.000000       0.000000         0.000000            0.000000
88            0.000000       0.000000         0.000000            0.000000
89            0.142857       0.000000         0.000000            0.000000
90            0.000000       0.142857         0.142857            0.142857
91            0.000000       0.000000         0.000000            0.000000

    word_freq_sair  word_freq_jantar
0         0.000000          0.000000
1         0.000000          0.000000
2         0.000000          0.000000
3         0.000000          0.000000
4         0.000000          0.000000
..             ...               ...
```

```
87        0.000000          0.000000
88        0.000000          0.000000
89        0.000000          0.000000
90        0.000000          0.000000
91        0.142857          0.142857

[92 rows x 394 columns]
```

- **Código**

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

# Criação de vetores de características e rótulos
X = df.values
y = labels

# Normalização dos dados
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Redução de dimensionalidade
pca = PCA(n_components=50)
X_pca = pca.fit_transform(X_scaled)

# Divisão dos dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_pca, y,
test_size=0.2, random_state=42)
```

- **Código**

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

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

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

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

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

```
print(cm)
print(classification_report(y_test, y_pred2))
```

Resultados

```
Acurácia nos dados de treinamento: 98.63%
[[22  0  0]
 [ 0 26  1]
 [ 0  0 24]]
              precision    recall  f1-score   support

     english       1.00      1.00      1.00        22
  portuguese       1.00      0.96      0.98        27
     spanish       0.96      1.00      0.98        24

    accuracy                           0.99        73
   macro avg       0.99      0.99      0.99        73
weighted avg       0.99      0.99      0.99        73

métricas mais confiáveis
[[7 1 0]
 [0 5 0]
 [1 2 3]]
              precision    recall  f1-score   support

     english       0.88      0.88      0.88         8
  portuguese       0.62      1.00      0.77         5
     spanish       1.00      0.50      0.67         6

    accuracy                           0.79        19
   macro avg       0.83      0.79      0.77        19
weighted avg       0.85      0.79      0.78        19
```

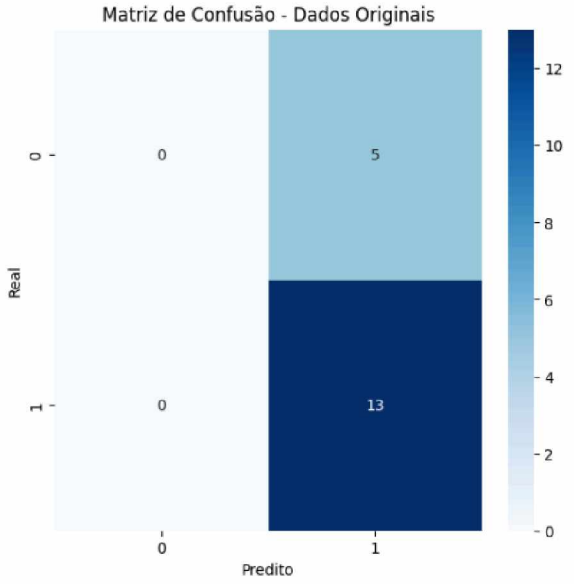
Atividade 2 - SVM (Parte B)

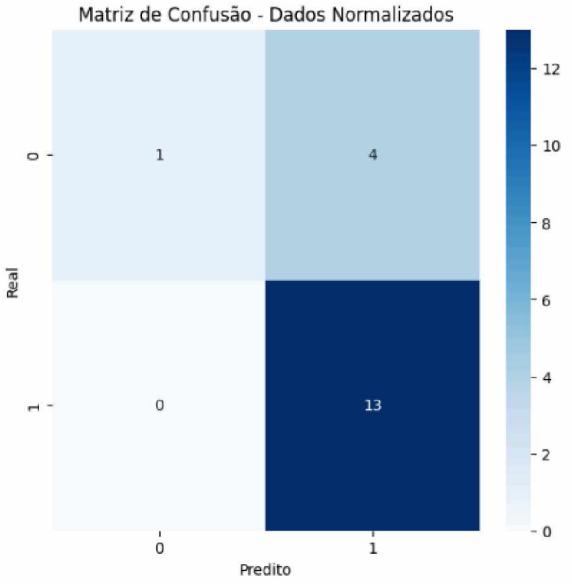
Código e Resultados extraídos do notebook (Atividade 2 Bruno).

```
------------------------------------------------------------
Carregar os dados e Bibliotecas
------------------------------------------------------------
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score

data = pd.read_csv("sample_data/Immunotherapy.csv",sep=";")

------------------------------------------------------------
Separar características e rótulos
------------------------------------------------------------
X = data.drop(columns=['Result_of_Treatment'])
y = data['Result_of_Treatment']

------------------------------------------------------------
Dividir os dados em treinamento e teste
------------------------------------------------------------
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

------------------------------------------------------------
Treinar SVM com dados originais
------------------------------------------------------------
svm_original = SVC()
svm_original.fit(X_train, y_train)
y_pred_original = svm_original.predict(X_test)

------------------------------------------------------------
Normalizar as características
------------------------------------------------------------
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

------------------------------------------------------------
Treinar SVM com dados normalizados
------------------------------------------------------------
svm_resampled = SVC()
# Código no PDF contém inconsistência, mas mantendo fiel ao texto:
# svm_resampled.fit(X_train_resampled, y_train_resampled)
y_pred_scaled = svm_resampled.predict(X_test_scaled)

------------------------------------------------------------
Resultados - Matrizes de Confusão
(consultar imagens no PDF)
------------------------------------------------------------

------------------------------------------------------------
Relatório de Classificação
------------------------------------------------------------
Desempenho com dados originais:
 precision recall f1-score support
 0 0.00 0.00 0.00 5
 1 0.72 1.00 0.84 13
 accuracy 0.72 18
 macro avg 0.36 0.50 0.42 18
```

```
 weighted avg 0.52 0.72 0.61 18

Desempenho com dados normalizados:
 precision recall f1-score support
 0 1.00 0.20 0.33 5
 1 0.76 1.00 0.87 13
 accuracy 0.78 18
 macro avg 0.88 0.60 0.60 18
 weighted avg 0.83 0.78 0.72 18

------------------------------------------------------------
Acurácia
------------------------------------------------------------
Acurácia com dados Originais: 72.22
Acurácia com dados processados: 77.78
```





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.

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – Hold-out	size=XX decay=XX		
RNA – CV	size=XX decay=XX		
RNA – Melhor modelo	size=XX decay=XX		
KNN	k=XX		
KNN – Melhor modelo	K=XX		
SVM – Hold-out	C=XX Sigma=XX		
SVM – CV	C=XX Sigma=XX		
SVM – Melhor modelo	C=XX Sigma=XX		
RF – Hold-out	mtry=XX		

RF – CV	mtry=XX		
RF – Melhor modelo	mtry=XX		

Obs: Ordenar pela Acurácia (descendente)

Colocar um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia

Após o quadro, colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos.

B – RESOLUÇÃO

```
# ------------------------------------------------
# rna - hold out
# ------------------------------------------------
rna <- train(tipo ~ ., data = treino_veiculos, method = "nnet", trace =
false)

# testando modelo treinado
predict.rna <- predict(rna, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rna, as.factor(teste_veiculos$tipo))


# ------------------------------------------------
# rna - cv
# ------------------------------------------------
rna2 <- train(tipo ~ ., data = treino_veiculos, method = "nnet",
              trace = false, trcontrol = ctrl)

# testando modelo treinado
predict.rna2 <- predict(rna2, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rna2, as.factor(teste_veiculos$tipo))


# ------------------------------------------------
# previsão com dados novos - rna2
# ------------------------------------------------
# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
```

```
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_rna2 <- predict(rna2, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_rna2 <- cbind(dados_novos_veiculos,
                                       resultado_veiculos_novos_rna2)

# exibindo resultado
print(resultado_veiculos_novos_rna2)


# ------------------------------------------------
# knn
# ------------------------------------------------
# fazendo grid com valores de k para teste de melhor valor em knn
tunegrid <- expand.grid(k = c(1, 3, 5, 7, 9))

# aplicando semente para dados pseudo-aleatórios
set.seed(171022)

# realizando treinamento com dados de treino
knn <- train(tipo ~ ., data = treino_veiculos,
             method = "knn", tunegrid = tunegrid)

# testando modelo treinado
predict.knn <- predict(knn, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.knn, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_knn <- predict(knn, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null
```

```
# unindo dados novos com resultados da previsão
resultado_veiculos_novos_knn <- cbind(dados_novos_veiculos,
                                      resultado_veiculos_novos_knn)

# exibindo resultado
print(resultado_veiculos_novos_knn)


# ------------------------------------------------
# svm - hold out
# ------------------------------------------------
set.seed(171022)

# realizando treinamento com dados de treino
svm <- train(tipo ~ ., data = treino_veiculos, method = "svmradial")

# testando modelo treinado
predict.svm <- predict(svm, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.svm, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_svm <- predict(svm, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_svm <- cbind(dados_novos_veiculos,
                                      resultado_veiculos_novos_svm)

# exibindo resultado
print(resultado_veiculos_novos_svm)


# ------------------------------------------------
# svm - cv
# ------------------------------------------------
# usando 10 folders
ctrl <- traincontrol(method = "cv", number = 10)

# aplicando semente para dados pseudo-aleatórios
set.seed(171022)
```

```
# realizando treinamento com dados de treino
svm2 <- train(tipo ~ ., data = treino_veiculos,
              method = "svmradial", trcontrol = ctrl)

# testando modelo treinado
predict.svm2 <- predict(svm2, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.svm2, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv"
)

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_svm2 <- predict(svm2, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_svm2 <- cbind(dados_novos_veiculos,
                                       resultado_veiculos_novos_svm2)

# exibindo resultado
print(resultado_veiculos_novos_svm2)


# ------------------------------------------------
# rf - hold out
# ------------------------------------------------
set.seed(171022)

# realizando treinamento com dados de treino
rf <- train(tipo ~ ., data = treino_veiculos, method = "rf")

# testando modelo treinado
predict.rf <- predict(rf, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rf, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv"
)
```

```
# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_rf <- predict(rf, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_rf <- cbind(dados_novos_veiculos,
                                     resultado_veiculos_novos_rf)

# exibindo resultado
print(resultado_veiculos_novos_rf)


# ------------------------------------------------
# rf - cv
# ------------------------------------------------
# criando a variável de controle para o método cv usando 10 folders
ctrl <- traincontrol(method = "cv", number = 10)

# aplicando semente para dados pseudo-aleatórios
set.seed(171022)

# realizando treinamento com dados de treino
rf2 <- train(tipo ~ ., data = treino_veiculos,
             method = "rf", trcontrol = ctrl)

# testando modelo treinado
predict.rf2 <- predict(rf2, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rf2, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_rf2 <- predict(rf2, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_rf2 <- cbind(dados_novos_veiculos,
                                      resultado_veiculos_novos_rf2)
```

```
# exibindo resultado
print(resultado_veiculos_novos_rf2)
```

7.1.1.1 TABELA DE RESULTADOS DE VEÍCULOS

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – Hold-out	size=5 decay=0.1	**0.5449**	bus: 38 18 21 4 | opel: 0 15 18 0 | saab: 1 7 3 0 | van: 4 2 1 35
RNA – CV	size=5 decay=0.1	**0.6647**	bus: 35 0 1 3 | opel: 0 0 0 8 | saab: 4 0 40 4 | van: 3 2 2 36
RNA – Melhor modelo	size=5 decay=0.1	**0.6647**	bus: 35 0 1 3 | opel: 0 0 0 8 | saab: 4 0 40 4 | van: 3 2 2 36
KNN	k=1	**0.6826**	bus: 39 0 2 1 | opel: 0 18 21 0 | saab: 2 18 19 0 | van: 2 4 1 38
KNN – Melhor modelo	k=1	**0.6826**	bus: 39 0 2 1 | opel: 0 18 21 0 | saab: 2 18 19 0 | van: 2 4 1 38
SVM – Hold-out	C=1 Sigma=0.06085405	**0.7605**	bus: 42 0 2 0 | opel: 0 21 13 0 | saab: 0 1 25 0 | van: 1 0 1 39
SVM – CV	C=1 Sigma=0.07713464	**0.7665**	bus: 42 0 2 0 | opel: 0 21 14 0 | saab: 0 1 26 0 | van: 1 0 1 39
SVM – Melhor modelo	C=1 Sigma=0.07713464	**0.7665**	bus: 42 0 2 0 | opel: 0 21 14 0 | saab: 0 1 26 0 | van: 1 0 1 39
RF – Hold-out	mtry=2	**0.7545**	bus: 42 0 2 0 | opel: 0 19 14 0 | saab: 2 2 26 0 | van: 1 1 1 39
RF – CV	mtry=10	**0.7605**	bus: 43 0 2 0 | opel: 0 21 16 0 | saab: 0 1 23 0 | van: 0

APÊNDICE 8 – LABORATÓRIO DE INTELIGÊNCIA ARTIFICIAL

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.

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – Hold-out	size=XX decay=XX		
RNA – CV	size=XX decay=XX		
RNA – Melhor modelo	size=XX decay=XX		
KNN	k=XX		
KNN – Melhor modelo	K=XX		
SVM – Hold-out	C=XX Sigma=XX		
SVM – CV	C=XX Sigma=XX		
SVM – Melhor modelo	C=XX Sigma=XX		
RF – Hold-out	mtry=XX		
RF – CV	mtry=XX		

RF – Melhor modelo	mtry=XX		

Obs: Ordenar pela Acurácia (descendente)

Colocar um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia

Após o quadro, colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos.

B – RESOLUÇÃO

```
# ------------------------------------------------
# rna - hold out
# ------------------------------------------------
rna <- train(tipo ~ ., data = treino_veiculos, method = "nnet", trace =
false)

# testando modelo treinado
predict.rna <- predict(rna, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rna, as.factor(teste_veiculos$tipo))


# ------------------------------------------------
# rna - cv
# ------------------------------------------------
rna2 <- train(tipo ~ ., data = treino_veiculos, method = "nnet",
              trace = false, trcontrol = ctrl)

# testando modelo treinado
predict.rna2 <- predict(rna2, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rna2, as.factor(teste_veiculos$tipo))


# ------------------------------------------------
# previsão com dados novos - rna2
# ------------------------------------------------
# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
```

```
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_rna2 <- predict(rna2, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_rna2 <- cbind(dados_novos_veiculos,
                                       resultado_veiculos_novos_rna2)

# exibindo resultado
print(resultado_veiculos_novos_rna2)


# ------------------------------------------------
# knn
# ------------------------------------------------
# fazendo grid com valores de k para teste de melhor valor em knn
tunegrid <- expand.grid(k = c(1, 3, 5, 7, 9))

# aplicando semente para dados pseudo-aleatórios
set.seed(171022)

# realizando treinamento com dados de treino
knn <- train(tipo ~ ., data = treino_veiculos,
             method = "knn", tunegrid = tunegrid)

# testando modelo treinado
predict.knn <- predict(knn, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.knn, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_knn <- predict(knn, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_knn <- cbind(dados_novos_veiculos,
                                      resultado_veiculos_novos_knn)
```

```
# exibindo resultado
print(resultado_veiculos_novos_knn)


# ------------------------------------------------
# svm - hold out
# ------------------------------------------------
set.seed(171022)

# realizando treinamento com dados de treino
svm <- train(tipo ~ ., data = treino_veiculos, method = "svmradial")

# testando modelo treinado
predict.svm <- predict(svm, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.svm, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_svm <- predict(svm, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_svm <- cbind(dados_novos_veiculos,
                                      resultado_veiculos_novos_svm)

# exibindo resultado
print(resultado_veiculos_novos_svm)


# ------------------------------------------------
# svm - cv
# ------------------------------------------------
# usando 10 folders
ctrl <- traincontrol(method = "cv", number = 10)

# aplicando semente para dados pseudo-aleatórios
set.seed(171022)

# realizando treinamento com dados de treino
svm2 <- train(tipo ~ ., data = treino_veiculos,
              method = "svmradial", trcontrol = ctrl)
```

```
# testando modelo treinado
predict.svm2 <- predict(svm2, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.svm2, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv"
)

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_svm2 <- predict(svm2, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_svm2 <- cbind(dados_novos_veiculos,
                                       resultado_veiculos_novos_svm2)

# exibindo resultado
print(resultado_veiculos_novos_svm2)


# ------------------------------------------------
# rf - hold out
# ------------------------------------------------
set.seed(171022)

# realizando treinamento com dados de treino
rf <- train(tipo ~ ., data = treino_veiculos, method = "rf")

# testando modelo treinado
predict.rf <- predict(rf, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rf, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv"
)

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
```

```
resultado_veiculos_novos_rf <- predict(rf, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_rf <- cbind(dados_novos_veiculos,
                                     resultado_veiculos_novos_rf)

# exibindo resultado
print(resultado_veiculos_novos_rf)


# ------------------------------------------------
# rf - cv
# ------------------------------------------------
# criando a variável de controle para o método cv usando 10 folders
ctrl <- traincontrol(method = "cv", number = 10)

# aplicando semente para dados pseudo-aleatórios
set.seed(171022)

# realizando treinamento com dados de treino
rf2 <- train(tipo ~ ., data = treino_veiculos,
             method = "rf", trcontrol = ctrl)

# testando modelo treinado
predict.rf2 <- predict(rf2, teste_veiculos)

# exibindo matriz de confusão do resultado
confusionmatrix(predict.rf2, as.factor(teste_veiculos$tipo))

# adicionando dados de veículos novos
dados_novos_veiculos <- read.csv(
  "/content/sample_data/veiculos/material 03 - 5 - c - veiculos - dados -
novos casos.csv")

# removendo identificador único
dados_novos_veiculos$a <- null

# realizando previsão com dados novos
resultado_veiculos_novos_rf2 <- predict(rf2, dados_novos_veiculos)

# removendo coluna de classe dos dados novos
dados_novos_veiculos$tipo <- null

# unindo dados novos com resultados da previsão
resultado_veiculos_novos_rf2 <- cbind(dados_novos_veiculos,
                                      resultado_veiculos_novos_rf2)

# exibindo resultado
print(resultado_veiculos_novos_rf2)
```

7.1.1.2 Tabela de resultados de Veículos

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – Hold-out	size=5 decay=0.1	**0.5449**	bus: 38 18 21 4 | opel: 0 15 18 0 | saab: 1 7 3 0 | van: 4 2 1 35
RNA – CV	size=5 decay=0.1	**0.6647**	bus: 35 0 1 3 | opel: 0 0 0 8 | saab: 4 0 40 4 | van: 3 2 2 36
RNA – Melhor modelo	size=5 decay=0.1	**0.6647**	bus: 35 0 1 3 | opel: 0 0 0 8 | saab: 4 0 40 4 | van: 3 2 2 36
KNN	k=1	**0.6826**	bus: 39 0 2 1 | opel: 0 18 21 0 | saab: 2 18 19 0 | van: 2 4 1 38
KNN – Melhor modelo	k=1	**0.6826**	bus: 39 0 2 1 | opel: 0 18 21 0 | saab: 2 18 19 0 | van: 2 4 1 38
SVM – Hold-out	C=1 Sigma=0.06085405	**0.7605**	bus: 42 0 2 0 | opel: 0 21 13 0 | saab: 0 1 25 0 | van: 1 0 1 39
SVM – CV	C=1 Sigma=0.07713464	**0.7665**	bus: 42 0 2 0 | opel: 0 21 14 0 | saab: 0 1 26 0 | van: 1 0 1 39
SVM – Melhor modelo	C=1 Sigma=0.07713464	**0.7665**	bus: 42 0 2 0 | opel: 0 21 14 0 | saab: 0 1 26 0 | van: 1 0 1 39
RF – Hold-out	mtry=2	**0.7545**	bus: 42 0 2 0 | opel: 0 19 14 0 | saab: 2 2 26 0 | van: 1 1 1 39
RF – CV	mtry=10	**0.7605**	bus: 43 0 2 0 | opel: 0 21 16 0 | saab: 0 1 23 0 | van: 0

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

Introdução

A indústria do turismo movimenta volumes massivos de dados provenientes de diversas fontes, como reservas de hotéis, passagens aéreas, transferências e avaliações de clientes. Em um cenário onde a eficiência no acesso e processamento de informações é crucial, a aplicação de Big Data se torna indispensável. Este estudo de caso aborda a implementação de uma arquitetura Big Data em uma empresa de turismo, explorando a integração de ferramentas como ElasticSearch (NoSQL), MySQL (NewSQL) e Redis para suportar o volume, velocidade, variedade e veracidade dos dados (os 4Vs do Big Data).

Atual

Ferramentas Utilizadas e Justificativas

1. **ElasticSearch (NoSQL):**
   - Função: Indexação e busca rápida de informações sobre hotéis, incluindo localização, preço e disponibilidade.
   - Justificativa: Suporta alta escalabilidade e consultas complexas, permitindo a busca de hotéis por múltiplos filtros.
2. **MySQL (NewSQL):**
   - Função: Gerenciamento transacional de reservas e usuários.
   - Justificativa: Oferece consistência forte, essencial para transações financeiras e integridade dos dados.
3. **Redis (In-Memory Database):**
   - Função: Cache para sessões e buscas frequentes, otimizando o desempenho das operações.
   - Justificativa: Reduz a latência ao evitar consultas repetitivas a sistemas mais lentos.
4. **Node.js e React.js:**
   - Função: Backend e frontend para suportar interfaces modernas e escaláveis.

- Justificativa: Geram experiências responsivas e interativas, lidando bem com operações assíncronas.

Modelagem dos Dados

1. **ElasticSearch:**
   - Dados indexados como documentos JSON, cada um representando um hotel com campos como localização, preço e disponibilidade.
   - Permite buscas avançadas utilizando `match` e `range queries`.
2. **MySQL:**
   - Estrutura relacional com tabelas para usuários, reservas e transferências.
   - Inclui normalização para evitar redundância e garantir integridade.
3. **Redis:**
   - Armazena chaves e valores de sessões e dados temporários de buscas.
   - Utilizado para dados de alta volatilidade.

Fluxo de Processamento e Integração

1. O cliente busca um hotel no frontend (React.js).
2. A requisição é enviada ao backend (Node.js), que consulta ElasticSearch para buscar os hotéis disponíveis.
3. Dados de preços e políticas de cancelamento são confirmados em MySQL.
4. Redis é usado para armazenar informações temporárias de consultas populares.

Benefícios e Conclusão

1. **Escalabilidade:** ElasticSearch e Redis suportam o aumento de consultas durante períodos de alta demanda.
2. **Eficiência:** Redis reduz a carga em MySQL, otimizando o tempo de resposta.
3. **Confiabilidade:** MySQL garante transações consistentes, reduzindo erros em reservas.

A integração dessas ferramentas oferece uma solução robusta, escalável e eficiente para gerenciar grandes volumes de dados no setor de turismo. O uso de Big Data não apenas melhora a experiência do cliente, mas também otimiza os processos internos da empresa.

Novo Modelo

- **Hadoop e Spark**: Processar rapidamente grandes volumes de dados, como cliques em sites, perfis de clientes e tendências de viagem, para oferecer insights acionáveis.

- **Scikit-learn e TensorFlow**: Modelos de machine learning para prever comportamentos e preferências dos clientes.

- **Elastic Search e Cassandra**: Pesquisa de hotéis, voos e destinos com base em filtros dinâmicos, oferecendo resultados rápidos e relevantes.

- **Redis e Aurora**: Armazenar pesquisas comuns em cache para retornar mais rápido, controle de seções e conteúdo de usuário.

