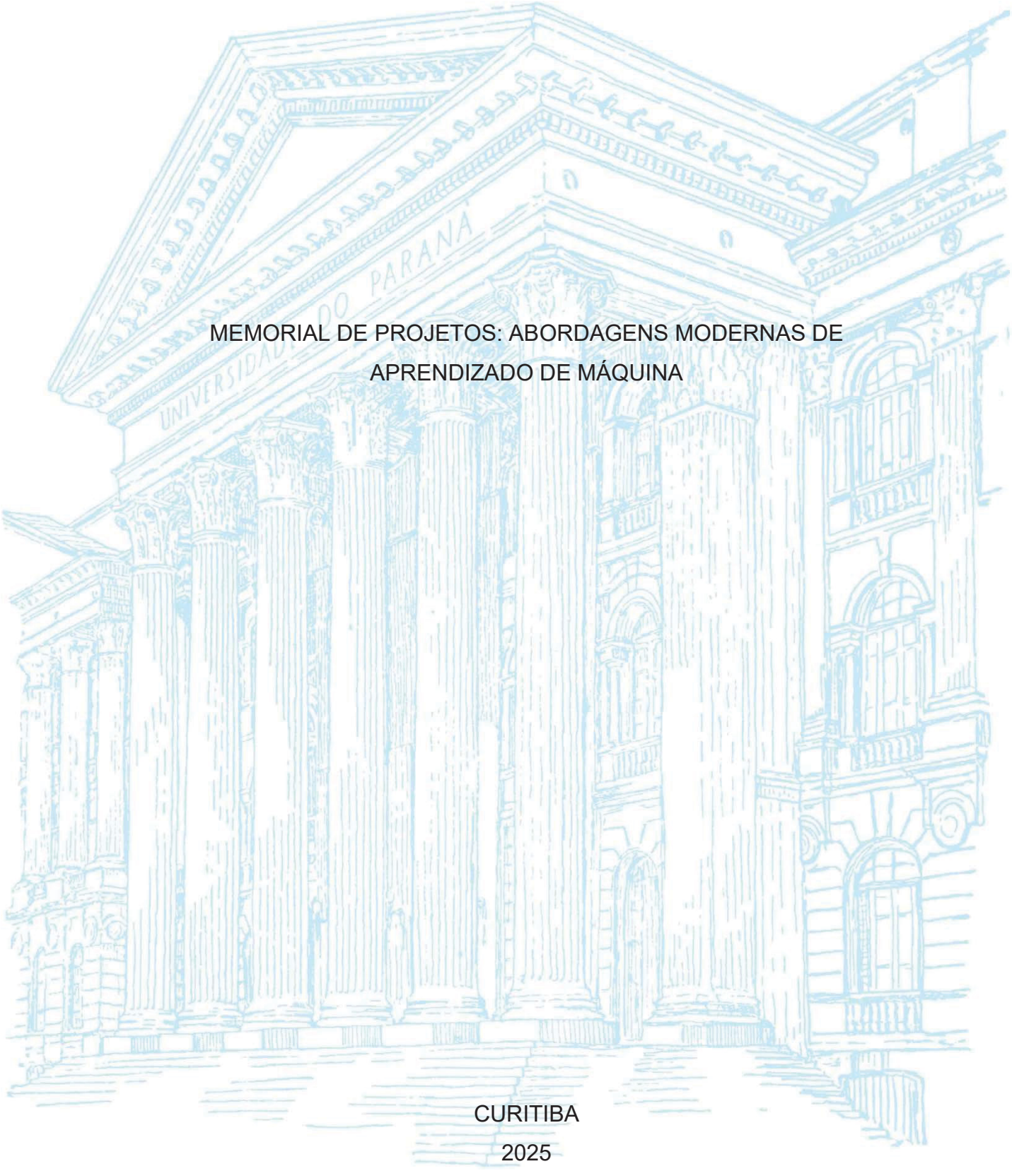
UNIVERSIDADE FEDERAL DO PARANÁ

WENDLY VIEIRA CUSTER



MEMORIAL DE PROJETOS: ABORDAGENS MODERNAS DE APRENDIZADO DE MÁQUINA

CURITIBA

2025

WENDLY VIEIRA CUSTER

MEMORIAL DE PROJETOS: ABORDAGENS MODERNAS DE APRENDIZADO DE MÁQUINA

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

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 **WENDLY VIEIRA CUSTER**, intitulada: **MEMORIAL DE PROJETOS: ABORDAGENS MODERNAS DE APRENDIZADO DE MÁQUINA**, 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, 16 de Outubro de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este trabalho tem como objetivo apresentar os fundamentos do Aprendizado de Máquina, discorrendo sobre cada um dos métodos clássicos, fazendo uma breve descrição das suas características e elencando algumas das principais técnicas usadas. São abordadas as possibilidades e as contribuições que esses métodos trouxeram para a Inteligência Artificial, mas principalmente as limitações que foram identificadas na aplicação destas técnicas em problemas modernos, principalmente quando envolvem dimensionalidade dos dados, processamento de linguagem natural e visão computacional. Considerando este contexto e a ampla adoção de ferramentas que usam o aprendizado de máquina, constatou-se que nem todas as barreiras poderiam ser superadas apenas com a aplicação das técnicas existentes, portanto tornou-se necessário o desenvolvimento de técnicas mais inovadoras, que poderiam explorar melhor o grande poder computacional contemporâneo e produzir resultados superiores ou mesmo inviáveis em relação aos métodos clássicos. Este trabalho também apresenta algumas das principais técnicas modernas desenvolvidas (*Automl*, Aprendizado Federado e *Deep Learning*), apontando como é o funcionamento de cada uma delas e especificando áreas em que elas podem ser aplicadas. A conclusão destaca a relevância da Inteligência Artificial atualmente e reforça a importância da evolução das técnicas de Aprendizado de Máquina.

Palavras-chave: inteligência artificial; aprendizado de máquina; a*utoml*; aprendizado profundo; aprendizado federado.

ABSTRACT

This work aims to present the fundamentals of Machine Learning, discussing each of the classical methods, briefly describing their characteristics, and listing some of the main techniques used. It addresses the possibilities and contributions that these methods have brought to Artificial Intelligence, but mainly the limitations that have been identified in the application of these techniques to modern problems, especially when they involve data dimensionality, natural language processing, and computer vision. Considering this context and the widespread adoption of tools that use machine learning, it was found that not all barriers could be overcome solely by applying existing techniques. Therefore, it became necessary to develop more innovative techniques that could better exploit contemporary computing power and produce superior or even unattainable results compared to classical methods. This work also presents some of the main modern techniques developed (Automl, Federated Learning, and Deep Learning), explaining how each one works and specifying areas in which they can be applied. The conclusion highlights the relevance of Artificial Intelligence today and reinforces the importance of the evolution of Machine Learning techniques.