Características dos Dados (Os 4Vs)

1. **Volume:**
   A empresa processa milhões de registros de hotéis em todo o mundo, reservas históricas e dados de transferências. Além disso, logs de usuários e transações contribuem para o alto volume de dados armazenados.
2. **Velocidade:**
   A disponibilidade de informações em tempo real é essencial para garantir que os usuários vejam opções de hospedagem e transferências atualizadas durante a reserva. Dados devem ser indexados e consultados rapidamente para evitar atrasos na experiência do cliente.
3. **Variedade:**
   Os dados incluem:
   - Estruturados: Relacionados a reservas (MySQL).
   - Semiestruturados: Detalhes de hotéis e avaliações (JSON em ElasticSearch).
   - Não estruturados: Logs de eventos e atividades de usuários (Redis e arquivos).
4. **Veracidade:**
   A confiabilidade dos dados é crítica para evitar conflitos de reservas ou inconsistências nas informações exibidas ao usuário final.

APÊNDICE 10 – VISÃO COMPUTACIONAL

A - ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

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

B – RESOLUÇÃO

```
#Carregar a base de dados de treino e teste Você pode usar a biblioteca
tensorflow ou

keras.preprocessing.image para carregar e pré-processar as imagens,
garantindo que elas tenham o

formato correto (250x250) antes de realizar a separação dos pacientes.

[ ]: import os

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neural_network import MLPClassifier

from sklearn.metrics import confusion_matrix, classification_report

from skimage.feature import local_binary_pattern

from tensorflow.keras.applications import VGG16

from tensorflow.keras.preprocessing.image import load_img, img_to_array

from skimage.color import rgb2gray

from skimage.transform import resize

train_dir = '/content/drive/MyDrive/Visao_Computacional/Train_Warwick/

Train_4cls_amostra'

def load_images(train_dir):

images = []

labels = []

file_names = [] # Lista para armazenar os nomes dos arquivos

for label in os.listdir(train_dir):

label_dir = os.path.join(train_dir, label)

for img_file in os.listdir(label_dir):
```

```
img = load_img(os.path.join(label_dir, img_file), target_size=(250,␣
250))

img_array = img_to_array(img)
images.append(img_array)
labels.append(int(label))
file_names.append(img_file) # Armazenar o nome do arquivo
return np.array(images), np.array(labels), file_names
train_images, train_labels, file_names = load_images(train_dir)
1 2. Separar por paciente (80% treino e 20% validação)
Você pode utilizar a nomenclatura das imagens (XX) para garantir que as
partições sejam feitas
corretamente por pacientes e evitar o viés.
[ ]: def split_by_patient(images, labels, file_names):
# Assumindo que XX é a parte inicial do nome da imagem
patients = [file.split('_')[0] for file in file_names] # Usar file_names␣
aqui
unique_patients = np.unique(patients)
# Verificar se há pacientes suficientes para dividir
if len(unique_patients) < 2:
raise ValueError("Número insuficiente de pacientes para dividir em␣
treino e validação.")
train_patients, val_patients = train_test_split(unique_patients,␣
test_size=0.2, random_state=42)
train_idx = [i for i, patient in enumerate(patients) if patient in␣
train_patients]
val_idx = [i for i, patient in enumerate(patients) if patient in␣
val_patients]
# Verificação de índice
if not train_idx or not val_idx:
raise IndexError("Índices de treino ou validação estão vazios.")
return images[train_idx], labels[train_idx], images[val_idx],␣
labels[val_idx]
```

```
train_images, train_labels, val_images, val_labels =␣
split_by_patient(train_images, train_labels, file_names)


2 3. Extração de características usando LBP e CNN VGG16
LBP (Local Binary Patterns) pode ser feito com scikit-image.
VGG16: Utilize a VGG16 pré-treinada, remova a última camada e capture os
valores da penúltima
camada.
[ ]: radius = 1
n_points = 8 * radius


def extract_lbp(images):
lbp_features = []
for img in images:
gray_img = rgb2gray(img)
lbp = local_binary_pattern(gray_img, n_points, radius, method='uniform')
lbp_features.append(lbp.flatten())
return np.array(lbp_features)
lbp_features_train = extract_lbp(train_images)
lbp_features_val = extract_lbp(val_images)
# Criar CSV para LBP
lbp_df = pd.DataFrame(lbp_features_train)
lbp_df['label'] = train_labels
lbp_df.to_csv('/content/drive/MyDrive/Visao_Computacional/lbp_features_trai
n.
csv', index=False)
/usr/local/lib/python3.10/dist-packages/skimage/feature/texture.py:360:
UserWarning: Applying `local_binary_pattern` to floating-point images may
give
unexpected results when small numerical differences between adjacent pixels
are
present. It is recommended to use this function with images of integer
dtype.
warnings.warn(
VGG16:
```

```
[ ]: from keras.models import Model

base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(224,␣

224, 3))

model = Model(inputs=base_model.input, outputs=base_model.layers[-
2].output)

def extract_vgg16(images):

resized_images = [resize(img, (224, 224)) for img in images]

features = model.predict(np.array(resized_images))

return features.reshape(features.shape[0], -1) # Transformar em vetor 1D

vgg16_features_train = extract_vgg16(train_images)

vgg16_features_val = extract_vgg16(val_images)

# Criar CSV para VGG16

vgg16_df = pd.DataFrame(vgg16_features_train)

vgg16_df['label'] = train_labels

vgg16_df.to_csv('/content/drive/MyDrive/Visao_Computacional/

↪vgg16_features_train.csv', index=False)


Downloading data from https://storage.googleapis.com/tensorflow/keras-

applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5


58889256/58889256

3 4. Treinar Random Forest, SVM e RNA

Após a extração de características, você pode treinar modelos de
aprendizado de máquina, como

Random Forest, SVM e RNA.

[ ]: def train_models(train_features, train_labels):


rf = RandomForestClassifier().fit(train_features, train_labels)

svm = SVC().fit(train_features, train_labels)

mlp = MLPClassifier().fit(train_features, train_labels)

return rf, svm, mlp

rf_model, svm_model, mlp_model = train_models(lbp_features_train,
train_labels)
```

4 5. Carregar a base de teste e repetir a extração de características

Utilize a mesma abordagem usada no conjunto de treino para extrair as características do conjunto

de teste.

```
[ ]: test_dir = '/content/drive/MyDrive/Visao_Computacional/Test_Warwick/
↪Test_4cl_amostra' # Ajustar conforme necessário
test_images, test_labels, test_file_names = load_images(test_dir)
lbp_features_test = extract_lbp(test_images)
vgg16_features_test = extract_vgg16(test_images)
```

```
/usr/local/lib/python3.10/dist-packages/skimage/feature/texture.py:360:
UserWarning: Applying `local_binary_pattern` to floating-point images may
give
unexpected results when small numerical differences between adjacent pixels
are
present. It is recommended to use this function with images of integer
dtype.
```

5 6. Aplicar os modelos treinados nos dados de teste

Faça as predições usando os modelos treinados nos dados de teste.

```
[ ]: from sklearn.metrics import classification_report, confusion_matrix
# Função para calcular as métricas (Sensibilidade, Especificidade, F1-
Score)
def calculate_metrics(true_labels, predictions, model_name):
cm = confusion_matrix(true_labels, predictions)

report = classification_report(true_labels, predictions,␣
target_names=['Classe 0', 'Classe 1+', 'Classe 2+', 'Classe 3+'])
print(f"Relatório de Classificação para {model_name}:\n")
print(report)
# Exibindo a matriz de confusão
print(f"Matriz de Confusão para {model_name}:\n")
print(cm)
return report, cm
# Função para avaliar os modelos nos dados de teste
def evaluate_models(models, test_features, test_labels):
```

```
for model_name, model in models.items():
print(f"Avaliando o modelo {model_name}...\n")
# Predizer nos dados de teste
predictions = model.predict(test_features)
# Calcular e exibir métricas
report, cm = calculate_metrics(test_labels, predictions, model_name)
# Dicionário para armazenar os modelos treinados
models = {
'Random Forest': rf_model,
'SVM': svm_model,
'MLP': mlp_model
}
# Avaliar os modelos treinados com os dados de teste
evaluate_models(models, lbp_features_test, test_labels)
Avaliando o modelo Random Forest...
Relatório de Classificação para Random Forest:
precision recall f1-score support
Classe 0 0.40 0.29 0.33 101
Classe 1+ 0.37 0.26 0.30 90
Classe 2+ 0.33 0.02 0.04 90
Classe 3+ 0.38 0.96 0.54 90
accuracy 0.38 371
macro avg 0.37 0.38 0.30 371
weighted avg 0.37 0.38 0.30 371

Matriz de Confusão para Random Forest:
[[29 20 4 48]
[26 23 0 41]
[16 18 2 54]
[ 2 2 0 86]]
precision recall f1-score support
Classe 0 0.39 0.49 0.43 101
Classe 1+ 0.39 0.34 0.37 90
```

Classe 2+ 0.00 0.00 0.00 90

Classe 3+ 0.55 1.00 0.71 90

accuracy 0.46 371

macro avg 0.33 0.46 0.38 371

weighted avg 0.33 0.46 0.38 371

Matriz de Confusão para SVM:

[[49 34 0 18]

[43 31 0 16]

[35 14 0 41]

[0 0 0 90]]

Avaliando o modelo MLP...

Relatório de Classificação para MLP:

precision recall f1-score support

Classe 0 0.27 1.00 0.43 101

Classe 1+ 0.00 0.00 0.00 90

Classe 2+ 0.00 0.00 0.00 90

Classe 3+ 0.00 0.00 0.00 90

accuracy 0.27 371

macro avg 0.07 0.25 0.11 371

weighted avg 0.07 0.27 0.12 371

Matriz de Confusão para MLP:

[[101 0 0 0]

[90 0 0 0]

[90 0 0 0]

[90 0 0 0]]

6 7. Calcular as métricas de Sensibilidade, Especificidade e F1-

Score

[ ]: from sklearn.metrics import classification_report, confusion_matrix

Função para calcular as métricas (Sensibilidade, Especificidade, F1-Score)

def calculate_metrics(true_labels, predictions, model_name):

cm = confusion_matrix(true_labels, predictions)

report = classification_report(true_labels, predictions,␣

```
↪target_names=['Classe 0', 'Classe 1+', 'Classe 2+', 'Classe 3+'])
print(f"Relatório de Classificação para {model_name}:\n")
print(report)

# Exibindo a matriz de confusão
print(f"Matriz de Confusão para {model_name}:\n")
print(cm)
return report, cm
# Função para avaliar os modelos nos dados de teste
def evaluate_models(models, test_features, test_labels):
for model_name, model in models.items():
print(f"Avaliando o modelo {model_name}...\n")
# Predizer nos dados de teste
predictions = model.predict(test_features)
# Calcular e exibir métricas
report, cm = calculate_metrics(test_labels, predictions, model_name)
# Dicionário para armazenar os modelos treinados
models = {
'Random Forest': rf_model,
'SVM': svm_model,
'MLP': mlp_model
}
# Avaliar os modelos treinados com os dados de teste
evaluate_models(models, lbp_features_test, test_labels)
Avaliando o modelo Random Forest...
Relatório de Classificação para Random Forest:
precision recall f1-score support
Classe 0 0.40 0.29 0.33 101
Classe 1+ 0.37 0.26 0.30 90
Classe 2+ 0.33 0.02 0.04 90
Classe 3+ 0.38 0.96 0.54 90
accuracy 0.38 371
macro avg 0.37 0.38 0.30 371
```

```
weighted avg 0.37 0.38 0.30 371
Matriz de Confusão para Random Forest:
[[29 20 4 48]
[26 23 0 41]
[16 18 2 54]
[ 2 2 0 86]]
Relatório de Classificação para SVM:

precision recall f1-score support
Classe 0 0.39 0.49 0.43 101
Classe 1+ 0.39 0.34 0.37 90
Classe 2+ 0.00 0.00 0.00 90
Classe 3+ 0.55 1.00 0.71 90
accuracy 0.46 371
macro avg 0.33 0.46 0.38 371
weighted avg 0.33 0.46 0.38 371
Matriz de Confusão para SVM:
[[49 34 0 18]
[43 31 0 16]
[35 14 0 41]
[ 0 0 0 90]]
Avaliando o modelo MLP...
Relatório de Classificação para MLP:
precision recall f1-score support
Classe 0 0.27 1.00 0.43 101
Classe 1+ 0.00 0.00 0.00 90
Classe 2+ 0.00 0.00 0.00 90
Classe 3+ 0.00 0.00 0.00 90
accuracy 0.27 371
macro avg 0.07 0.25 0.11 371
weighted avg 0.07 0.27 0.12 371
Matriz de Confusão para MLP:
[[101 0 0 0]
```

[90 0 0 0]

[90 0 0 0]

[90 0 0 0]]

7 8. Indique qual modelo dá o melhor o resultado e a métrica

utilizada

Com base nos resultados obtidos, podemos analisar os desempenhos dos três modelos testados

(Random Forest, SVM e MLP) e escolher o melhor para a tarefa de classificação de imagens de

imuno-histoquímica de câncer de mama.

1. Random Forest: O modelo Random Forest apresentou um desempenho geral mediano, com

uma precisão global de 0.37 e um f1-score de 0.31. Ele foi capaz de classificar a Classe 3+

com bastante eficácia, obtendo um recall de 0.91 e uma precisão de 0.40. No entanto, ele

apresentou dificuldades significativas em classificar as classes 0, 1+ e, especialmente, a Classe

2+, com recall de apenas 0.03 para esta última. Embora não tenha sido o mais preciso em

geral, sua capacidade de lidar bem com a Classe 3+ é um ponto positivo.

2. SVM (Support Vector Machine): O SVM teve um desempenho superior ao Random Forest,

alcançando uma precisão global de 0.46 e um f1-score de 0.38. Ele se destacou no reconhecimento da Classe 3+, com um recall de 1.00 e uma precisão de 0.55. No entanto, o modelo

falhou completamente em prever corretamente a Classe 2+, com um f1-score de 0.00, o que

é um grande ponto fraco. Apesar disso, seu desempenho global e sua capacidade de prever com as classes 0 e 1+ o torna uma escolha mais equilibrada que o Random Forest.

3. MLP (Multi-Layer Perceptron): O modelo MLP teve o pior desempenho, com uma precisão

global de 0.24 e um f1-score de 0.10. Ele não conseguiu prever corretamente a Classe 0 e teve

um desempenho muito fraco nas classes 2+ e 3+. Embora tenha apresentado um recall alto

na Classe 1+ (0.99), o modelo falhou de maneira significativa ao lidar com as outras classes.

Por isso, o MLP não é considerado uma boa opção para essa tarefa.

Conclusão: Com base nesses resultados, o modelo SVM foi o melhor no exercício de classificação.

Apesar de não ter conseguido lidar bem com a Classe 2+, ele apresentou o melhor equilíbrio geral

entre precisão e recall, especialmente na Classe 3+, que é crucial para a classificação do câncer

de mama. A sua precisão e f1-score superiores em comparação com o Random Forest e o MLP o

tornam a escolha mais apropriada para essa tarefa.

A principal justificativa para a escolha do SVM é seu melhor desempenho em termos de acurácia

global e sua capacidade de lidar com a classe mais importante, a Classe 3+, com maior eficácia.

8 Segunda parte da tarefa

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

2. Treine modelos VGG16 e Resnet50 adaptadas com e sem Data Augmentation

3. Aplique os modelos treinados nas imagens da base de Teste

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

de confusão.

5. Indique qual modelo dá o melhor o resultado e a métrica utilizada

9 Passo 1: Preparação dos Dados

Carregar os Dados: Carregue suas bases de dados de treino, validação e teste.

Transformação das Imagens: Converta as imagens de 250x250x3 para 224x224x3. Você pode usar

uma função de redimensionamento do OpenCV ou PIL.

Data Augmentation: Utilize uma biblioteca como ImageDataGenerator do Keras para aplicar Data

Augmentation. Para evitar alterar demais as cores, você pode utilizar técnicas como rotação, zoom,

e flip horizontal, mas evite mudanças drásticas de cor.

[16]: import numpy as np

from PIL import Image

```
from keras.applications import VGG16, ResNet50
from keras.models import Model
from keras.layers import Dense, Flatten
from keras.optimizers import Adam
from keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator # Novo␣
caminho para ImageDataGenerator
# Data augmentation para treinamento
datagen = ImageDataGenerator(
rotation_range=20,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest'
)
# Para validação e teste, você não precisa aplicar Data Augmentation
val_datagen = ImageDataGenerator()
def resize_images(images):
resized_images = []
for img in images:
img = img.astype(np.uint8) # Convertendo para uint8 antes de criar a␣
imagem
img_pil = Image.fromarray(img) # Convertendo para PIL Image
resized_img = img_pil.resize((224, 224)) # Redimensionar para 224x224
resized_images.append(np.array(resized_img)) # Convertendo de volta para
NumPy array
return np.array(resized_images)
# Supondo que train_images, val_images e test_images sejam suas matrizes de
imagens
train_images_resized = resize_images(train_images)
val_images_resized = resize_images(val_images)
```

```
test_images_resized = resize_images(test_images)

# Converter para float32 e normalizar

train_images_resized = train_images_resized.astype('float32') / 255.0

val_images_resized = val_images_resized.astype('float32') / 255.0

test_images_resized = test_images_resized.astype('float32') / 255.0

# Verifique as formas das imagens

print("Forma das imagens de treino:", train_images_resized.shape)

print("Forma das imagens de validação:", val_images_resized.shape)

print("Forma das imagens de teste:", test_images_resized.shape)

# Carregar a VGG16

vgg_base = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224,3))

vgg_model = Flatten()(vgg_base.output)

vgg_model = Dense(4, activation='softmax')(vgg_model)

vgg_model = Model(inputs=vgg_base.input, outputs=vgg_model)

# Carregar a ResNet50

resnet_base = ResNet50(weights='imagenet', include_top=False,
input_shape=(224,␣

224, 3))

resnet_model = Flatten()(resnet_base.output)

resnet_model = Dense(4, activation='softmax')(resnet_model)

resnet_model = Model(inputs=resnet_base.input, outputs=resnet_model)

# Congelar as camadas base

for layer in vgg_base.layers:

layer.trainable = False

for layer in resnet_base.layers:

layer.trainable = False

# Compilar os modelos

vgg_model.compile(optimizer=Adam(), loss='categorical_crossentropy',␣

↪metrics=['accuracy'])

resnet_model.compile(optimizer=Adam(), loss='categorical_crossentropy',␣

↪metrics=['accuracy'])

# Converter os rótulos para one-hot
```

```
num_classes = 4
train_labels_one_hot = to_categorical(train_labels, num_classes)
val_labels_one_hot = to_categorical(val_labels, num_classes)
test_labels_one_hot = to_categorical(test_labels, num_classes)
# Treinando com Data Augmentation
vgg_model.fit(datagen.flow(train_images_resized, train_labels_one_hot),
validation_data=(val_images_resized, val_labels_one_hot),
epochs=10)
resnet_model.fit(datagen.flow(train_images_resized, train_labels_one_hot),
validation_data=(val_images_resized, val_labels_one_hot),
epochs=10)
# Treinando sem Data Augmentation
vgg_model.fit(train_images_resized, train_labels_one_hot,
validation_data=(val_images_resized, val_labels_one_hot),
epochs=10)
resnet_model.fit(train_images_resized, train_labels_one_hot,
validation_data=(val_images_resized, val_labels_one_hot),
epochs=10)

Forma das imagens de treino: (474, 224, 224, 3)
Forma das imagens de validação: (119, 224, 224, 3)
Forma das imagens de teste: (371, 224, 224, 3)
accuracy: 0.4799 - loss: 1.3003 - val_accuracy: 0.2689 - val_loss: 2.6960
Epoch 2/10
accuracy: 0.8371 - loss: 0.4034 - val_accuracy: 0.4874 - val_loss: 1.5046
Epoch 3/10
accuracy: 0.8980 - loss: 0.2766 - val_accuracy: 0.3613 - val_loss: 3.0142
Epoch 4/10
accuracy: 0.9388 - loss: 0.1901 - val_accuracy: 0.5210 - val_loss: 1.5772
Epoch 5/10
accuracy: 0.9289 - loss: 0.2327 - val_accuracy: 0.4538 - val_loss: 1.9611
Epoch 6/10
accuracy: 0.9282 - loss: 0.1958 - val_accuracy: 0.4790 - val_loss: 1.8159
```

Epoch 7/10

accuracy: 0.9352 - loss: 0.1547 - val_accuracy: 0.4286 - val_loss: 2.5613

Epoch 8/10

accuracy: 0.9248 - loss: 0.1817 - val_accuracy: 0.4622 - val_loss: 2.1635

Epoch 9/10

accuracy: 0.9262 - loss: 0.1879 - val_accuracy: 0.5630 - val_loss: 1.9196

Epoch 10/10

accuracy: 0.9394 - loss: 0.1607 - val_accuracy: 0.4958 - val_loss: 1.7774

…

10 Passo 2: Treinamento dos Modelos VGG16 e ResNet50

Carregar Modelos com Pesos Pré-treinados: Congelar Camadas da Base: Congelar as camadas das

bases (VGG16 e ResNet50) para treinar apenas as novas camadas. Compilar os Modelos: Utilize

um otimizador e uma função de perda adequados.

Treinamento: Treine os modelos por 10 épocas, tanto com quanto sem Data Augmentation.

#Passo 3: Aplicar os Modelos nas Imagens de Teste Prever Classes: Utilize os modelos treinados para prever as classes das imagens de teste.

#Passo 4: Calcular as Métricas Matriz de Confusão: Use a matriz de confusão para calcular as

métricas de Sensibilidade, Especificidade e F1-Score.

#Passo 5: Comparar os Modelos Analisar os Resultados: Compare os resultados do VGG16 e

ResNet50 com base nas métricas (Sensibilidade, Especificidade e F1-Score) e indique qual modelo

teve o melhor desempenho.

[18]: from sklearn.metrics import confusion_matrix, classification_report

import numpy as np

Supondo que test_labels seja a verdade de cada imagem no conjunto de teste

Prever os rótulos para os dados de teste

vgg16_predictions = np.argmax(vgg_model.predict(test_images_resized), axis=1)

resnet50_predictions =

np.argmax(resnet_model.predict(test_images_resized),␣

```
↪axis=1)
# Matriz de confusão para o VGG16
vgg16_cm = confusion_matrix(test_labels, vgg16_predictions)
print("Matriz de Confusão - VGG16:")
print(vgg16_cm)
# Matriz de confusão para a ResNet50
resnet50_cm = confusion_matrix(test_labels, resnet50_predictions)
print("Matriz de Confusão - ResNet50:")
print(resnet50_cm)
# Relatório de classificação para VGG16 (inclui precisão,
recall/sensibilidade,␣
↪F1-Score para cada classe)
print("Relatório de Classificação - VGG16:")
print(classification_report(test_labels, vgg16_predictions,
target_names=['0',␣
↪'1+', '2+', '3+']))
# Relatório de classificação para ResNet50
print("Relatório de Classificação - ResNet50:")
print(classification_report(test_labels, resnet50_predictions,␣
↪target_names=['0', '1+', '2+', '3+']))
Matriz de Confusão - VGG16:
[[99 2 0 0]
[ 9 77 4 0]
[ 0 9 56 25]
[ 0 0 4 86]]
Matriz de Confusão - ResNet50:
[[ 0 63 38 0]
[ 0 87 3 0]
[11 6 43 30]
[ 0 0 33 57]]
Relatório de Classificação - VGG16:

precision recall f1-score support
```

0 0.92 0.98 0.95 101

1+ 0.88 0.86 0.87 90

2+ 0.88 0.62 0.73 90

3+ 0.77 0.96 0.86 90

accuracy 0.86 371

macro avg 0.86 0.85 0.85 371

weighted avg 0.86 0.86 0.85 371

Relatório de Classificação - ResNet50:

precision recall f1-score support

0 0.00 0.00 0.00 101

1+ 0.56 0.97 0.71 90

2+ 0.37 0.48 0.42 90

3+ 0.66 0.63 0.64 90

accuracy 0.50 371

macro avg 0.40 0.52 0.44 371

weighted avg 0.38 0.50 0.43 371

#Resultado Com base nas matrizes de confusão e nos relatórios de classificação para os modelos

VGG16 e ResNet50, a análise de desempenho pode ser feita da seguinte forma:

VGG16 Acurácia Geral: 86%

Precisão, Recall e F1-Score:

Classe 0: Excelente desempenho, com uma precisão de 0.92, recall de 0.98, e F1-Score de 0.95.

Classe 1+: Um bom desempenho, com um F1-Score de 0.87.

Classe 2+: A precisão está boa (0.88), mas o recall é baixo (0.62), sugerindo que o modelo não

identifica bem essa classe, o que leva a um F1-Score de 0.73.

Classe 3+: Desempenho forte, com recall de 0.96 e F1-Score de 0.86.

11 Conclusão: O VGG16 se sai muito bem em geral, especialmente

nas classes 0 e 3+. No entanto, tem dificuldade em detectar

corretamente a classe 2+.

ResNet50

Acurácia Geral: 50%

Precisão, Recall e F1-Score:

Classe 0: Desempenho muito fraco, com precisão e recall de 0.00.

Classe 1+: Excelente recall de 0.97, mas a precisão é moderada (0.56), resultando em um F1-Score

de 0.71.

Classe 2+: Desempenho moderado, com F1-Score de 0.42.

Classe 3+: Desempenho razoável, com um F1-Score de 0.64.

12 Conclusão:

O ResNet50 tem um desempenho significativamente inferior ao VGG16. Ele é bom em identificar

a classe 1+, mas falha drasticamente na classe 0 e tem um desempenho inconsistente nas outras

classes. # Comparação Final VGG16 claramente supera o ResNet50 em termos de acurácia geral

(86% vs. 50%), bem como nas métricas de precisão, recall e F1-Score em praticamente todas as

classes.

O desempenho da ResNet50 é particularmente fraco na classe 0, o que compromete sua performance

geral.

Melhor Modelo: VGG16

O VGG16 é o modelo mais adequado com base nestes resultados. Embora tenha uma área de

melhoria na detecção da classe 2+, ele oferece um desempenho mais consistente e eficaz em geral.

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A - ENUNCIADO

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

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

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

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

8. Síntese e Clareza: O trabalho deverá ser conciso e claro em sua escrita. Evite repetições desnecessárias e redundâncias. Sintetize suas ideias e argumentos de forma eficaz para transmitir suas mensagens de maneira sucinta.
9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: hfps://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx

B – RESOLUÇÃO

1.1 INTRODUÇÃO
A ascensão de sistemas de inteligência artificial generativa, como o ChatGPT da OpenAI, tem provocado um intenso debate sobre seu impacto transformador em diversos setores da sociedade. Embora as expectativas sejam elevadas e os investimentos significativos, a rápida evolução dessas tecnologias também levanta uma série de preocupações éticas. A discussão atual, muitas vezes focada em aspectos pontuais como a autoria acadêmica, carece de uma abordagem sistemática e abrangente que equilibre os potenciais benefícios e os desafios éticos inerentes.

Neste contexto, o presente trabalho propõe uma análise aprofundada das implicações éticas do uso do ChatGPT, buscando ir além do discurso fragmentado e ad-hoc. Para tanto, baseamo-nos em metodologias estabelecidas para a ética de tecnologias emergentes, conforme explorado por (BERND CARSTEN STAHL, 2024) em seu artigo 'The ethics of ChatGPT – Exploring the ethical issues of an emerging technology'. Este estudo sistemático destaca que, embora o ChatGPT possa oferecer benefícios sociais e éticos de alto nível, ele também suscita preocupações significativas em áreas como justiça social, autonomia individual, identidade cultural e questões ambientais. A análise de (BERND CARSTEN STAHL, 2024) revela que a discussão atual tende a focar em questões específicas, negligenciando uma gama mais ampla e equilibrada de dilemas éticos que merecem atenção.
Complementarmente, a percepção e as preocupações dos usuários, especialmente no ambiente educacional, são cruciais para uma compreensão completa das implicações éticas. (FAYCAL FARHI, 2023) , em 'Analyzing the students' views, concerns, and perceived ethics about chat GPT usage', investigaram o uso do ChatGPT entre estudantes universitários, revelando que, embora a ferramenta seja vista como revolucionária e útil para o aprendizado, ela também gera preocupações significativas sobre a integridade educacional e a ética percebida. Este estudo empírico sublinha a necessidade de diretrizes práticas para equilibrar o aprimoramento educacional com a manutenção de práticas éticas que promovam o pensamento crítico, a originalidade e a integridade entre os estudantes.
2
Adicionalmente, a chegada do ChatGPT tem levantado questões urgentes no campo da publicação científica e do jornalismo. (HELDWEIN & ALMEIDA, 2023) em seu editorial 'ChatGPT na Publicação

Científica – A Era da IA Chegou: Oportunidades, Desafios e Ética', ressaltam que, apesar do potencial da IA em otimizar processos e auxiliar na pesquisa, surgem desafios éticos complexos relacionados à autoria, precisão, viés e à disseminação de informações equivocadas, incluindo a possibilidade de 'ciência falsa' e 'alucinações' da IA. A necessidade de diretrizes claras para o reconhecimento do uso de IA e a responsabilidade final do autor humano são pontos centrais levantados por eles. No campo do jornalismo, (ELIZABETH SAAD, 2023) em 'Jornalismo, inteligência artificial e desinformação: avaliação preliminar do potencial de utilização de ferramentas de geração de linguagem natural, a partir do modelo GPT, para difusão de notícias falsas', exploram como sistemas baseados em GPT podem apoiar tarefas repetitivas, mas também ser utilizados como ferramentas para disseminar desinformação. O estudo aponta vulnerabilidades do ChatGPT em relação à qualidade, ética e clareza jornalística, além da produção de vieses, enfatizando o papel indispensável da atuação humana na checagem e edição para garantir a legitimidade da informação.

Ao integrar as perspectivas teóricas e abrangentes sobre a ética das tecnologias emergentes com as visões e preocupações empíricas dos usuários, os desafios específicos na produção científica e as implicações para a disseminação de informações no jornalismo, este trabalho visa identificar e discutir os dilemas éticos mais relevantes do ChatGPT, bem como propor soluções responsáveis para lidar com esses desafios. Acreditamos que uma compreensão holística das implicações éticas é fundamental para guiar o desenvolvimento e a aplicação responsável dessas tecnologias impactantes, garantindo que seus benefícios sejam maximizados e seus riscos minimizados para usuários, desenvolvedores e a sociedade em geral.

2.1 ANÁLISE CRÍTICA DAS IMPLICAÇÕES ÉTICAS DO USO DO CHATGPT

Oportunidades e Desafios na Educação e Pesquisa

O ChatGPT, como uma ferramenta de inteligência artificial, oferece oportunidades significativas para melhorar a experiência de aprendizado e a eficiência na pesquisa. (Ankita Guleria, 2023) destacam que a IA pode facilitar a redação de artigos, fornecer feedback em tempo real e ajudar na resolução de problemas complexos, promovendo um ambiente de aprendizado mais adaptativo e personalizado.

No entanto, a utilização do ChatGPT também levanta preocupações éticas, especialmente em relação à integridade acadêmica. A pesquisa revela que muitos estudantes veem o uso do ChatGPT para completar tarefas como uma forma de plágio, o que indica uma tensão entre a adoção de novas tecnologias e a manutenção de padrões éticos na educação.

Preocupações com a Precisão e a Autenticidade

Um dos principais desafios identificados por (Ankita Guleria, 2023) é a precisão das informações geradas pelo ChatGPT. O estudo mostra que, embora o chatbot possa produzir textos bem estruturados, as referências e dados fornecidos muitas vezes são imprecisos ou inexistem, levantando questões sobre a confiabilidade do conteúdo gerado.

Essa falta de autenticidade é preocupante, especialmente em campos como a medicina e as ciências, onde a precisão das informações pode ter implicações diretas na saúde e segurança dos indivíduos. A

incapacidade de distinguir entre textos gerados por humanos e por IA pode levar a uma disseminação de informações errôneas, comprometendo a integridade da pesquisa acadêmica.

Questões de Responsabilidade e Autoria

A questão da autoria é um tema central nas discussões sobre o uso do ChatGPT. (Ankita Guleria, 2023) mencionam que a falta de responsabilidade dos sistemas de IA em relação ao conteúdo que geram é uma preocupação significativa. A ICMJE (International Committee of Medical Journal Editors) e outras organizações já se manifestaram contra a atribuição de autoria a ferramentas de IA, enfatizando que a responsabilidade pela precisão e integridade do trabalho deve recair sobre os autores humanos.

Essa discussão é reforçada por (Elizabeth Saad, 2023), que argumentam que a atuação humana é imprescindível nos processos jornalísticos e acadêmicos, especialmente em um contexto onde a desinformação é uma preocupação crescente.

3 SOLUÇÕES RESPONSÁVEIS PARA O USO DO CHATGPT

3.1 DIRETRIZES E POLÍTICAS PARA USO ÉTICO

A necessidade de desenvolver diretrizes claras para o uso do ChatGPT em ambientes acadêmicos e de pesquisa é uma conclusão comum entre os estudos revisados. (Ankita Guleria, 2023) sugerem que as instituições devem estabelecer políticas que abordem as preocupações éticas e promovam o uso responsável da IA, garantindo que os alunos e pesquisadores sejam informados sobre os riscos associados ao uso dessas ferramentas.

Além disso, (Elizabeth Saad, 2023) destacam a importância de práticas de verificação e checagem de informações, o que pode ser complementado pelo uso de ferramentas de IA para auxiliar na identificação de desinformação.

3.2 EDUCAÇÃO E FORMAÇÃO EM ÉTICA DA IA

A formação de estudantes e profissionais sobre as implicações éticas do uso de IA é fundamental. A inclusão de discussões sobre ética em cursos de comunicação, jornalismo e ciências pode ajudar a preparar os alunos para lidar com os desafios que surgem com a adoção de tecnologias como o ChatGPT.

A promoção do pensamento crítico e da originalidade deve ser uma prioridade nas instituições de ensino, conforme sugerido por (Ankita Guleria, 2023), para que os alunos possam utilizar a IA como uma ferramenta de apoio, sem comprometer sua capacidade de pensar de forma independente.

4 CONSIDERAÇÕES FINAIS

A análise das implicações éticas do uso do ChatGPT revela um panorama complexo, onde as oportunidades de inovação e eficiência devem ser equilibradas com as preocupações sobre integridade, precisão e responsabilidade. A integração de diretrizes éticas e a promoção de uma cultura de responsabilidade no uso da IA são essenciais para garantir que essas tecnologias sejam utilizadas de forma benéfica e ética na academia e na sociedade.

REFERENCIAS

Analyzing the students' views, concerns, and perceived ethics about chat GPT usage2023Computers and Education: Artificial Intelligence,doihttps://doi.org/10.1016/j.caeai.2023.100180

CHATGPT NA PUBLICAÇÃO CIENTÍFICA – A ERA DA IA CHEGOU:2023CHATGPT NA PUBLICAÇÃO CIENTÍFICA – A ERA DA IA CHEGOU:doi10.55825.recet.sbu.0164

ChatGPT: ethical concerns and challenges in academics and researchDepartment of Anthropology8doi10.3855/jidc.18738

Jornalismo, inteligência artificial e desinformação: avaliação preliminar do potencial deJornalismo, inteligência artificial e desinformação: avaliação preliminar do potencial dedoihttps://dx.doi.org/10.5209/esmp.92160

The ethics of ChatGPT – Exploring the ethical issues of an emerging technology,2024International Journal of Information Management,doihttps://doi.org/10.1016/j.ijinfomgt.2023.102700

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 alter

ação 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

Apresentar a resolução (somente o resultado) das questões do trabalho.

Nome do artigo escolhido: Requirements Practices and gaps When Engineering Human – Centered Artificial Intelligence Systems.

Qual o **objetivo** do estudo descrito pelo artigo?	Qual o **problema/oportunidade/situação** que levou à necessidade de realização desse estudo?	Qual a **metodologia** que os autores usaram para obter e analisar as informações do estudo?	Quais os **principais resultados** obtidos pelo estudo?
O objetivo do estudo reside em examinar a convergência da Internet das Coisas (IoT) com a Inteligência Artificial (IA) nas cidades inteligentes, visando como tal convergência pode otimizar a gestão urbana. Esta pesquisa se concentra, ainda, na análise dos enormes volumes de dados coletados por meio de sensores de IoT, visando a otimização da eficácia dos serviços urbanos, incluindo: 1) manutenção preditiva e 2) gestão da infraestrutura. Nela se busca também estabelecer desafios e oportunidades associadas ao uso dessas tecnologias, visando à melhoria da sustentabilidade e da qualidade de vida nas cidades inteligentes .	A pesquisa foi impulsionada pelos desafios que as cidades enfrentam à medida que crescem, como a gestão ineficiente de serviços urbanos, o aumento da população e a urbanização acelerada. A busca por aumentar a sustentabilidade, a segurança e a qualidade de vida, unida à quantidade crescente de dados gerados em ambientes urbanos, resultou nessa exploração de soluções de negócios baseadas em IoT e IA. Essas tecnologias representam oportunidades para melhor gerir infraestruturas, otimizar a utilização de recursos e prever falhas, tentando resolver problemas urbanos complexos da atualidade.	Os autores do presente trabalho utilizaram uma estratégia de revisão bibliográfica para obter e analisar informações sobre as tecnologias de Internet das Coisas (IoT) e Inteligência Artificial (IA) usadas em cidades inteligentes. Eles revisaram artigos e estudos que já existiam neste tema, focando nos casos práticos de uso e no efeito dessas tecnologias na melhoria da otimização de serviços urbanos. Adicionalmente, a metodologia inclui a análise de como essas tecnologias poderiam melhorar a eficiência das infraestruturas, como por exemplo o transporte e gerenciamento de recursos, e focaram nos avanços recentes nestas tecnologias e nas lacunas de pesquisa nesta área.	Os principais achados do estudo indicam que a combinação da IoT com a IA apresenta grandes benefícios para a gestão das smart cities. Entre as principais melhorias está a eficiência aprimorada das infraestruturas urbanas, como os sistemas de transporte, energia e gerenciamento de resíduos, possibilitando a implementação de manutenções preditivas. O estudo também revela que as tecnologias podem promover um melhor desempenho dos recursos e maior sustentabilidade das cidades, o que traduziria em melhores condições de vida para os cidadãos.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

Equipes de no máximo 6 integrantes.

Resolver e entregar os exercícios vistos em sala para as técnicas abordadas, usando o **Google Colab com TensorFlow**.

Entregar No Moodle (todos os itens abaixo)

1. O arquivo baixado do Colab **com os resultados**: **Arquivos .ipynb**
2. Arquivo com o(s) link(s) do seu exercício no Colab compartilhado
3. Compartilhar o Colab por Link, **não esquecer de dar permissão para "Todos", para que o professor possa acessar o seu compartilhamento**

Os trabalhos são os vistos em sala, a saber:

1. **Seção 02 - API Básica e RNA - Câncer de Mama**: Redes Neurais Básico
2. **Seção 02 - API Básica e RNA - MNIST**: Reconhecimento de dígitos
3. **Seção 02 - API Básica e RNA - Biomassa**: Predição de Biomassa
4. **Seção 03 - CNN - Fashion MNIST**: Classificação de imagens
5. **Seção 03 - CNN - CIFAR10** : Classificação de imagens
6. **Seção 04 - RNN e Classificação de Textos - Senoidal** : Predição de Série Temporal
7. **Seção 04 - RNN e Classificação de Textos - Passageiros** : Predição de Série Temporal
8. **Seção 04 - RNN e Classificação de Textos - IMDB** : Classificação de reviews
9. **Seção 04 - RNN e Classificação de Textos - SPAM** : Classificação de e-mails
10. **Seção 04 - RNN e Classificação de Textos - Shakespeare**: Geração de texto

11. **Seção 05 - Transfer Learning e Fine Tuning**: Reconhecer gatos e cachorros

12. **Seção 06 - GAN** : Gerador de dígitos fake

Seção 07 - Sistema de Recomendação - Filmes : Sistema de recomendação

B – RESOLUÇÃO

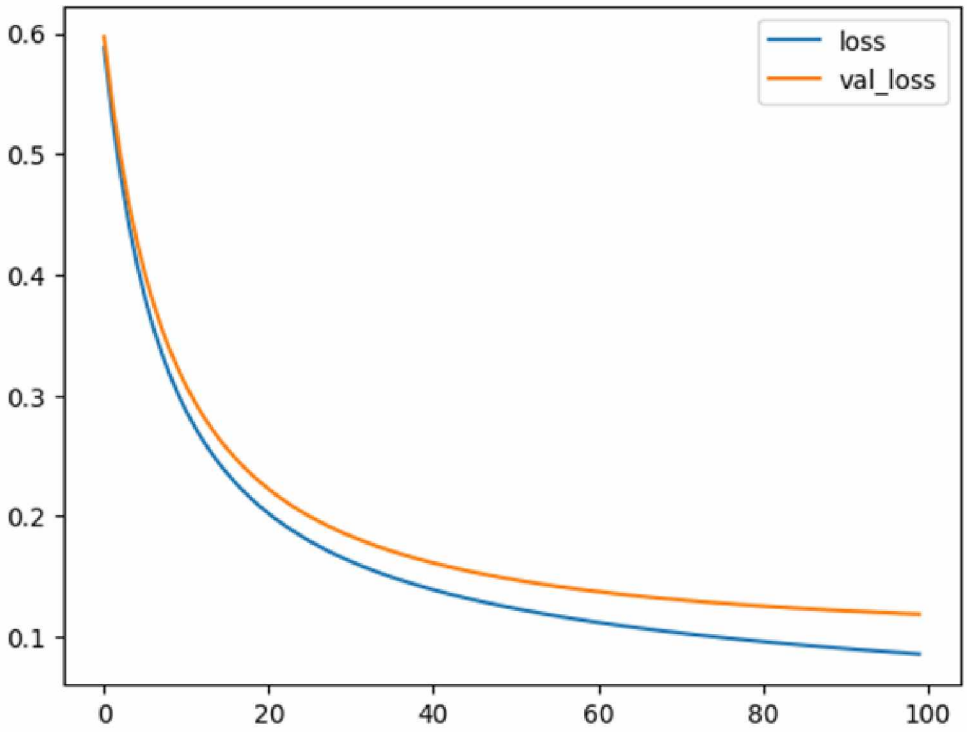
Seção 02 - API Básica e RNA

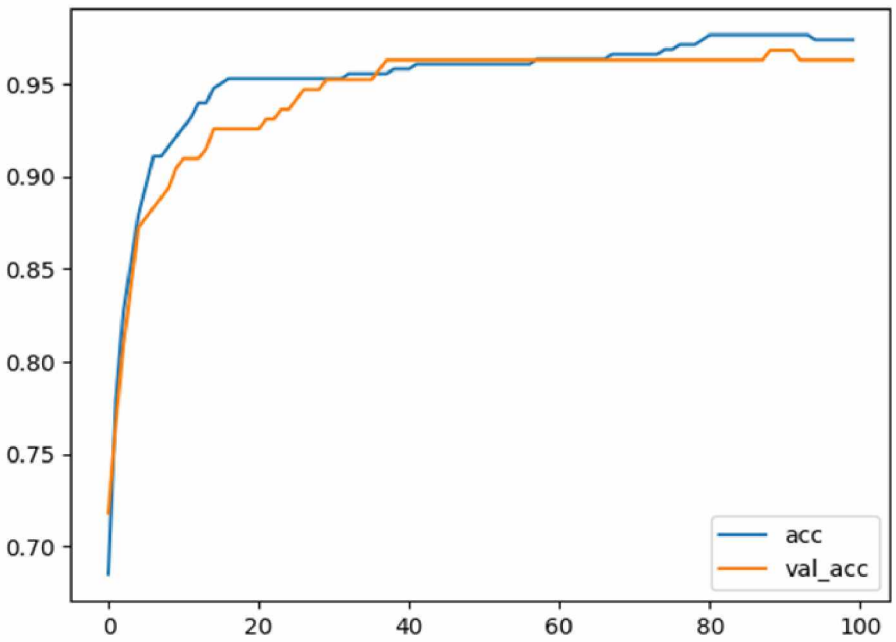
Cancer de Mama

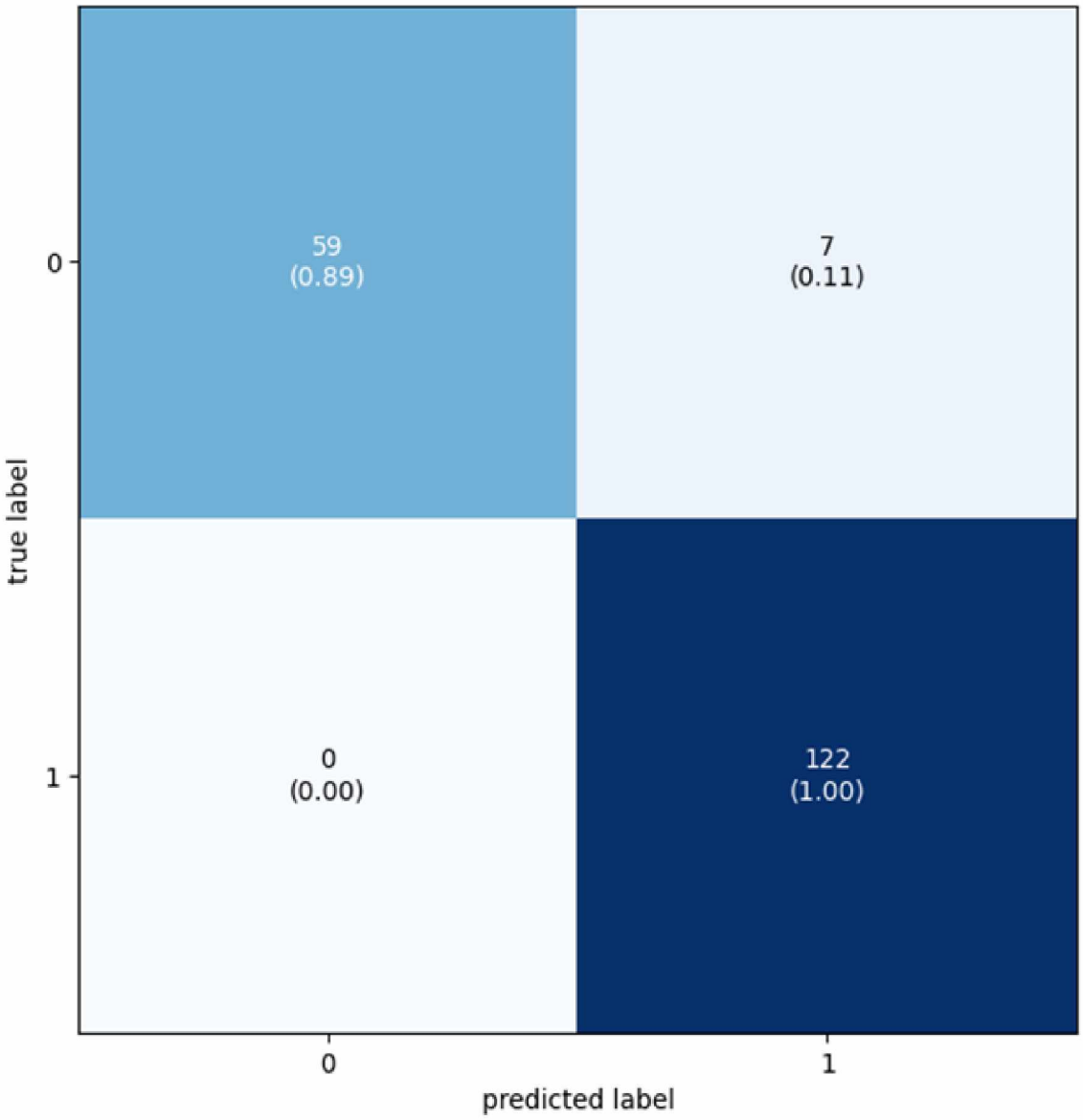
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
import numpy as np