Keywords: artificial intelligence; machine learning; automl; deep learning; federated learning.

SUMÁRIO

1 PARECER TÉCNICO

Nos últimos anos, a disponibilidade de grandes volumes de dados nas diversas áreas, aliada a maior capacidade computacional tem trazido maiores possibilidades de estudo, pesquisa e consequentemente o desenvolvimento de novos conceitos ou soluções computacionais, porém mesmo com o grande volume de dados, ainda existem obstáculos a serem superados, pois o desenvolvimento de *software* tradicional não poderia lidar com a complexidade dos dados, visto que habitualmente o desenvolvimento é feito com a concepção de entradas esperadas. Neste contexto, a Inteligência Artificial, mais precisamente o Aprendizado de Máquina (uma subárea da Inteligência Artificial) aparece como uma abordagem apropriada para enfrentar esses obstáculos.

Segundo Mitchel (1997), o Aprendizado de Máquina consiste na habilidade do computador aprender sem ser programado explicitamente, portanto pode-se afirmar que a tarefa de compreender a complexidade dos dados será feita diretamente pelo computador e por isso o Aprendizado de Máquina vem se tornando um dos pilares das abordagens modernas que usam Inteligência Artificial, visto que as técnicas desta área permitem que o computador adquira conhecimento por meio de exemplos.

Ludermir (2021) afirma que o Aprendizado de Máquina pode ser dividido em três tipos clássicos: supervisionado, não supervisionado e por reforço. O aprendizado supervisionado é aquele em que um conjunto de dados é apresentado com rótulos, ou seja, cada exemplo já é pertencente a uma determina classe. Neste método, o objetivo é aprender os padrões dos dados e usar esse conhecimento para predizer a classe de um novo conjunto de dados que não está previamente identificado. O aprendizado é feito por meio de alteração de parâmetros internos durante a fase de treinamento e a validação do percentual de acertos fazendo a predição das classes em um determinado conjunto de dados separado para validar a acurácia do modelo. Algumas das principais técnicas deste método são *KNN* (k-nearest neighbors) e *Random Forest.* O *KNN* tem como princípio a similaridade entre instâncias, pressupondo que dados semelhantes tendem a se agrupar próximos uns dos outros no espaço. Dessa forma, a predição de novas instâncias é feita baseando-se em instâncias existentes no conjunto de treinamento (Halder et al., 2024). O *Random Forest* consiste em combinar várias árvores de decisão, cujas

previsões são agrupadas por votação uniforme, utilizando amostragem de exemplos com reposição (*bagging*) e seleção aleatória de atributos (Faceli et al.,2011).