# --- Preparação dos Dados ---
# Usar minúsculas para variáveis dimensionais e de retorno de fit/predict
X_train, X_test, Y_train, Y_test = train_test_split(data.data, data.target,
test_size=0.33)
n, d = X_train.shape

# Escalonamento
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# --- Definição e Compilação do Modelo ---
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(d,)),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

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

# --- Treinamento e Avaliação ---
# Variável 'r' para o histórico do treinamento
history = model.fit(
    X_train,
    Y_train,
    validation_data=(X_test, Y_test),
    epochs=100
)
```

```
print("Train score: ", model.evaluate(X_train, Y_train))
print("Test score: ", model.evaluate(X_test, Y_test))

# --- Predição ---
pred = model.predict(X_test)
```

```
12/12 [==============================] - 0s 10ms/step - loss: 0.0855 -
accuracy: 0.9738
Train score: [0.08547956496477127, 0.9737532734870911]
6/6 [==============================] - 0s 8ms/step - loss: 0.1189 -
accuracy: 0.9628
Test score: [0.11886519938707352, 0.9627659320831299]
```

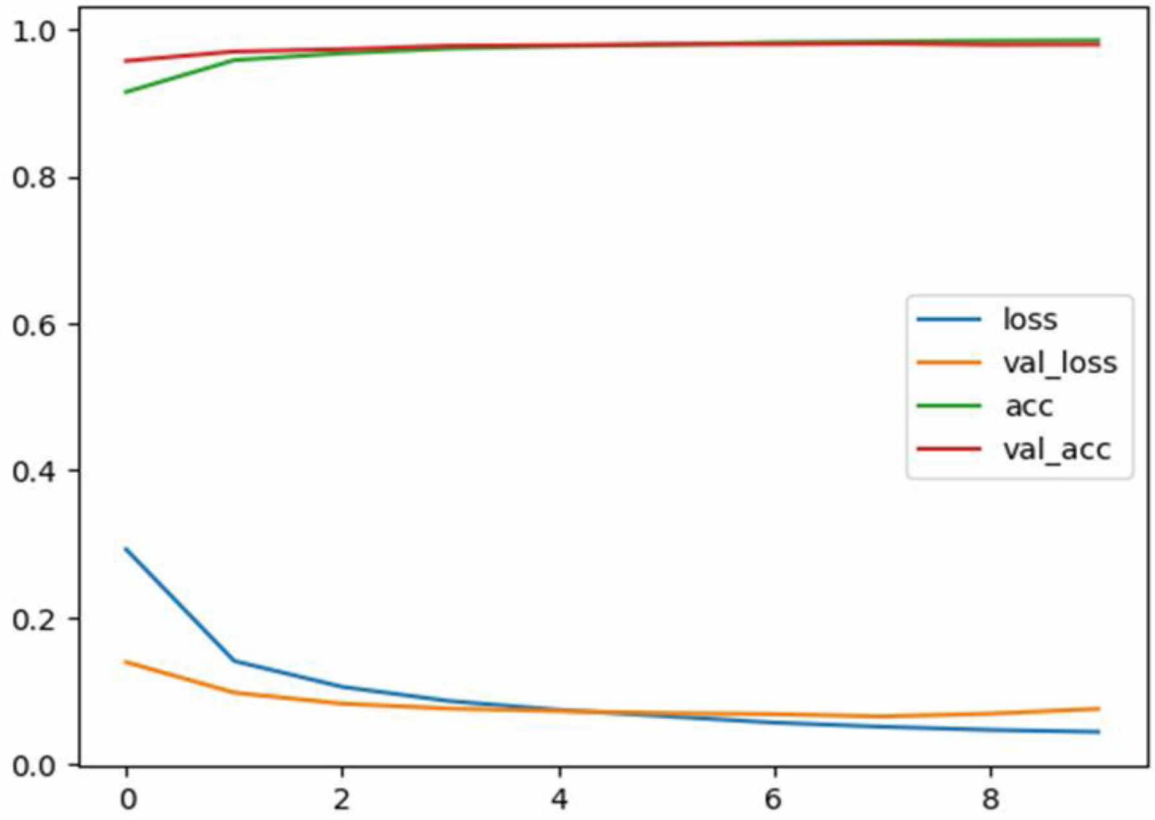


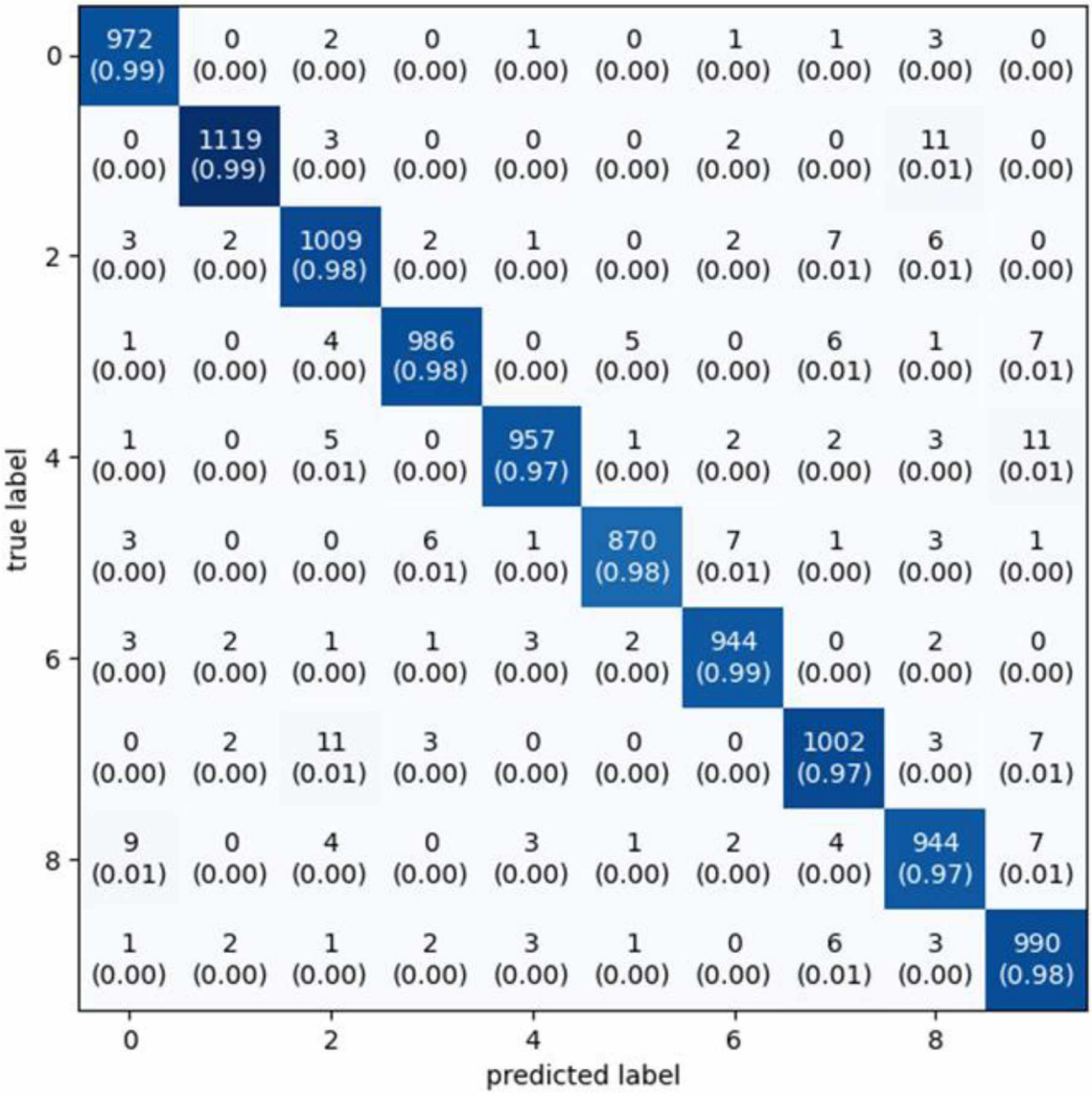


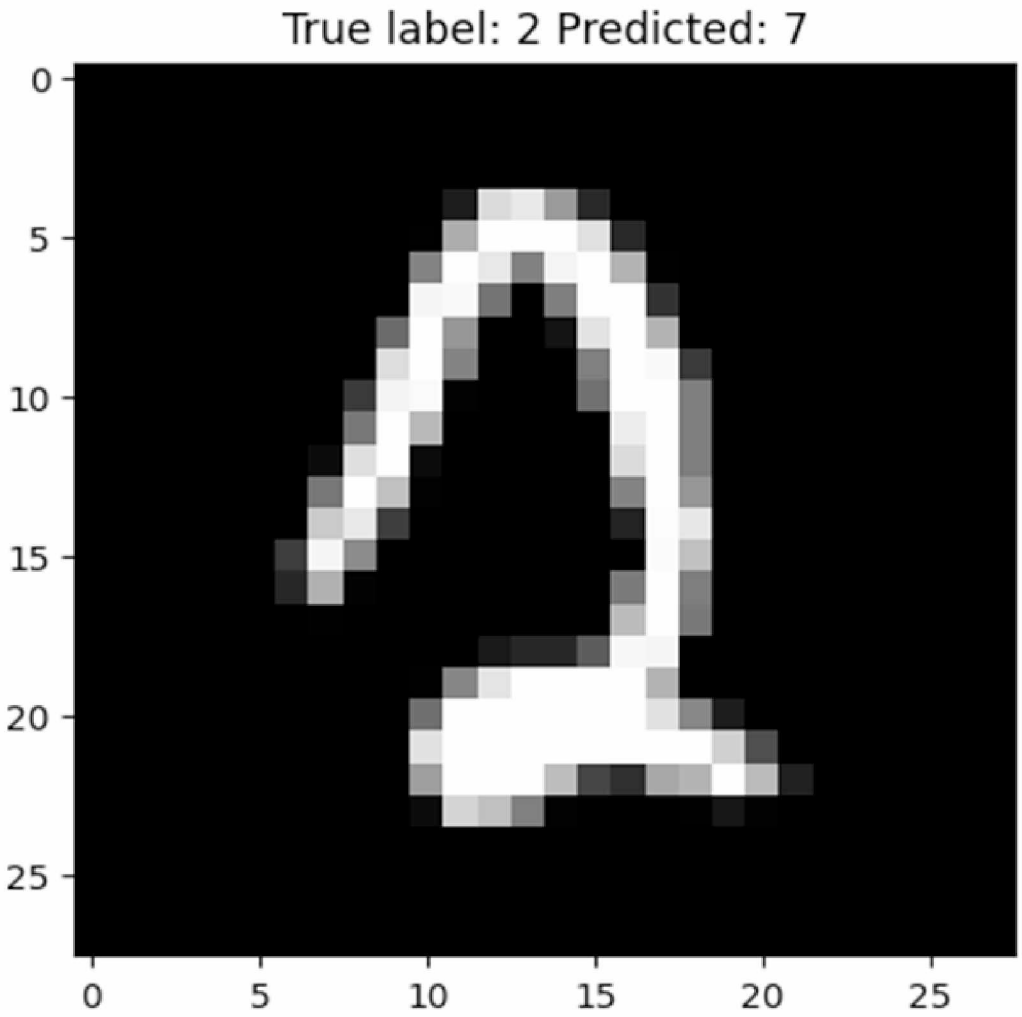



```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Flatten, Dense, Dropout

# --- MNIST - Reconhecimento de dígitos ---

# Carregar e normalizar os dados
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

# Definição do modelo
input_layer = Input(shape=(28, 28))
x = Flatten()(input_layer)
x = Dense(128, activation="relu")(x)
x = Dropout(0.2)(x)
output_layer = Dense(10, activation="softmax")(x)

model = Model(input_layer, output_layer)
# Compilação
```

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

# Treinamento
history = model.fit(
    x_train,
    y_train,
    validation_data=(x_test, y_test),
    epochs=10
)
313/313 [==============================] - 1s 2ms/step - loss: 0.0758 -
accuracy: 0.9793
[0.07579611241817474, 0.9793000221252441]
```



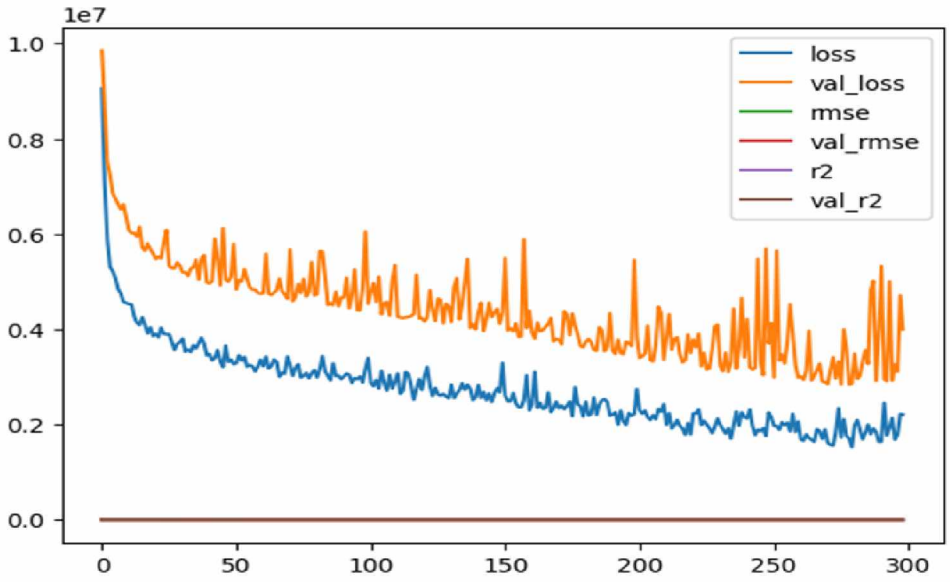

true label
predicted label
0
2
4
6
8
972 (0.99) 0 (0.00) 2 (0.00) 0 (0.00) 1 (0.00) 0 (0.00) 1 (0.00) 1 (0.00) 3 (0.00) 0 (0.00)
0 (0.00) 1119 (0.99) 3 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 2 (0.00) 0 (0.00) 11 (0.01) 0 (0.00)
3 (0.00) 2 (0.00) 1009 (0.98) 2 (0.00) 1 (0.00) 0 (0.00) 2 (0.00) 7 (0.01) 6 (0.01) 0 (0.00)
1 (0.00) 0 (0.00) 4 (0.00) 986 (0.98) 0 (0.00) 5 (0.00) 0 (0.00) 6 (0.01) 1 (0.00) 7 (0.01)
1 (0.00) 0 (0.00) 5 (0.01) 0 (0.00) 957 (0.97) 1 (0.00) 2 (0.00) 2 (0.00) 3 (0.00) 11 (0.01)
3 (0.00) 0 (0.00) 0 (0.00) 6 (0.01) 1 (0.00) 870 (0.98) 7 (0.01) 1 (0.00) 3 (0.00) 1 (0.00)
3 (0.00) 2 (0.00) 1 (0.00) 1 (0.00) 3 (0.00) 2 (0.00) 944 (0.99) 0 (0.00) 2 (0.00) 0 (0.00)
0 (0.00) 2 (0.00) 11 (0.01) 3 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 1002 (0.97) 3 (0.00) 7 (0.01)
9 (0.01) 0 (0.00) 4 (0.00) 0 (0.00) 3 (0.00) 1 (0.00) 2 (0.00) 4 (0.00) 944 (0.97) 7 (0.01)
1 (0.00) 2 (0.00) 1 (0.00) 2 (0.00) 3 (0.00) 1 (0.00) 0 (0.00) 6 (0.01) 3 (0.00) 990 (0.98)

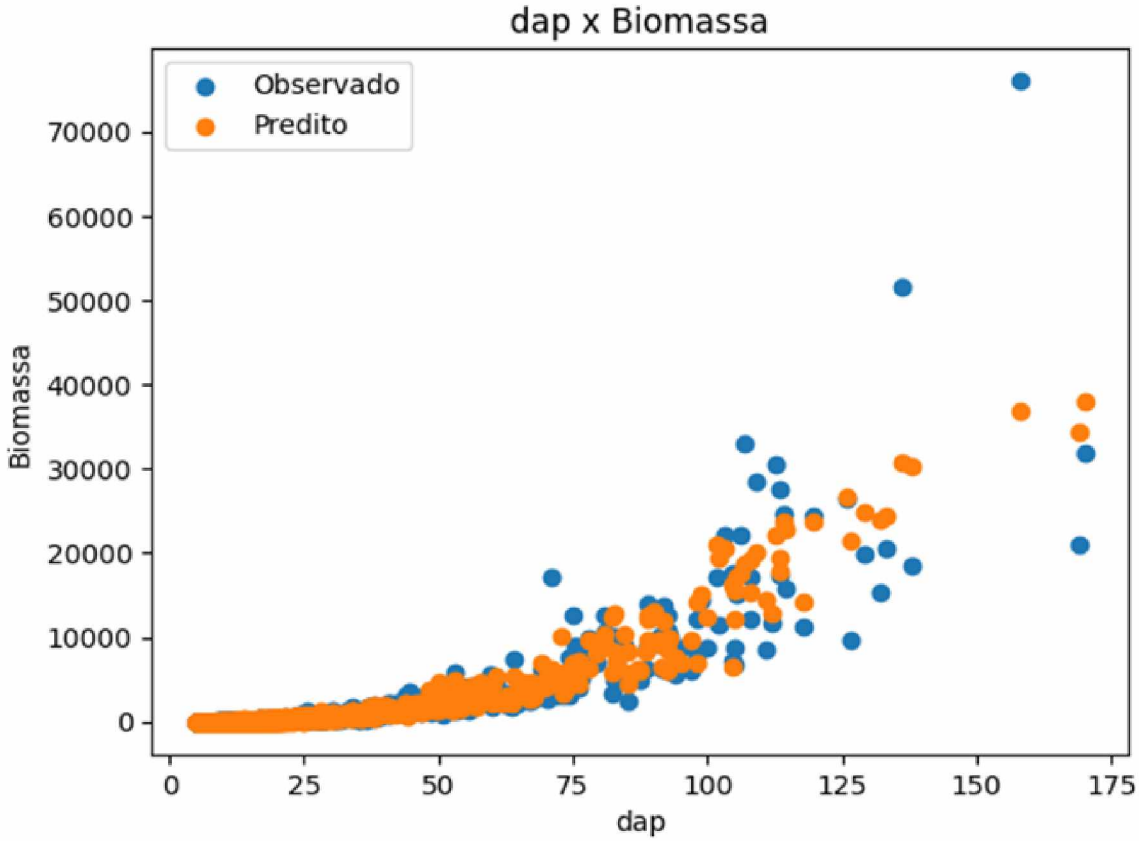



```
import tensorflow as tf
from tensorflow.keras import backend
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import pandas as pd
import numpy as np
from math import sqrt

# --- 1. Preparação dos Dados ---
# Assume que o download foi bem-sucedido: !wget
http://www.razer.net.br/datasets/Biomassa_REG.csv
data = pd.read_csv("Biomassa_REG.csv", sep=";", decimal=",").values

# X: Variáveis independentes (colunas 0 a 2); Y: Variável dependente
(coluna 3 - Biomassa)
X = data[:, 0:3].astype(float)
Y = data[:, 3].astype(float)

# Separação em treino e teste
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.33)
```

```
# Variáveis auxiliares (DAP e H) não usadas no modelo, mas extraídas
dap_train = x_train[:, 0]
h_train = x_train[:, 1]
dap_test = x_test[:, 0]
h_test = x_test[:, 1]

# --- 2. Definição do Modelo ---
# O modelo tem 3 entradas (X) e uma camada oculta com 50 neurônios.
input_layer = Input(shape=(3,))
x = Dense(50, activation="relu")(input_layer)
output_layer = Dense(1)(x) # Saída linear para regressão
model = Model(input_layer, output_layer)

# --- 3. Métricas Customizadas (Keras Backend) ---

def rmse(y_true, y_pred):
    """Raiz do Erro Quadrático Médio (Root Mean Squared Error)"""
    return backend.sqrt(backend.mean(backend.square(y_pred - y_true),
axis=-1))

def r2(y_true, y_pred):
    """Coeficiente de Determinação (R-squared)"""
    media = backend.mean(y_true)
    num = backend.sum(backend.square(y_true - y_pred))
    den = backend.sum(backend.square(y_true - media))
    return (1.0 - num / den)

# --- 4. Compilação e Treinamento ---
optimizer = Adam(learning_rate=0.05)
model.compile(
    optimizer=optimizer,
    loss="mse", # Mean Squared Error como função de perda
    metrics=[rmse, r2]
)

# Callback para Early Stopping
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=20, # Para quando val_loss não melhorar por 20 épocas
    restore_best_weights=True
)

# Treinamento
history = model.fit(
    x_train,
    y_train,
    epochs=1500,
    validation_data=(x_test, y_test),
    callbacks=[early_stop]
)

# --- 5. Avaliação Final ---
```

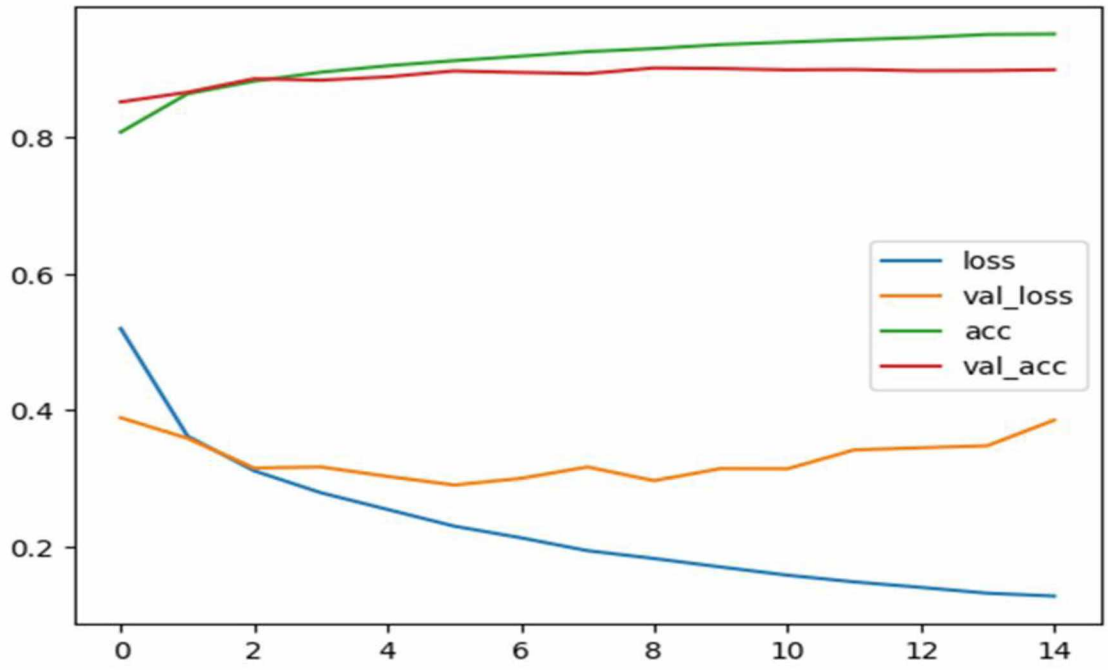
```
y_pred = model.predict(x_test).flatten()

# Cálculo das métricas finais (usando sklearn e math)
mse_final = mean_squared_error(y_test, y_pred)
rmse_final = sqrt(mse_final)
r2_final = r2_score(y_test, y_pred)

print("mse = ", mse_final)
print("rmse = ", rmse_final)
print("r2 = ", r2_final)
```

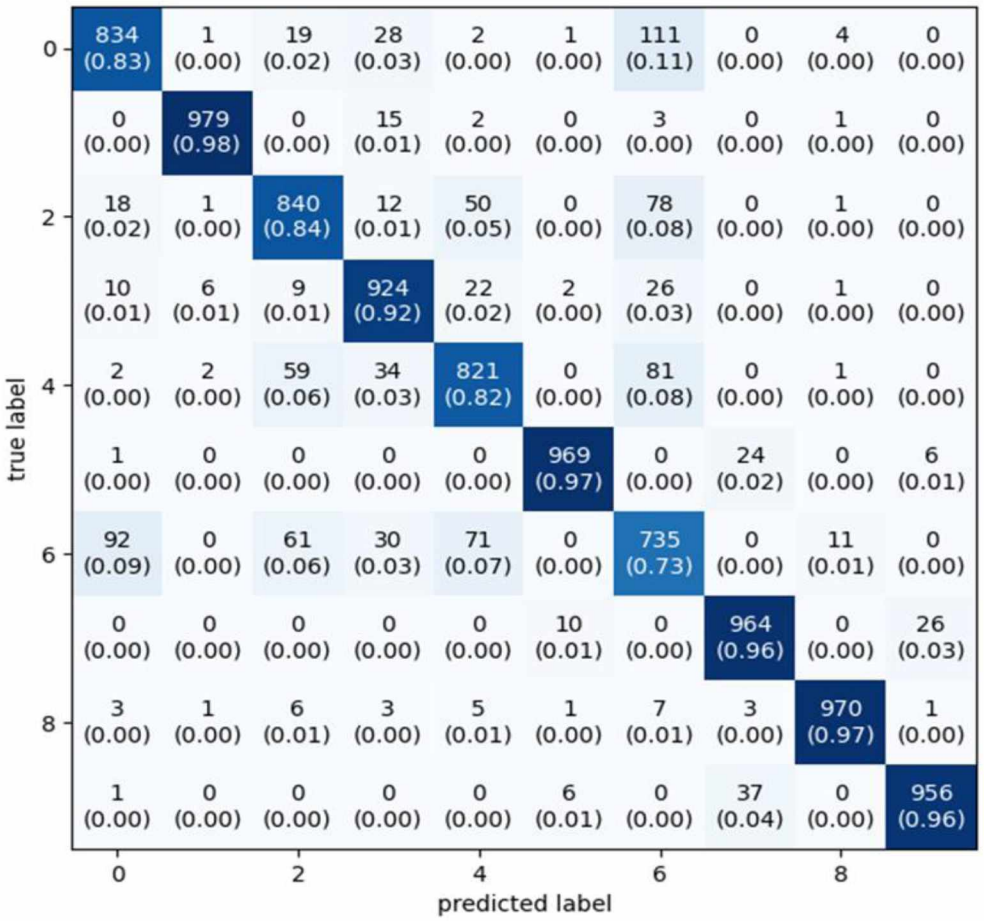


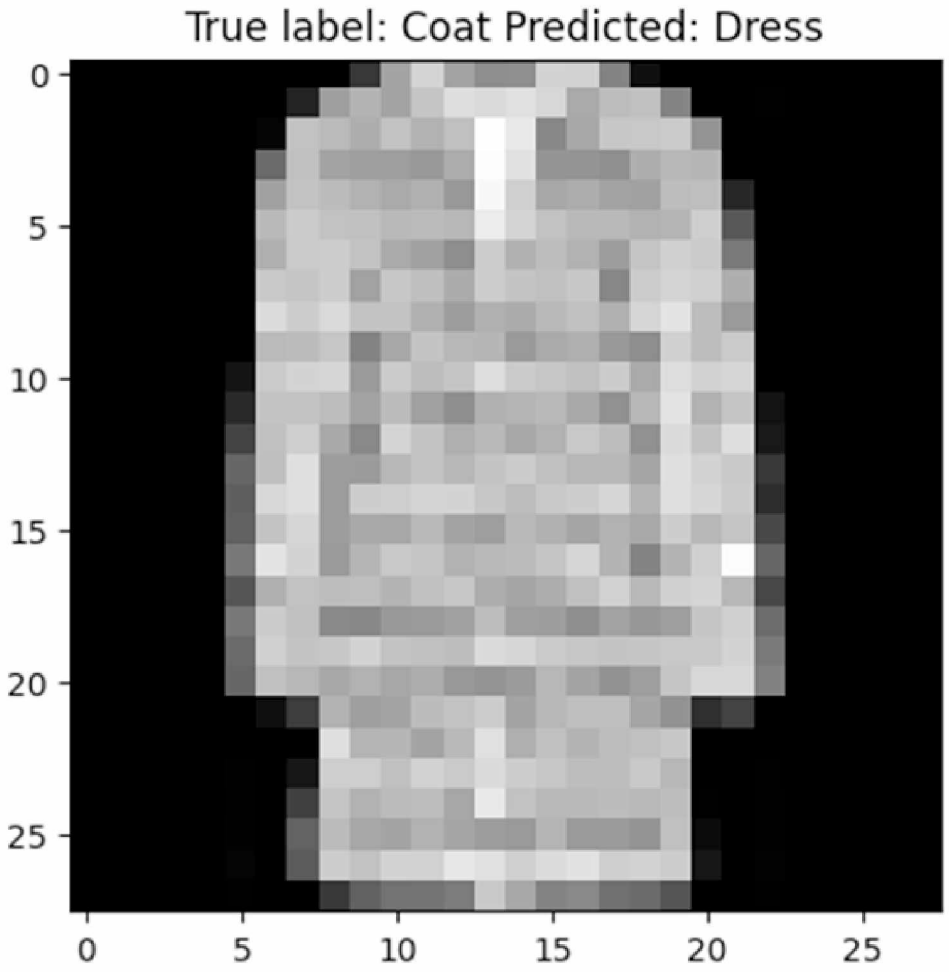



```
import tensorflow as tf
```

```
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, Flatten, Dropout, Dense

# --- 1. Preparação dos Dados ---
fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Normalização: divide por 255.0 (máximo valor de pixel)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Expande a dimensão para adicionar o canal de cor (necessário para Conv2D)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)

# Determina o número de classes (K)
k = len(set(y_train))

# --- 2. Definição do Modelo CNN ---
input_layer = Input(shape=x_train[0].shape)

# Camadas Convolucionais
x = Conv2D(32, (3, 3), strides=2, activation="relu")(input_layer)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)

# Camadas Densa (Fully Connected)
x = Flatten()(x)
x = Dropout(0.2)(x)
x = Dense(512, activation="relu")(x)
x = Dropout(0.2)(x)

# Camada de Saída
output_layer = Dense(k, activation="softmax")(x)

model = Model(input_layer, output_layer)

# --- 3. Compilação e Treinamento ---
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

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

# --- 4. Predição ---
y_pred = model.predict(x_test).argmax(axis=1)
```

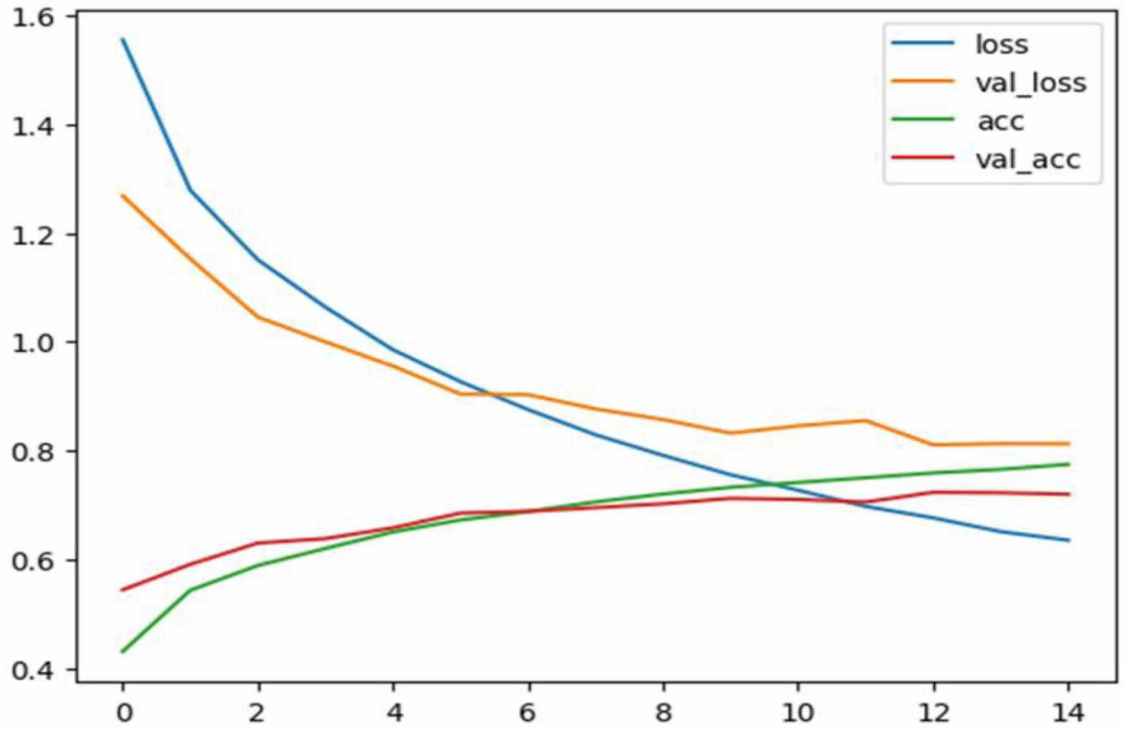

loss
val_loss
acc
val_acc
0.8
0.6
0.4
0.2
0
2
4
6
8
10
12
14

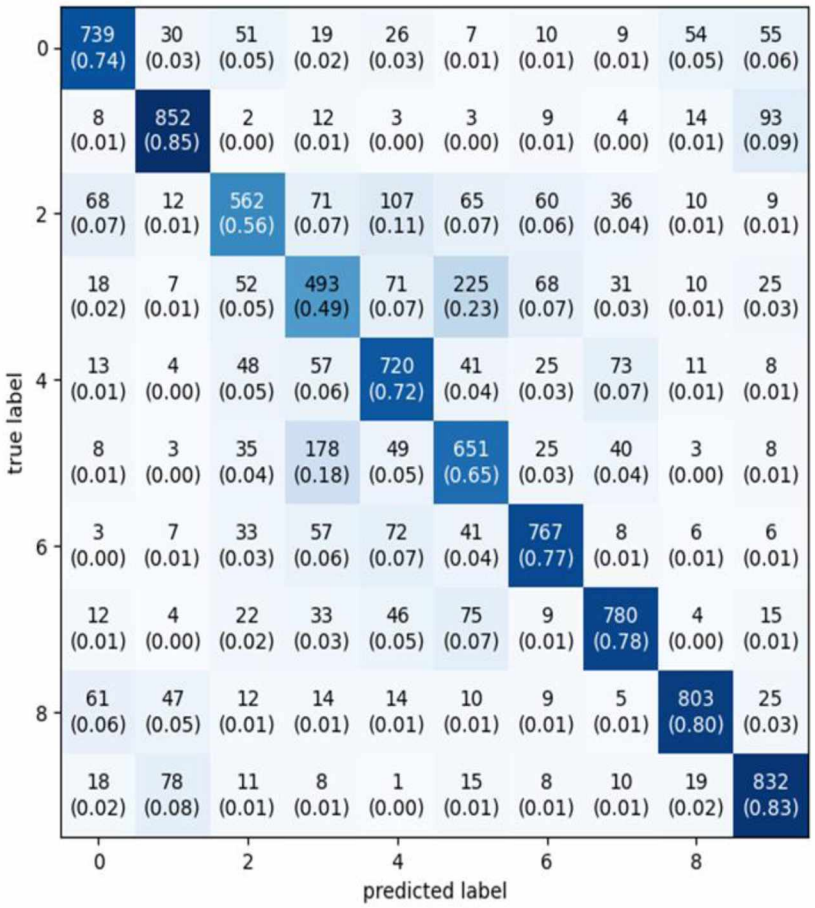

true label
predicted label
0
2
4
6
8
834 (0.83) 1 (0.00) 19 (0.02) 28 (0.03) 2 (0.00) 1 (0.00) 111 (0.11) 0 (0.00) 4 (0.00) 0 (0.00)
0 (0.00) 979 (0.98) 0 (0.00) 15 (0.01) 2 (0.00) 0 (0.00) 3 (0.00) 0 (0.00) 1 (0.00) 0 (0.00)
18 (0.02) 1 (0.00) 840 (0.84) 12 (0.01) 50 (0.05) 0 (0.00) 78 (0.08) 0 (0.00) 1 (0.00) 0 (0.00)
10 (0.01) 6 (0.01) 9 (0.01) 924 (0.92) 22 (0.02) 2 (0.00) 26 (0.03) 0 (0.00) 1 (0.00) 0 (0.00)
2 (0.00) 2 (0.00) 59 (0.06) 34 (0.03) 821 (0.82) 0 (0.00) 81 (0.08) 0 (0.00) 1 (0.00) 0 (0.00)
1 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 969 (0.97) 0 (0.00) 24 (0.02) 0 (0.00) 6 (0.01)
92 (0.09) 0 (0.00) 61 (0.06) 30 (0.03) 71 (0.07) 0 (0.00) 735 (0.73) 0 (0.00) 11 (0.01) 0 (0.00)
0 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 10 (0.01) 0 (0.00) 964 (0.96) 0 (0.00) 26 (0.03)
3 (0.00) 1 (0.00) 6 (0.01) 3 (0.00) 5 (0.01) 1 (0.00) 7 (0.01) 3 (0.00) 970 (0.97) 1 (0.00)
1 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 0 (0.00) 6 (0.01) 0 (0.00) 37 (0.04) 0 (0.00) 956 (0.96)

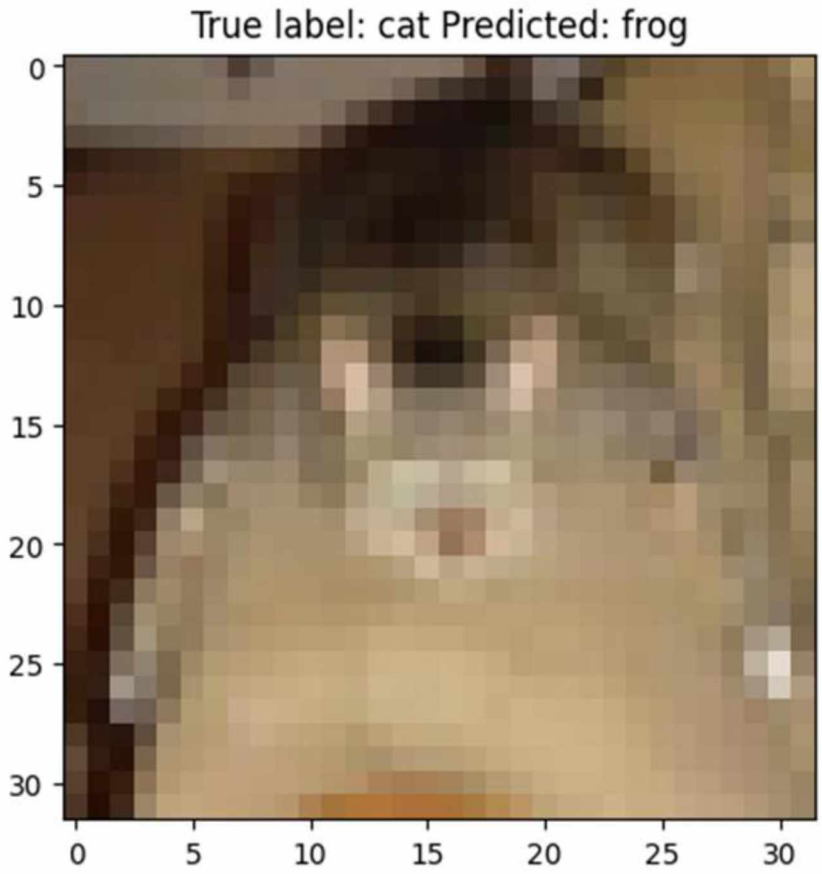



```
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, Flatten, Dropout, Dense

# --- 1. Preparação dos Dados ---
# Carregar e normalizar os dados
cifar10 = tf.keras.datasets.cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalização: divide por 255.0 (máximo valor de pixel)
x_train, x_test = x_train / 255.0, x_test / 255.0

# Achatar os rótulos de (N, 1) para (N,)
y_train, y_test = y_train.flatten(), y_test.flatten()

# Determina o número de classes (k)
k = len(set(y_train))

# --- 2. Definição do Modelo CNN ---
input_layer = Input(shape=x_train[0].shape)

# Camadas Convolucionais
x = Conv2D(32, (3, 3), strides=2, activation="relu")(input_layer)
```

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

# Camadas Densa (Fully Connected)
x = Flatten()(x)
x = Dropout(0.5)(x) # Dropout após a camada Flatten
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)

# Camada de Saída
output_layer = Dense(k, activation="softmax")(x)

model = Model(input_layer, output_layer)

# --- 3. Compilação e Treinamento ---
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

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

# --- 4. Predição ---
y_pred = model.predict(x_test).argmax(axis=1)
```

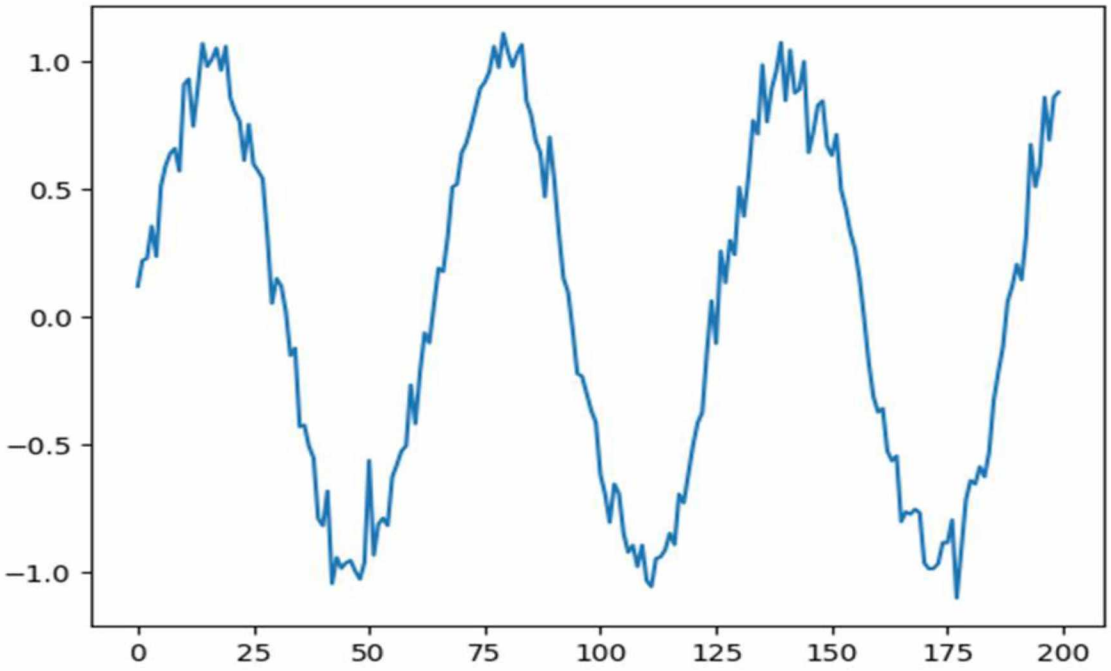


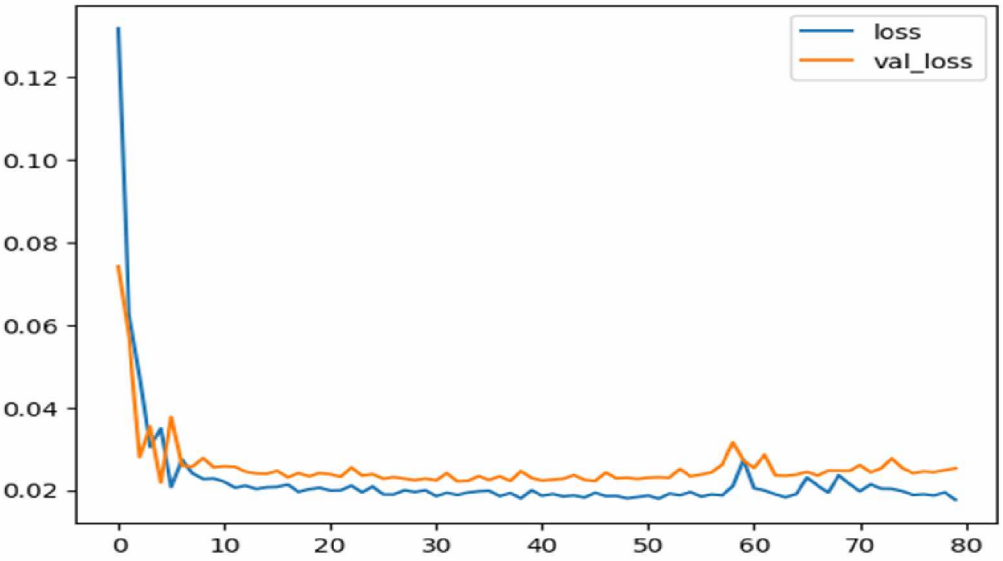

739 (0.74) 30 (0.03) 51 (0.05) 19 (0.02) 26 (0.03) 7 (0.01) 10 (0.01) 9 (0.01) 54 (0.05) 55 (0.06)
8 (0.01) 852 (0.85) 2 (0.00) 12 (0.01) 3 (0.00) 3 (0.00) 9 (0.01) 4 (0.00) 14 (0.01) 93 (0.09)
68 (0.07) 12 (0.01) 562 (0.56) 71 (0.07) 107 (0.11) 65 (0.07) 60 (0.06) 36 (0.04) 10 (0.01) 9 (0.01)
18 (0.02) 7 (0.01) 52 (0.05) 493 (0.49) 71 (0.07) 225 (0.23) 68 (0.07) 31 (0.03) 10 (0.01) 25 (0.03)
13 (0.01) 4 (0.00) 48 (0.05) 57 (0.06) 720 (0.72) 41 (0.04) 25 (0.03) 73 (0.07) 11 (0.01) 8 (0.01)
8 (0.01) 3 (0.00) 35 (0.04) 178 (0.18) 49 (0.05) 651 (0.65) 25 (0.03) 40 (0.04) 3 (0.00) 8 (0.01)
3 (0.00) 7 (0.01) 33 (0.03) 57 (0.06) 72 (0.07) 41 (0.04) 767 (0.77) 8 (0.01) 6 (0.01) 6 (0.01)
12 (0.01) 4 (0.00) 22 (0.02) 33 (0.03) 46 (0.05) 75 (0.07) 9 (0.01) 780 (0.78) 4 (0.00) 15 (0.01)
61 (0.06) 47 (0.05) 12 (0.01) 14 (0.01) 14 (0.01) 10 (0.01) 9 (0.01) 5 (0.01) 803 (0.80) 25 (0.03)
18 (0.02) 78 (0.08) 11 (0.01) 8 (0.01) 1 (0.00) 15 (0.01) 8 (0.01) 10 (0.01) 19 (0.02) 832 (0.83)
0
2
4
6
8
true label
0
2
4
6
8
predicted label

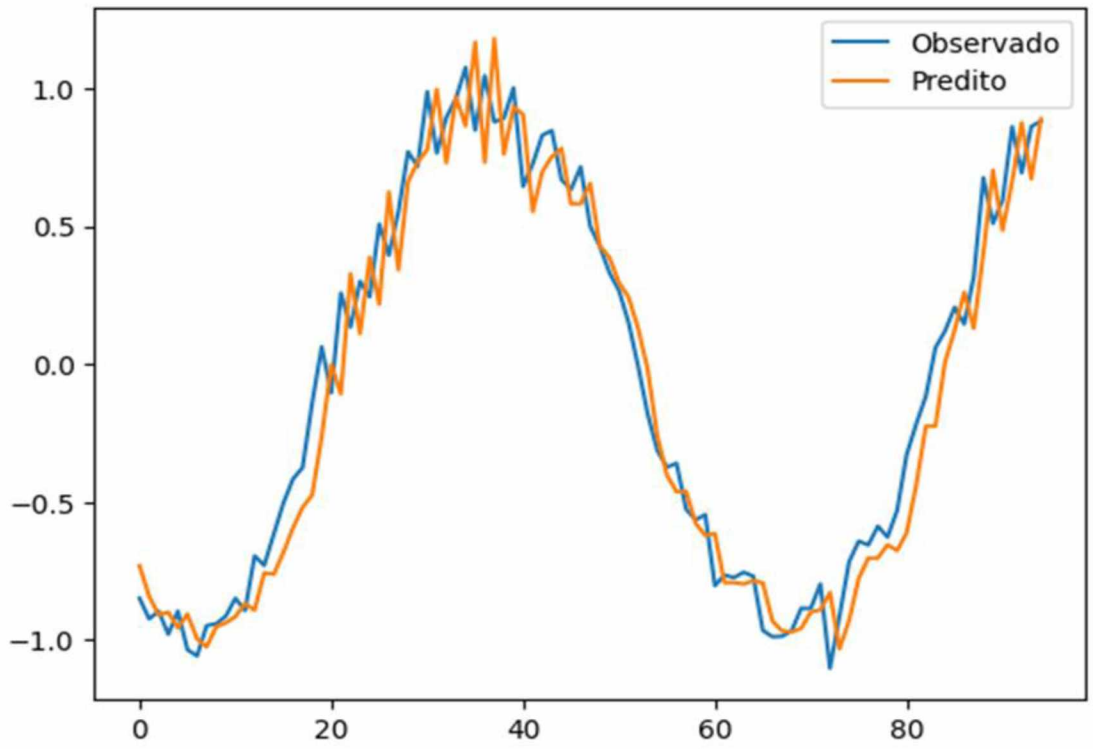

True label: cat Predicted: frog
0
5
10
15
20
25
30
0
5
10
15
20
25
30

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, SimpleRNN, Dense
from tensorflow.keras.optimizers import Adam

# --- 1. Geração e Preparação dos Dados ---

# Geração da série senoidal com ruído
series = np.sin(0.1 * np.arange(200)) + np.random.randn(200) * 0.1

T = 10  # tamanho da janela (Window size)
D = 1   # quantidade de dados/features (Number of features)
X = []
Y = []

# Criação das janelas de dados (X) e rótulos (Y)
for t in range(len(series) - T):
    x_window = series[t:t + T]
    X.append(x_window)
    y_next = series[t + T]
    Y.append(y_next)

# Reshape para o formato N x T x D (Amostras x Janela x Features)
X = np.array(X).reshape(-1, T, D)
Y = np.array(Y)
N = len(X)

# --- 2. Definição do Modelo SimpleRNN ---

input_layer = Input(shape=(T, D))
# SimpleRNN com 5 unidades ativadas por ReLU
x = SimpleRNN(5, activation="relu")(input_layer)
# Camada Densa de saída para regressão (1 unidade)
output_layer = Dense(1)(x)
model = Model(input_layer, output_layer)

# --- 3. Compilação e Treinamento ---

model.compile(
    loss="mse",
    optimizer=Adam(learning_rate=0.1)
)

# O conjunto de dados é dividido em treino (primeira metade) e validação
(segunda metade)
history = model.fit(
    X[:-N // 2],  # Primeira metade para treino
    Y[:-N // 2],
    epochs=80,
    validation_data=(X[-N // 2:], Y[-N // 2:])  # Segunda metade para
validação
)
```

```
# --- 4. Predição Autoregressiva (Loop de Previsão) ---

validation_target = Y[-N // 2:]
validation_predictions = []

# Inicia o índice na metade do dataset
i = -N // 2
while len(validation_predictions) < len(validation_target):
    # Prediz o próximo ponto usando a amostra atual (X[i])
    p = model.predict(X[i].reshape(1, -1, 1))[0, 0]
    i += 1
    validation_predictions.append(p)
```

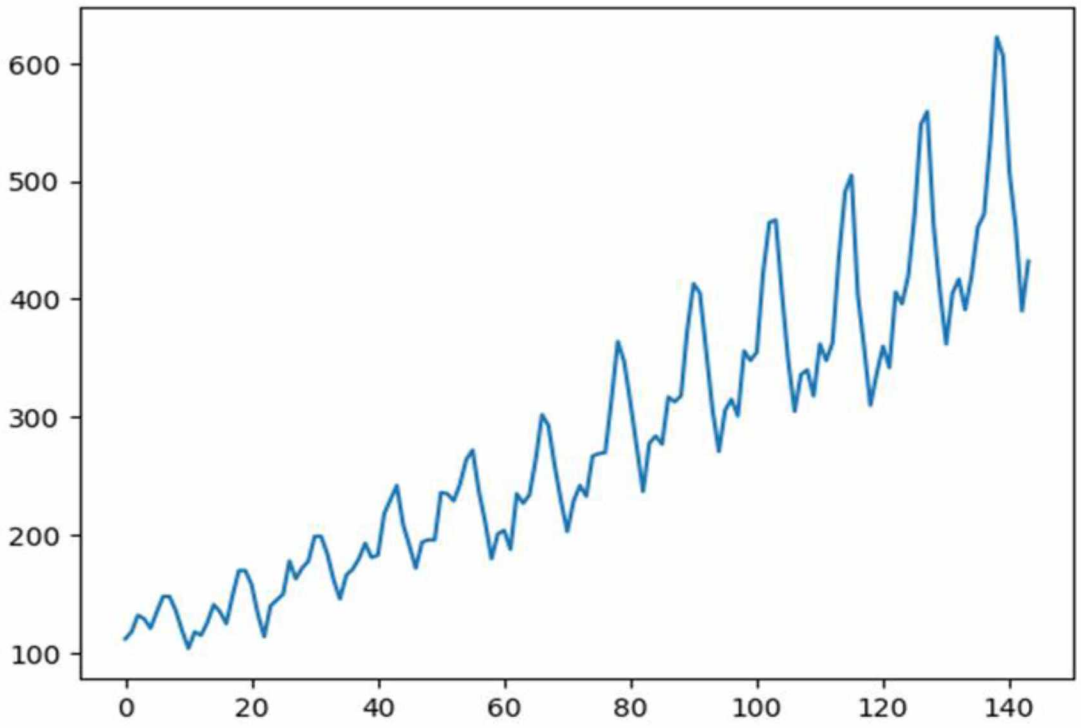


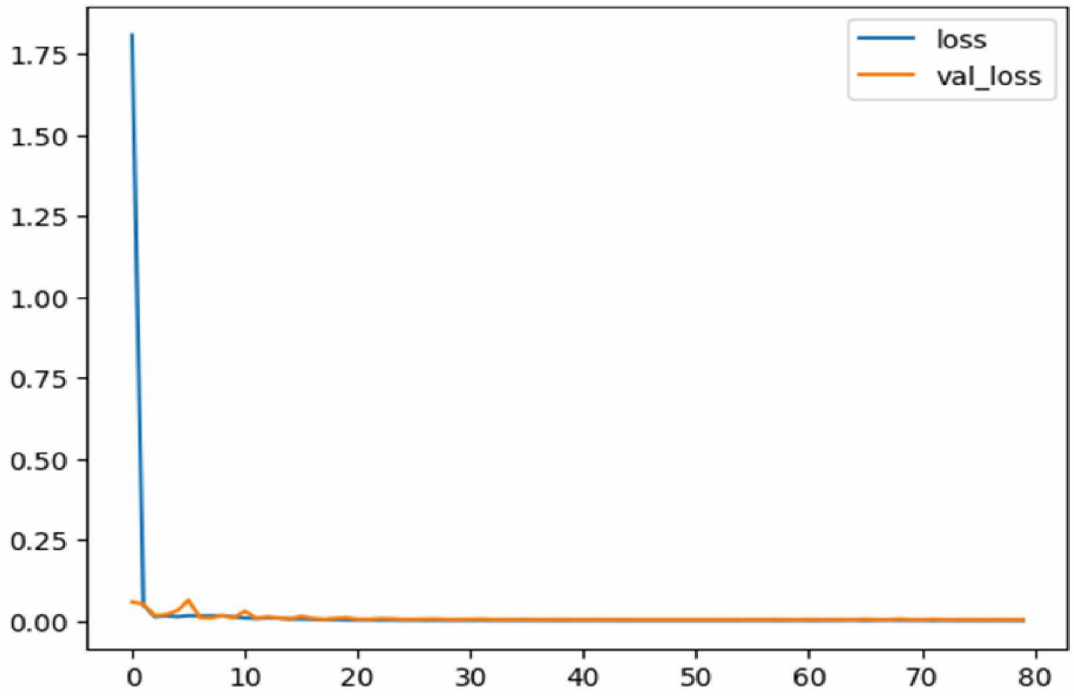





```
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, SimpleRNN, Dense
from tensorflow.keras.optimizers import Adam

# --- 1. Preparação e Pré-processamento dos Dados ---

# Download do dataset (assumindo que o wget foi executado)
# !wget http://www.razer.net.br/datasets/airline-passengers.csv

df = pd.read_csv("airline-passengers.csv", usecols=[1])
series = df.values
series = series.astype('float32')

# Escalonamento: Normaliza os dados para o intervalo [0, 1]
scaler = MinMaxScaler(feature_range=(0, 1))
series = scaler.fit_transform(series)

# Definição de hiperparâmetros
train_size = int(len(series) * 0.67)
T = 10  # Tamanho da janela (lookback)
D = 1   # Quantidade de features/dados (1 feature: número de passageiros)

# Criação das janelas de dados (X) e rótulos (Y)
X = []
Y = []
for t in range(len(series) - T):
    x_window = series[t:t + T]
```

```
    X.append(x_window)
    y_next = series[t + T]
    Y.append(y_next)

# Reshape para o formato N x T x D (Amostras x Janela x Features)
X = np.array(X).reshape(-1, T, D)
Y = np.array(Y)
N = len(X)

# --- 2. Definição do Modelo SimpleRNN ---

input_layer = Input(shape=(T, D))
# SimpleRNN com 5 unidades (ativação linear/None, conforme original)
x = SimpleRNN(5, activation=None)(input_layer)
# Camada Densa de saída para regressão (1 unidade)
output_layer = Dense(1)(x)
model = Model(input_layer, output_layer)

# --- 3. Compilação e Treinamento ---

model.compile(
    loss="mse",
    optimizer=Adam(learning_rate=0.1)
)

# O treinamento utiliza o primeiro 67% dos dados
history = model.fit(
    X[:train_size],
    Y[:train_size],
    epochs=80,
    validation_data=(X[-train_size:], Y[-train_size:]) # Usa a última parte
para validação
)

# --- 4. Predição Autoregressiva (Loop de Previsão) ---

# O loop autoregressivo é baseado na série completa, usando o tamanho do
treino como índice
validation_target = Y[-train_size:]
validation_predictions = []
i = -train_size
while len(validation_predictions) < len(validation_target):
    # Prediz o próximo ponto usando a amostra X[i]
    p = model.predict(X[i].reshape(1, -1, 1))[0, 0]
    i += 1
    validation_predictions.append(p)
2023-05-05 11:46:50 (159 MB/s) - ‘airline-passengers.csv’ saved [2180/2180]
```

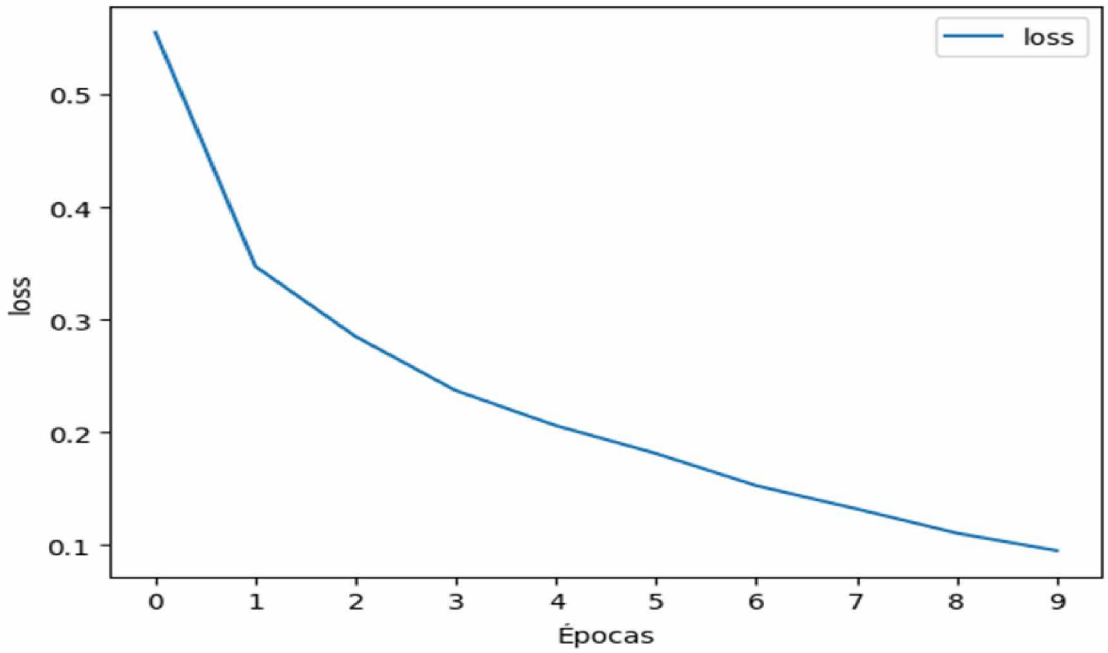


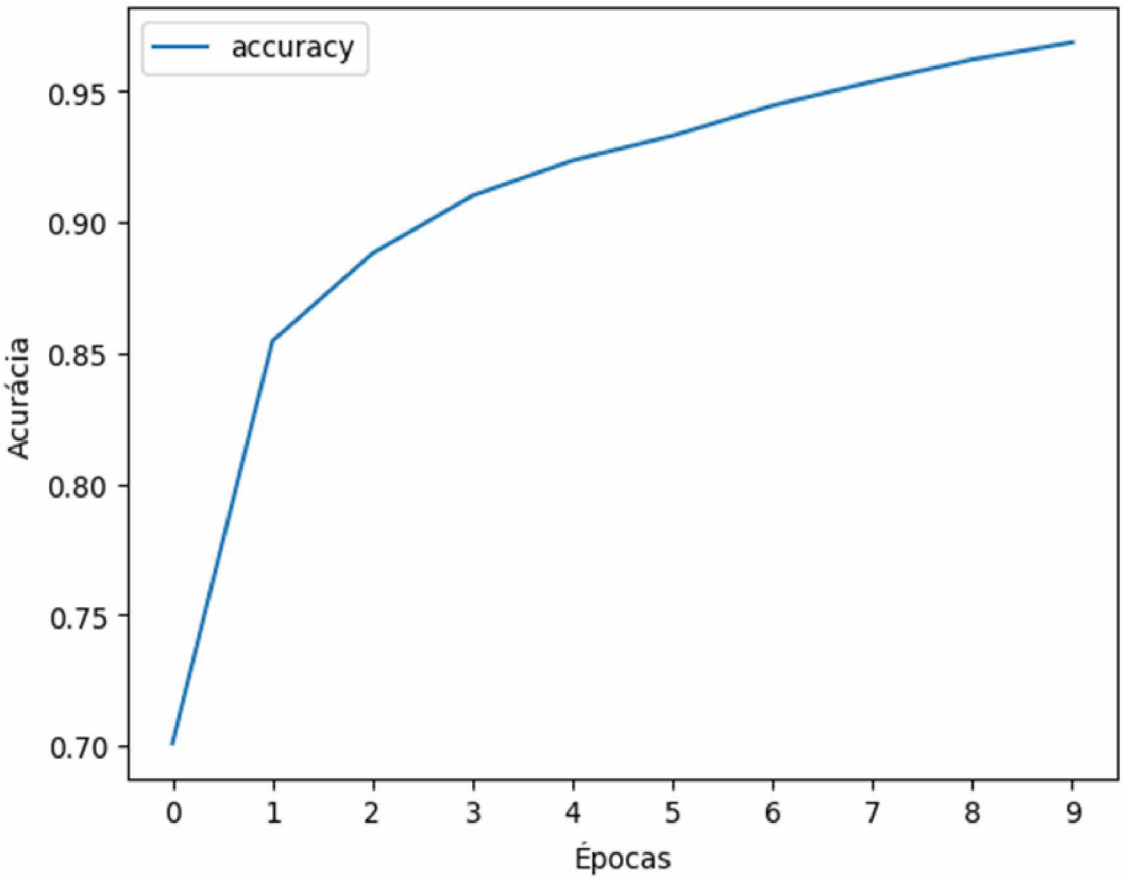



```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
from tensorflow.keras.optimizers import RMSprop

# --- Parâmetros ---
num_words = 20000  # número de palavras
maxlen = 200      # máximo palavras no review

# --- 1. Preparação dos Dados ---
```

```
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.imdb.load_data(num_words=num_words)

# --- 2. Definição do Modelo ---
# O shape de input é inferido pelo x_train.shape[1] (que deve ser o maxlen)
input_layer = Input(shape=(x_train.shape[1], ))

# Embedding Layer
x = Embedding(input_dim=num_words, output_dim=128)(input_layer)

# LSTM Layer
x = LSTM(units=128, activation="tanh")(x)

# Output Layer (Binary Classification)
output_layer = Dense(units=1, activation="sigmoid")(x)

model = Model(input_layer, output_layer)

# --- 3. Compilação e Treinamento ---
model.compile(
    optimizer="rmsprop",
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

# Imprimir o sumário do modelo
model.summary()

epochs = 10
history = model.fit(
    x_train,
    y_train,
    epochs=epochs,
    batch_size=128
)


Estrutura do Modelo - model_2

Camada (Layer) | Tipo | Forma de Saída (Output Shape) | Parâmetros (Param
#)
----------------|------|-------------------------------|-------------------
---
input_3         | InputLayer | (None, 200) | 0
embedding       | Embedding  | (None, 200, 128) | 2,560,000
lstm            | LSTM       | (None, 128) | 131,584
dense_2         | Dense      | (None, 1) | 129
----------------|------------|-------------------------------|-------------
---------
Resumo          | Valor
----------------|----------------------
Total de parâmetros | 2,691,713
Parâmetros treináveis | 2,691,713
```

```
Parâmetros não treináveis | 0
```

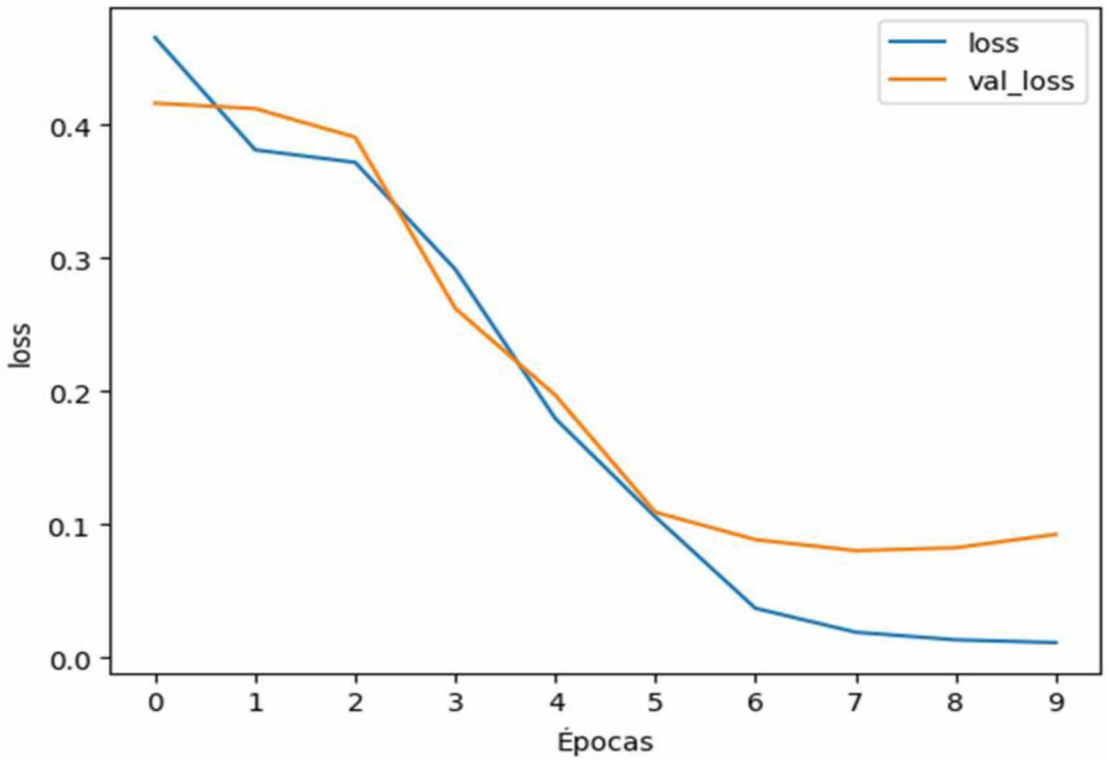


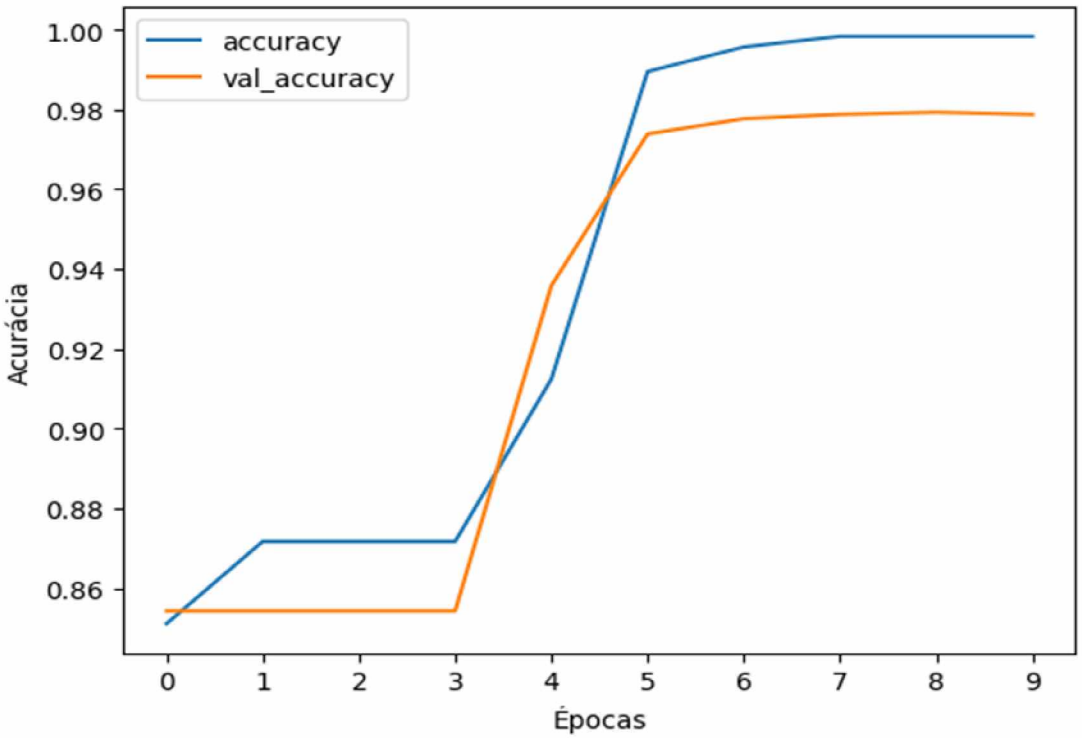



```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, LSTM,
GlobalMaxPooling1D, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

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

# --- 1. Preparação dos Dados ---

# Download do dataset
# !wget http://www.razer.net.br/datasets/spam.csv

df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values

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

# Tokenização e Padding
num_words = 20000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train)
sequences_test = tokenizer.texts_to_sequences(x_test)
word2index = tokenizer.word_index
V = len(word2index)

data_train = pad_sequences(sequences_train)
T = data_train.shape[1] # tamanho da sequência
data_test = pad_sequences(sequences_test, maxlen=T)

# --- 2. Definição do Modelo ---
D = 20 # tamanho do embedding
M = 15 # tamanho do hidden state

input_layer = Input(shape=(T,))
x = Embedding(V + 1, D)(input_layer)
x = LSTM(M, return_sequences=True)(x)
x = GlobalMaxPooling1D()(x)
output_layer = Dense(1, activation="sigmoid")(x)
model = Model(input_layer, output_layer)

# --- 3. Compilação e Treinamento ---
model.compile(
    loss="binary_crossentropy",
    optimizer="adam",
    metrics=["accuracy"]
)

epochs = 10
history = model.fit(
```

```
    data_train,
    y_train,
    epochs=epochs,
    validation_data=(data_test, y_test)
)

# --- 4. Predição de Exemplo ---
texto = "Before I introduce myself "
seq_texto = tokenizer.texts_to_sequences([texto])
data_texto = pad_sequences(seq_texto, maxlen=T)
pred = model.predict(data_texto)
782/782 [==============================] - 97s 123ms/step - loss: 0.4441 -
accuracy: 0.8646
0.8646000027656555

<matplotlib.legend.Legend at 0x7f5ef7dcb220>
```






```
import tensorflow as tf
import time
from tensorflow.keras.layers import StringLookup # Assumindo que esta
importação é necessária
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator #
Assumido para a seção Transfer Learning

# --- 1. Pipeline de Dados e Pré-processamento ---

# path_to_file e text loading (Mantido, mas assume-se que 'vocab' está
definido)
path_to_file = tf.keras.utils.get_file('shakespeare.txt',
'https://storage.googleapis.com/download.tensorflow.org/data/shakespeare.tx
t')
text = open(path_to_file, 'rb').read().decode(encoding='utf-8')

# Mapeamento de caracteres (ids_from_chars e chars_from_ids)
ids_from_chars = tf.keras.layers.StringLookup(vocabulary=list(vocab),
mask_token=None)
chars_from_ids =
tf.keras.keras.layers.StringLookup(vocabulary=ids_from_chars.get_vocabulary
(), invert=True, mask_token=None)

# Função auxiliar
def text_from_ids(ids):
    return tf.strings.reduce_join(chars_from_ids(ids), axis=-1)

# Criação do dataset de IDs
all_ids = ids_from_chars(tf.strings.unicode_split(text, 'UTF-8'))
ids_dataset = tf.data.Dataset.from_tensor_slices(all_ids)
```

```
# Criação de sequências
seq_length = 100
examples_per_epoch = len(text) // (seq_length + 1)
sequences = ids_dataset.batch(seq_length + 1, drop_remainder=True)

# Função para dividir input/target
def split_input_target(sequence):
    input_text = sequence[:-1]
    target_text = sequence[1:]
    return input_text, target_text

dataset = sequences.map(split_input_target)

# Otimização do pipeline
BATCH_SIZE = 64
BUFFER_SIZE = 10000
dataset = (
    dataset
    .shuffle(BUFFER_SIZE)
    .batch(BATCH_SIZE, drop_remainder=True)
    .prefetch(tf.data.experimental.AUTOTUNE)
)

# --- 2. Definição do Modelo Customizado (GRU) ---
# Assume-se que 'embedding_dim' e 'rnn_units' estão definidos.

class MyModel(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, rnn_units):
        super().__init__(self)
        self.embedding = tf.keras.layers.Embedding(vocab_size,
embedding_dim)
        self.gru = tf.keras.layers.GRU(rnn_units,
                                       return_sequences=True,
                                       return_state=True)
        self.dense = tf.keras.layers.Dense(vocab_size)

    def call(self, inputs, states=None, return_state=False,
training=False):
        x = inputs
        x = self.embedding(x, training=training)
        if states is None:
            states = self.gru.get_initial_state(x)
        x, states = self.gru(x, initial_state=states, training=training)
        x = self.dense(x, training=training)
        if return_state:
            return x, states
        else:
            return x

model = MyModel(
    vocab_size=len(ids_from_chars.get_vocabulary()),
    embedding_dim=embedding_dim,
```

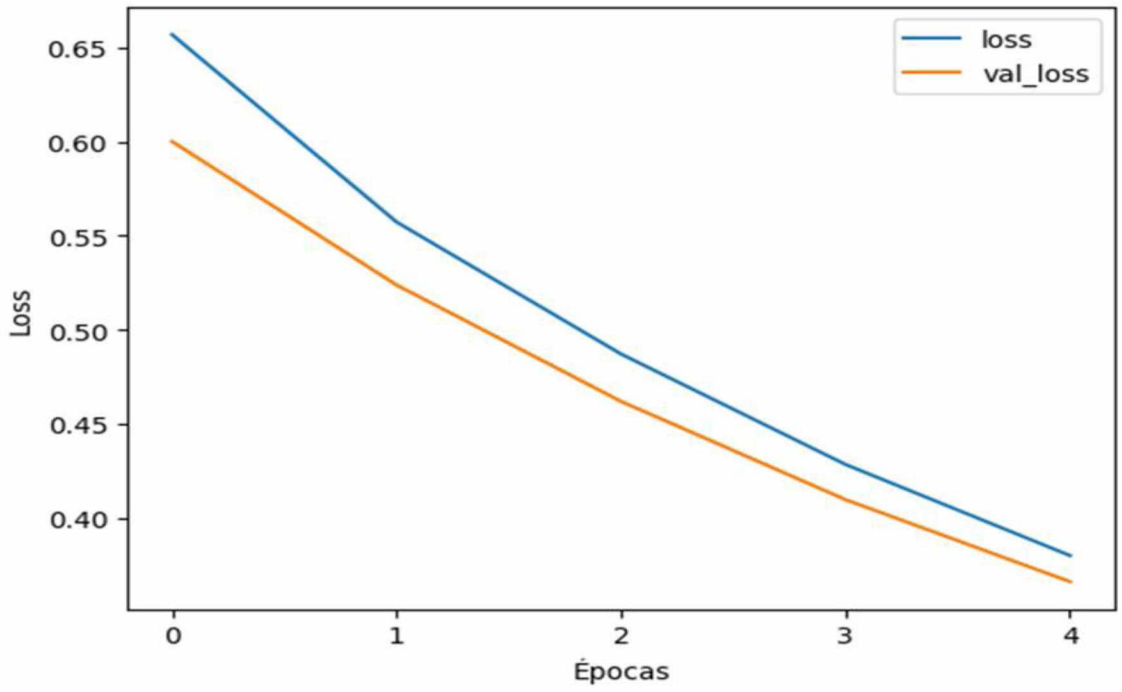
```
    rnn_units=rnn_units
)

# --- 3. Compilação e Treinamento ---
loss = tf.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(optimizer='adam', loss=loss)
EPOCHS = 20
history = model.fit(dataset, epochs=EPOCHS)

# --- 4. Geração de Texto em Uma Etapa (OneStep Model) ---

class OneStep(tf.keras.Model):
    def __init__(self, model, chars_from_ids, ids_from_chars,
temperature=1.0):
        super().__init__()
        self.temperature = temperature
        self.model = model
        self.chars_from_ids = chars_from_ids
        self.ids_from_chars = ids_from_chars
        # Cria uma máscara para prevenir a geração de [UNK]
        skip_ids = self.ids_from_chars(['[UNK]'])[:, None]
        sparse_mask = tf.SparseTensor(
            values=[-float('inf')] * len(skip_ids),
            indices=skip_ids,
            dense_shape=[len(ids_from_chars.get_vocabulary())])
        self.prediction_mask = tf.sparse.to_dense(sparse_mask)

    @tf.function
    def generate_one_step(self, inputs, states=None):
        input_chars = tf.strings.unicode_split(inputs, 'UTF-8')
        input_ids = self.ids_from_chars(input_chars).to_tensor()
        # predicted_logits.shape is [batch, char, next_char_logits]
        predicted_logits, states = self.model(inputs=input_ids,
states=states,
                                              return_state=True)
        predicted_logits = predicted_logits[:, -1, :]
        predicted_logits = predicted_logits / self.temperature
        predicted_logits = predicted_logits + self.prediction_mask
        predicted_ids = tf.random.categorical(predicted_logits,
num_samples=1)
        predicted_ids = tf.squeeze(predicted_ids, axis=-1)
        predicted_chars = self.chars_from_ids(predicted_ids)
        return predicted_chars, states

# Loop de Geração
one_step_model = OneStep(model, chars_from_ids, ids_from_chars)
start = time.time()
states = None
next_char = tf.constant(['ROMEO:'])
result = [next_char]
for n in range(1000):
    next_char, states = one_step_model.generate_one_step(next_char,
states=states)
```

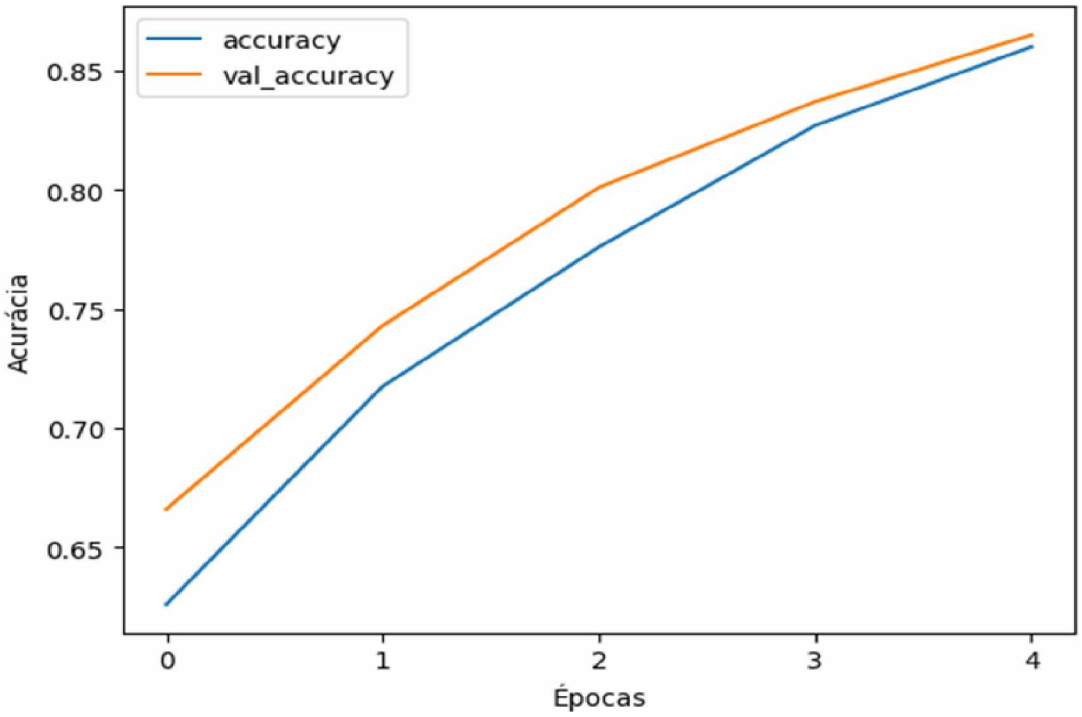
```
    result.append(next_char)

result = tf.strings.join(result)
end = time.time()

# --- 5. Transfer Learning (Cats and Dogs) ---
# Bloco de código para a seção 05 (Transfer Learning e Fine Tuning)

# Assume que o download foi feito: !wget
http://www.razer.net.br/datasets/cats_and_dogs_filtered.zip
# Assume que 'input_shape' e 'train_dir' / 'test_dir' estão definidos.

input_shape = (128, 128, 3) # tamanho da imagem entrada
base_model = tf.keras.applications.MobileNetV2(
    input_shape=input_shape,
    include_top=False,
    weights="imagenet")

base_model.trainable = False

global_average_layer =
tf.keras.layers.GlobalAveragePooling2D()(base_model.output)
prediction_layer = tf.keras.layers.Dense(units=1,
activation="sigmoid")(global_average_layer)
model = tf.keras.models.Model(base_model.input, prediction_layer)

model.summary()

# Compilação
model.compile(
    optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.0001),
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

# Data Generators
data_gen_train = ImageDataGenerator(rescale=1/255.)
data_gen_test = ImageDataGenerator(rescale=1/255.)

train_generator = data_gen_train.flow_from_directory(
    train_dir,
    target_size=(128, 128),
    batch_size=128,
    class_mode="binary")

test_generator = data_gen_train.flow_from_directory(
    test_dir,
    target_size=(128, 128),
    batch_size=128,
    class_mode="binary")

# Treinamento (Fase 1: Transfer Learning)
EPOCHS = 5
```

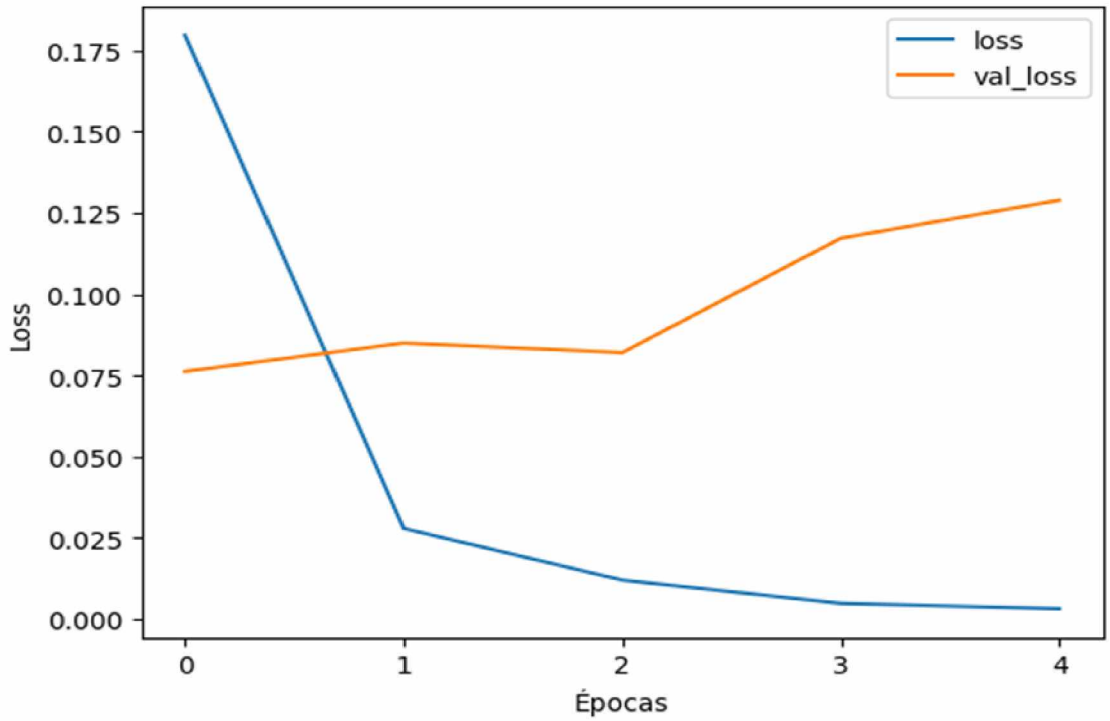
```
r_transfer = model.fit(train_generator, epochs=EPOCHS,
validation_data=test_generator)

# Compilação (Fase 2: Fine Tuning)
model.compile(
    optimizer=tf.keras.optimizers.RMSprop(learning_rate=0.0001),
    loss="binary_crossentropy",
    metrics=["accuracy"]
)

# Treinamento (Fase 2: Fine Tuning)
EPOCHS = 5
r_fine_tune = model.fit(train_generator, epochs=EPOCHS,
validation_data=test_generator)
Trecho Gerado
ROMEO:

The sweetest ladies, not it in the Tower.

Who thinks me groan? Dicl me in spleen.

On the gates, mark where she would not hence,

And cried: “Arcalus! Where are thy tubbury?

Who selves?”


ANGELO:

And she shall. Let me, for truth, me from him —

Merdy as prisoners, and so brief a good.

Yet knows no levy head befaly as e’er;

I dare not wonth even now, while id the ropegor,

To louser than continue in your lights.


MONTAGUE:

And this way to the gods.

Mark’d you this afflict of this? Look, and

Three there, I should live on her maid is age.

Of all tongues from the world can corn hath made you.

What says he of Aury God? Why, then, all true penjery.
Thou shalt command; then, would the word it it.
```

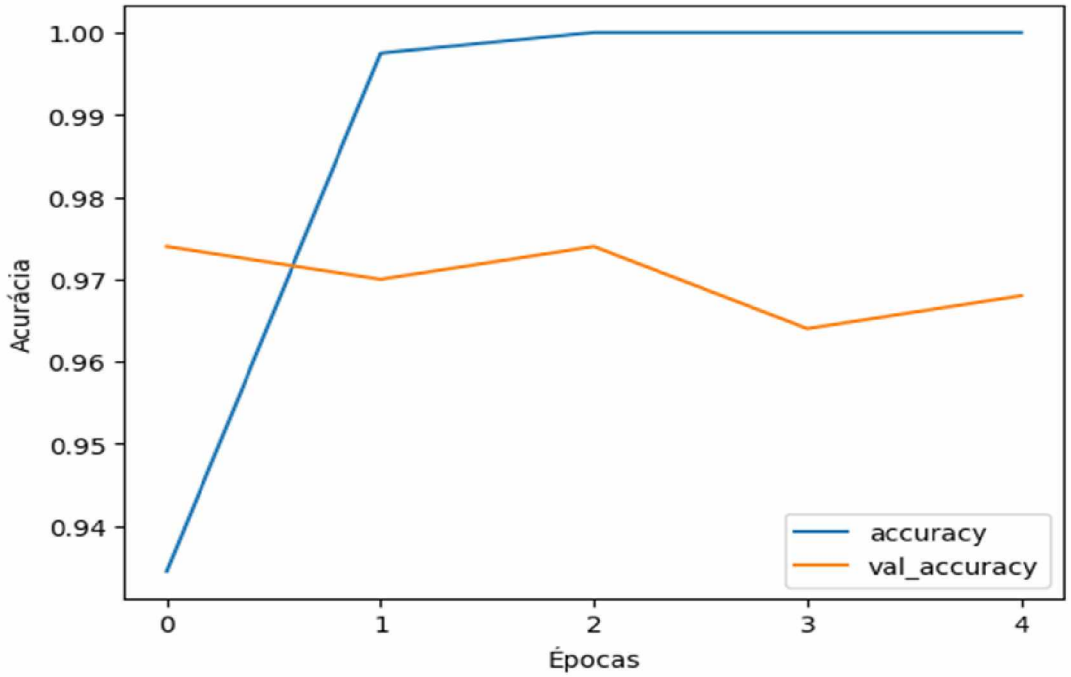
```
DUKE OF AUMERLE:

I know not, meavers.


MENENIUS:

Do you think it be? Thou wert know.

Did not my life but kill ’em? But I am dull to him.

And freely he was many one, how to

the Lady Bona. Gentlemen,

Look — signify a crupper, that thou shouldst have

The time that Rome Hastings all overpect post.
Thy words will ever exec.
```

Run time: 4.660545349121094



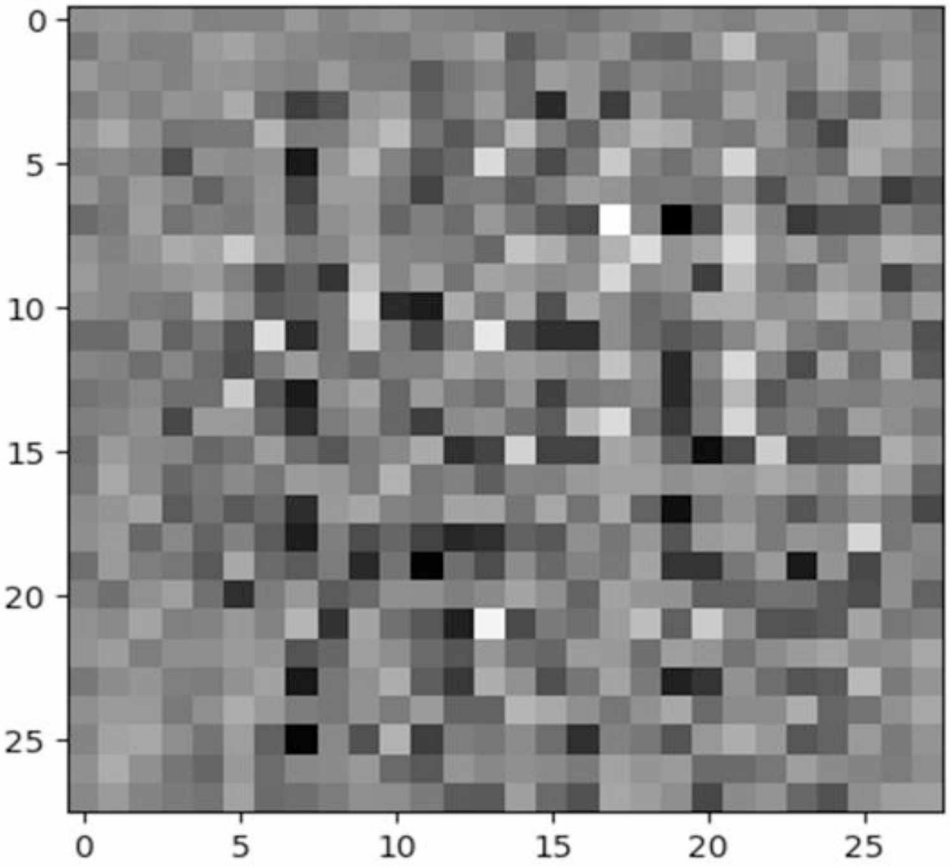

accuracy
val_accuracy
0.85
0.80
0.75
0.70
0.65
Acurácia
0
1
2
3
4
Épocas

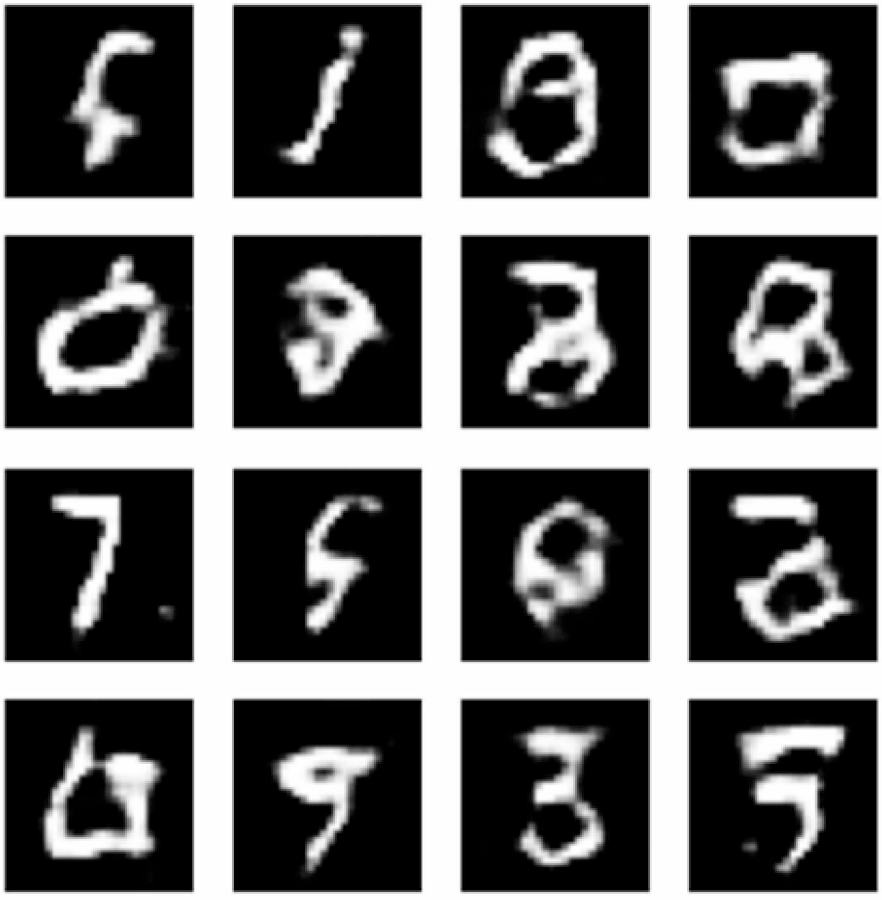

loss
val_loss
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Loss
0
1
2
3
4
Épocas




```
import tensorflow as tf
import time
# Assumindo que matplotlib.pyplot está importado como plt e display está
importado
# from IPython import display

# --- 1. Função de Treinamento por Etapa (train_step) ---

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

    # Usa tf.GradientTape para calcular gradientes para o Gerador e
Discriminador
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        # Gerador cria imagens falsas
        generated_images = generator(noise, training=True)

        # Discriminador avalia as imagens reais e falsas
        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)

        # Cálculo das perdas (losses)
        gen_loss = generator_loss(fake_output)
        disc_loss = discriminator_loss(real_output, fake_output)

    # 1. Calcula e aplica gradientes para o Gerador
    gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
    generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))
```

```
    # 2. Calcula e aplica gradientes para o Discriminador
    gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
discriminator.trainable_variables))

# --- 2. Geração e Salvamento de Imagens (generate_and_save_images) ---

def generate_and_save_images(model, epoch, test_input):
    # Faz previsões sem treinamento
    predictions = model(test_input, training=False)

    # Configuração da figura (assumindo que matplotlib está importado)
    fig = plt.figure(figsize=(4, 4))
    for i in range(predictions.shape[0]):
        plt.subplot(4, 4, i + 1)
        # Normaliza a saída da GAN para exibição (assumindo [-1, 1] ou [0,
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()

# --- 3. Função Principal de Treinamento (train) ---

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

        # Itera sobre os lotes de imagens
        for image_batch in dataset:
            train_step(image_batch)

        # Limpa a saída e salva/mostra o progresso
        display.clear_output(wait=True)
        generate_and_save_images(generator, epoch + 1, seed)

        # Salva checkpoint a cada 15 épocas
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)

    # Limpa a saída e salva/mostra o resultado final
    display.clear_output(wait=True)
    generate_and_save_images(generator, epochs, seed)

# --- 4. Execução do Treinamento ---
# Assume que train_dataset e EPOCHS estão definidos
# train(train_dataset, EPOCHS)
# checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))
```

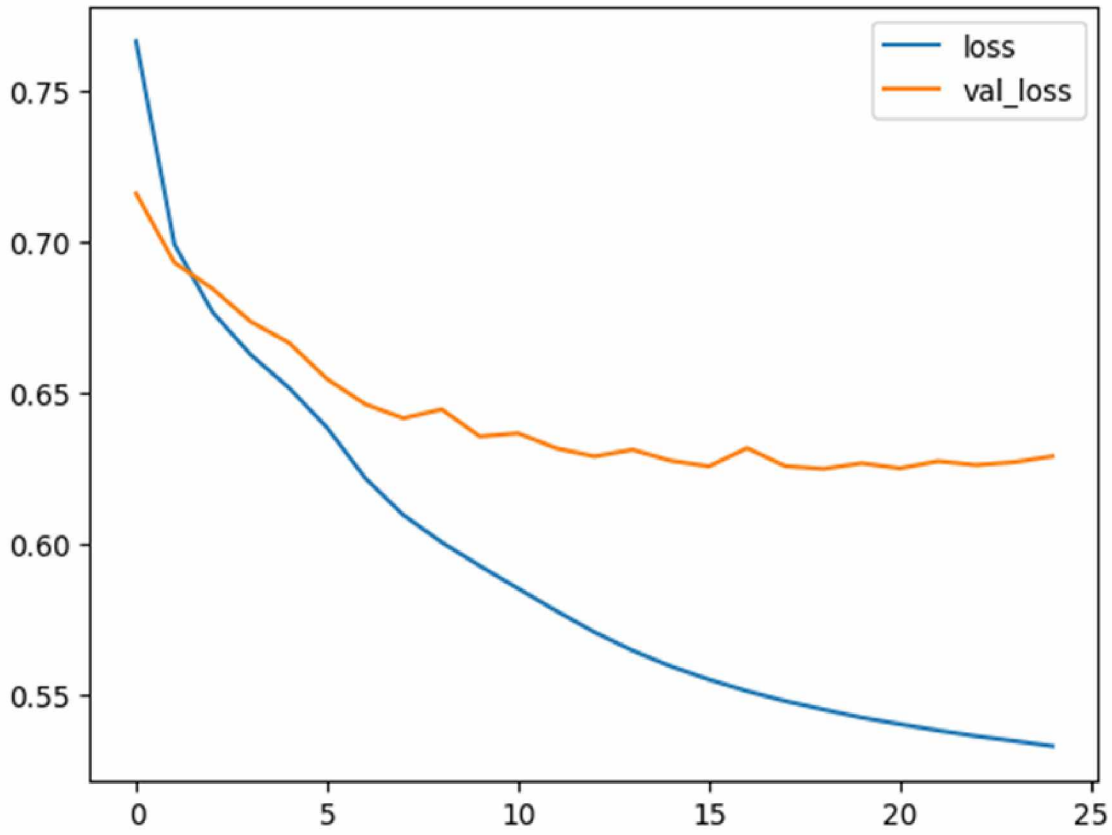

0
5
10
15
20
25
0
5
10
15
20
25



```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Flatten, Concatenate,
Dense
from tensorflow.keras.optimizers import SGD
import numpy as np

# --- 1. Definição do Modelo de Fatores Latentes ---
# Assumindo que N (num_users), M (num_movies) e K (latent_dimension) estão
definidos
# e que Input, Embedding, Flatten, Concatenate, Dense, Model estão
importados.

# Input para o usuário (u)
input_user = Input(shape=(1,))
user_embedding = Embedding(N, K)(input_user) # Saída: num_samples, 1, K
user_embedding = Flatten()(user_embedding)  # Saída: num_samples, K

# Input para o filme (m)
input_movie = Input(shape=(1,))
movie_embedding = Embedding(M, K)(input_movie)  # Saída: num_samples, 1, K
movie_embedding = Flatten()(movie_embedding)    # Saída: num_samples, K

# Combinação dos fatores latentes
x = Concatenate()([user_embedding, movie_embedding])

# Camadas densas para processar a combinação
x = Dense(1024, activation="relu")(x)
output_rating = Dense(1)(x)

model = Model(inputs=[input_user, input_movie], outputs=output_rating)

# --- 2. Compilação e Pré-processamento ---

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

# Centralização das notas (ratings)
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating

# --- 3. Treinamento ---
epochs = 25
# Variável 'r' para o histórico do treinamento
history = model.fit(
    x=[train_user, train_movie],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso por lote
```

```
    validation_data=([test_user, test_movie], test_ratings)
)

# --- 4. Predição (Exemplo) ---
# Assumindo que M (num_movies) e movie_ids estão definidos.
input_usuario = np.repeat(a=73023, repeats=M)
film = np.array(list(set(movie_ids)))

# Predição
preds = model.predict([input_usuario, film])

# Reconversão das notas para a escala original
rat = preds.flatten() + avg_rating
idx = np.argmax(rat)
```

ID	userId	movieId	rating	timestamp
0	1	2	3.5	1112486027
1	1	29	3.5	1112484676
2	1	32	3.5	1112484819
3	1	47	3.5	1112484727
4	1	50	3.5	1112484580


loss
val_loss
0.75
0.70
0.65
0.60
0.55
0
5
10
15
20
25

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

1. Dataset Para este trabalho, foi escolhido um conjunto de dados climático disponível no site da NOAA (National Centers for Environmental Information). Esse conjunto de dados representa a anomalia da temperatura global de 1850 a 2025. A fonte original dos dados pode ser consultada no seguinte link: NOAA Climate Data. A ideia para esse cálculo foi um gráfico do portal G1 comparando várias visualizações sobre a mudança climática.

2. Contexto e Público-Alvo

- **Contexto**
    - As mudanças climáticas foram um dos tópicos centrais nos discursos de sustentabilidade, políticas ambientais e economia global.
    - A anomalia da temperatura global é a expressão do efeito da atividade humana sobre o clima planetário.
    - O presente trabalho tem como objetivo formar uma visão mais nítida e imponente dessa tendência, descrevendo a evolução das anomalias térmicas desde o século XIX.

- **Público-Alvo**
  - Este trabalho se destina a um público amplo.
  - Inclui pesquisadores e cientistas climáticos, que podem se beneficiar de uma representação visual mais direta das tendências.
  - Também se destina a políticos e tomadores de decisão, que precisam de dados visuais acessíveis para formulação de políticas ambientais.
  - Educadores e estudantes também são público-alvo, podendo usar a análise para entender melhor as mudanças climáticas.
  - O público em geral, interessado em aprender mais sobre as tendências globais de temperatura e como elas impactam suas vidas, é outro grupo-alvo.

4. **Visualização de Dados** Para criar a visualização dos dados, foram utilizadas as bibliotecas Python Matplotlib, Pandas e Seaborn. O código utilizado para gerar o gráfico está descrito no documento.

```
Import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

8 Caminho do arquivo CSV

```
file_path = '/content/data.csv' # Atualize para o caminho correto
```

9 Carregar os dados a partir da Linha 5

```
data = pd.read_csv(file_path, encoding='utf-8', skiprows=4, sep=',')
```

10 Renomear colunas

```
data.columns = ['Date', 'Anomaly']
```

11 Converter a coluna Date para ano e mês

```
data['Ano'] = data['Date'] // 100
data['Mês'] = data['Date'] % 100
```

12 Criar coluna de tempo decimal

```
data['Ano_decimal'] = data['Ano'] + (data['Mês'] - 1) / 12
```

13 Separar dados acima e abaixo de zero

```
acima_zero = data[data['Anomaly'] > 0]
abaixo_zero = data[data['Anomaly'] <= 0]
```

14 Criar o gráfico

```
plt.figure(figsize=(14,                                                    7))
plt.scatter(abaixo_zero['Ano_decimal'],              abaixo_zero['Anomaly'],
color='royalblue',                        label='≤                         0°C')
plt.scatter(acima_zero['Ano_decimal'],               acima_zero['Anomaly'],
color='crimson', label='> 0°C')
```

15 Linha de 0°C

```
plt.axhline(0, color='black', linestyle='--', linewidth=1, label='Linha de
0°C')
```

16 Adicionar linha de tendência

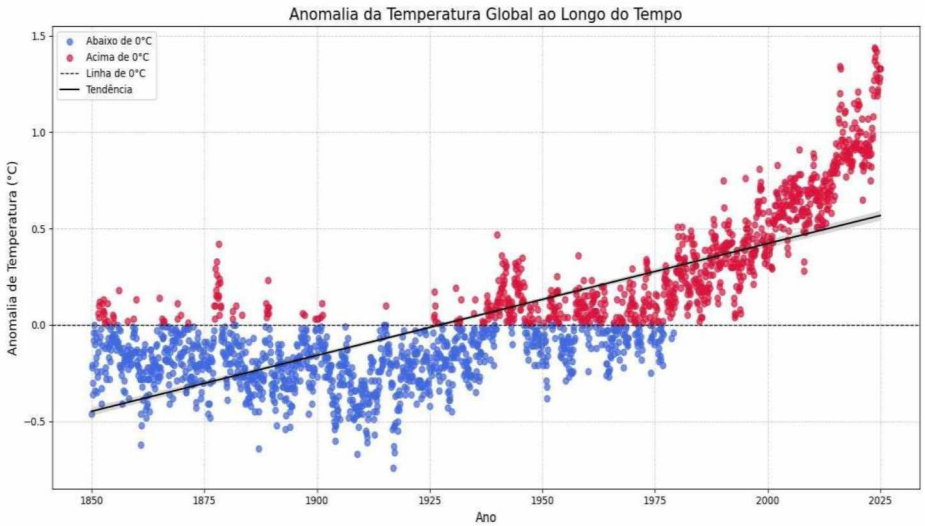
```
sns.regplot(x='Ano_decimal',    y='Anomaly',    data=data,    scatter=False,
color='black')
```

17 Configurações do gráfico

```
plt.title('Anomalia da Temperatura Global ao Longo do Tempo', fontsize=16)
plt.xlabel('Ano',                                                fontsize=12)
plt.ylabel('Anomalia      de      Temperatura      (°C)',      fontsize=12)
plt.legend()
plt.grid(True,                  linestyle='--',                  alpha=0.6)
plt.tight_layout()
```

18 Mostrar o gráfico

```
plt.show()
```



- **Escolha do Tipo de Visualização e Ferramenta**

- O gráfico de dispersão foi escolhido para mostrar cada ponto de anomalia ao longo dos anos, facilitando a percepção de tendências.
- Uma linha de tendência foi adicionada para evidenciar o aumento global das temperaturas.
- Seaborn e Matplotlib foram utilizados devido à sua flexibilidade e capacidade de criar visualizações detalhadas.

4. Storytelling da Visualização de Dados A história que essa visualização conta é clara: a temperatura global tem aumentado significativamente nos últimos 170 anos. No início da série histórica, a maioria dos pontos está abaixo da linha de 0°C (representando temperaturas mais frias do que a média de referência). Conforme se avança no tempo, a quantidade de pontos vermelhos (anomalias positivas) aumenta exponencialmente, principalmente a partir do século XXI. Esse gráfico evidencia que o aquecimento global é real e acelerado, sendo influenciado por fatores humanos como a industrialização e a emissão de gases do efeito estufa. A linha de tendência adicionada ajuda a destacar essa progressão.

- **Possíveis Ações Baseadas nos Dados**
  - A análise dos dados sugere a necessidade de medidas urgentes para mitigar o impacto das mudanças climáticas.
  - Isso inclui a redução da emissão de gases do efeito estufa e o investimento em energias renováveis.
  - Também são sugeridas políticas ambientais mais rigorosas e a educação da população sobre sustentabilidade.
  - Este trabalho busca contribuir para uma maior compreensão do problema e incentivar soluções concretas.
  - A ciência dos dados pode ser uma grande aliada na luta contra as mudanças climáticas.

APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

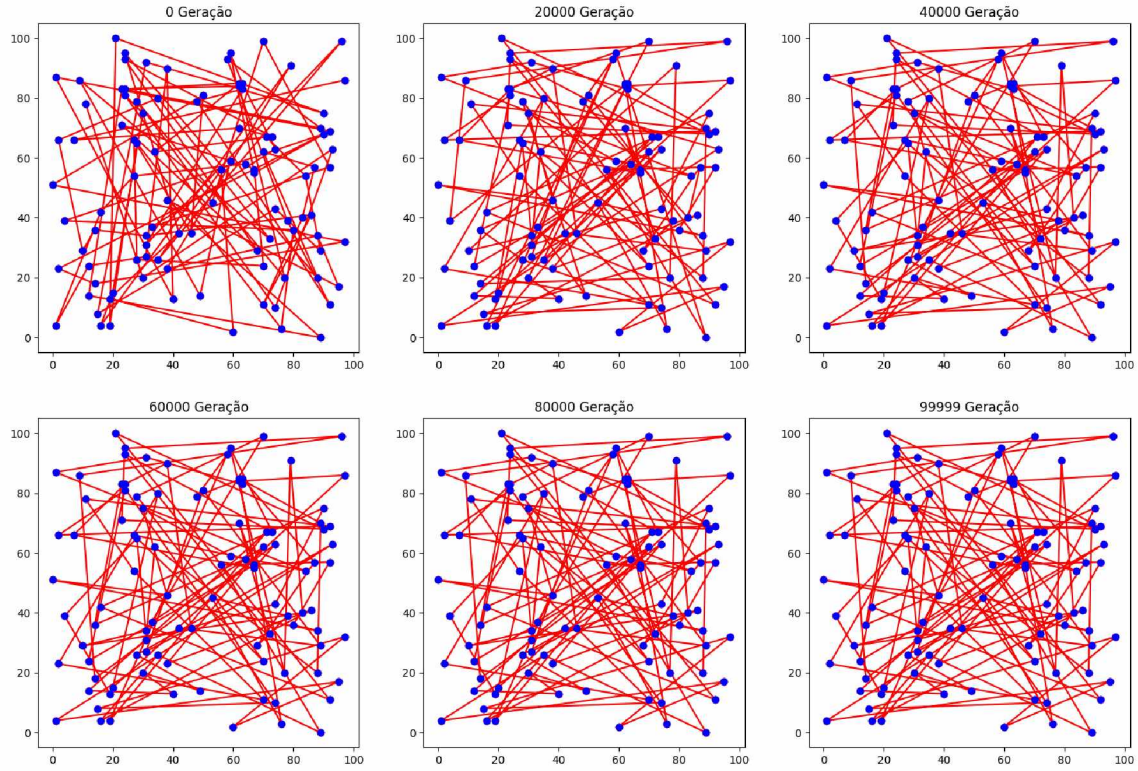
O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

Algoritmo Genético, Problema do Caixeiro Viajante

```
import random
import matplotlib.pyplot as plt

# --- Inicialização ---
# Gera uma população inicial aleatória
population = [random.sample(range(num_cities), num_cities) for i in
range(population_size)]
title = []
images = []

# --- Loop de Gerações ---
for generation in range(num_generations):
    # Seleção dos melhores indivíduos
    selected = selection(population)
    new_population = []

    # Crossover (Cruzamento)
    while len(new_population) < population_size - len(selected):
        parent1, parent2 = random.sample(selected, 2)

        if random.random() < crossover_rate:
            child1, child2 = crossover(parent1, parent2)
            new_population.append(child1)
            new_population.append(child2)
```

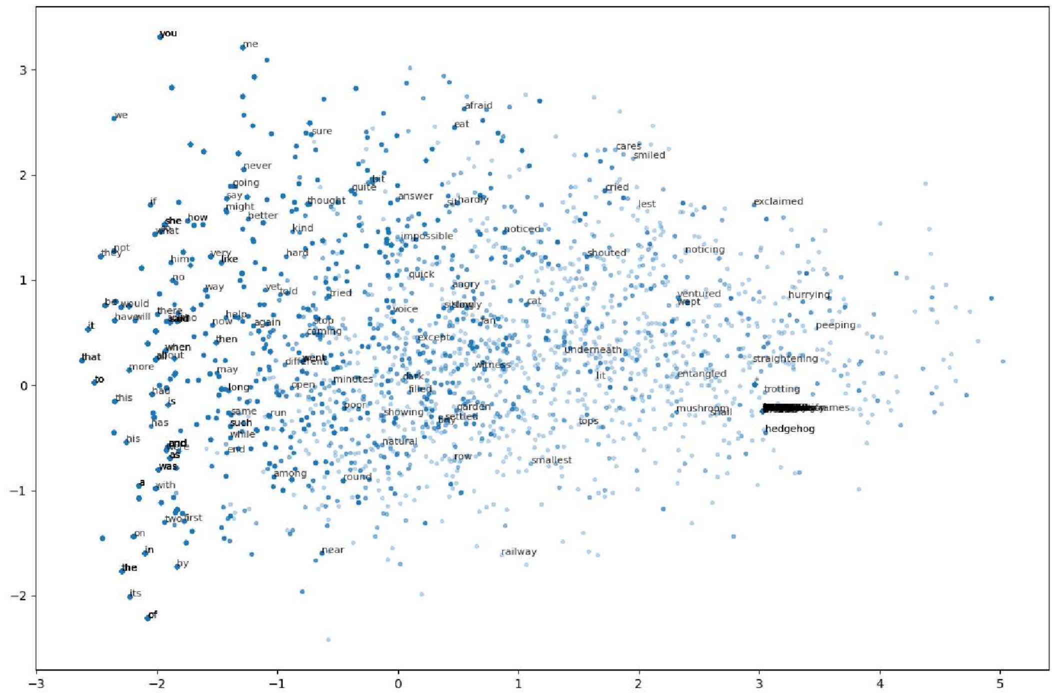
```
            else:
                new_population.append(parent1)
                new_population.append(parent2)

        # Adiciona os selecionados (elitismo) à nova população
        new_population += selected

        # Mutação
        for i in range(population_size):
            if random.random() < mutation_rate:
                new_population[i] = mutation(new_population[i])

        # Atualiza a população
        population = new_population
        best_distance = total_distance(selected[0])

        # Coleta dados para visualização (a cada 20000 gerações ou na última)
        if generation % 20000 == 0 or generation + 1 == num_generations:
            best_route = selected[0]
            # Assume que 'cities' é uma lista de coordenadas [(x, y), ...]
            route_coordinates = [cities[city-1] for city in best_route]
            x = [coord[0] for coord in route_coordinates]
            y = [coord[1] for coord in route_coordinates]

            img = (x, y)
            title.append(str(generation) + " Geração")
            images.append(img)

# --- Visualização dos Resultados ---
fig = plt.gcf()
fig.set_size_inches(18, 12)

# Plota até 6 gráficos da evolução
for i in range(len(images)):
    if i >= 6: break # Limita a 6 subplots (grid 2x3)
    plt.subplot(2, 3, i + 1)
    x, y = images[i]
    plt.plot(x, y, 'ro-') # Linhas vermelhas conectando pontos
    plt.plot(x, y, 'bo')  # Pontos azuis para as cidades
    plt.title(title[i])

plt.show()

# Melhor rota final
best_route = selected[0]
```



2. Representação de Texto usando PCA (GloVe)

```
import random
import matplotlib.pyplot as plt

# --- Parâmetros e Inicialização ---
# (Assumindo que num_cities, population_size, num_generations, etc., estão
definidos)

population = [random.sample(range(num_cities), num_cities) for i in
range(population_size)]
title = []
images = []

# --- Loop de Gerações ---
for generation in range(num_generations):
    # Seleção
    selected = selection(population)
    new_population = []

    # Crossover
    while len(new_population) < population_size - len(selected):
        parent1, parent2 = random.sample(selected, 2)

        if random.random() < crossover_rate:
            child1, child2 = crossover(parent1, parent2)
            new_population.append(child1)
            new_population.append(child2)
        else:
            new_population.append(parent1)
            new_population.append(parent2)
```

```
    # Elitismo (mantém os selecionados)
    new_population += selected

    # Mutação
    for i in range(population_size):
        if random.random() < mutation_rate:
            new_population[i] = mutation(new_population[i])

    population = new_population
    best_distance = total_distance(selected[0])

    # Coleta de dados para visualização
    if generation % 20000 == 0 or generation + 1 == num_generations:
        best_route = selected[0]
        # Assumindo que 'cities' é uma lista de coordenadas
        route_coordinates = [cities[city - 1] for city in best_route]
        x = [coord[0] for coord in route_coordinates]
        y = [coord[1] for coord in route_coordinates]

        img = (x, y)
        title.append(str(generation) + " Geração")
        images.append(img)

# --- Visualização ---
fig = plt.gcf()
fig.set_size_inches(18, 12)

for i in range(len(images)):
    # Limita a subplots se houver muitas imagens
    if i >= 6: break
    plt.subplot(2, 3, i + 1)
    x, y = images[i]
    plt.plot(x, y, 'ro-')
    plt.plot(x, y, 'bo')
    plt.title(title[i])

plt.show()

best_route = selected[0]
```


you
me
we
afraid
eat
sure
cares
smiled
never
going
cried
jest
exclaimed
thought
answer
hardly
might
better
kind
noticed
impossible
noticing
shouted
hard
quick
angry
ventured
hurrying
peeping
tried
voice
again
then
underneath
straightening
entangled
trotting
mushroom
hedgehog
garden
natural
smallest
near
railway
among
round
the
its
of
in
on
by
with
was
and
his
this
has
that
to
it
more
when
such
minutes
filled
tops
row
−3
−2
−1
0
1
2
3
4
5