No aprendizado não supervisionado não se conhece as classes previamente, portanto a máquina deverá receber os dados de treinamento, identificar as similaridades e agrupar os conjuntos que são chamados de *clusters*, conforme Campos (2020). Essa similaridade é definida matematicamente considerando uma métrica que seja adequada ao problema. Destacam-se neste método os algoritmos *K-means* e *Apriori*. O *K-means* busca dividir elementos em *k* grupos, reunindo-os em elementos com maior similaridade entre si e o *Apriori* busca formar grupos de elementos que aparecem juntos frequentemente, formando regras de associação.

No aprendizado por reforço, de acordo com Ludermir (2021), o algoritmo não recebe exatamente a resposta correta, mas recebe recompensas ou punições após fazer uma hipótese baseada nos exemplos recebidos e esse processo é feito continuamente até que o algoritmo efetivamente aprenda as tarefas. O *Q-Learning* é uma das principais técnica deste método e consiste em utilizar uma função de valor (*Q-function*) para estimar a qualidade das ações em um determinado estado. Estas estimativas são armazenadas e atualizadas de forma interativa, auxiliando no aprimoramento de decisões futuras.

Apesar dos tipos clássicos terem proporcionado grandes avanços na área, segundo Chagas (2019), eles se mostraram limitados para resolver problemas modernos mais complexos que envolvem, por exemplo, reconhecimento de linguagem natural, visão computacional e quando os dados apresentam uma alta dimensionalidade. Considerando esse contexto, surgem abordagens mais modernas, com o *Deep Learning* (subgrupo das técnicas de *Machine Learning*), que emprega redes neurais profundas (Chagas, 2019), o *Automl (Automatized Machine Learning)*, que se trata de uma técnica que permite automatização de processos do Aprendizado de Máquina, como por exemplo, a seleção de algoritmos (Xin et al., 2021), e o aprendizado federado que essencialmente visa a construção de modelos de forma distribuída (McMahan et al., 2017).

O *Deep Learning* pode ser visto como uma evolução das redes neurais clássicas, pois baseia-se nos mesmos princípios, mas utiliza múltiplas camadas ocultas no processamento, o que permite que a máquina possa aprender com dados de vários níveis de abstração (Chagas, 2019). Segundo os autores Goodfellow,

Bengio e Courville (2016), um dos grandes diferenciais desta técnica, é a eliminação de praticamente toda a necessidade de extração manual de características, visto que apenas os dados brutos são necessários para o aprendizado. Contudo, para alcançar resultados mais eficientes, está técnica requer grande poder computacional e um grande volume de dados, porém e evolução na capacidade de processamento e o crescimento do *Big Data* tem contribuído para que o *Deep Learning* se torne uma das principais áreas de crescimento dentro da Inteligência Artificial.

Outra técnica que vem se destacando é o *Automl*, que busca automatizar tarefas do aprendizado de máquina, como a escolha de algoritmos, ajuste de hiperparâmetros e engenharia de atributos. Segundo Xin et al. (2021), o principal objetivo do *Automl* é possibilitar que pessoas com menos experiência em *Machine Learning* possam desenvolver soluções sem depender de um especialista, o que consequentemente impulsiona o uso de ferramentas que implementam essa técnica.

Enquanto o *Deep Learning* e *Automl* buscam otimizar a eficiência e automatizar tarefas, o Aprendizado Federado se preocupa com a maneira que os modelos serão treinados. Tradicionalmente, os dados costumam estar centralizados em um servidor e são treinados localmente, porém desta forma existe a possibilidade de armazenamento de dados particulares ou sensíveis, o que pode interferir em questões de privacidade. A proposta desta técnica é distribuir o processo de aprendizado em vários dispositivos, o que preserva a privacidade, pois dispensa o compartilhamento dos dados, além de permitir economias com transmissão de dados, visto que as informações ficam apenas no dispositivo que executa o treinamento e fornece os resultados ao servidor (McMahan et al., 2017).

Abordagens modernas de Aprendizado de Máquina estão tornando-se cada vez mais essenciais para diversas áreas do conhecimento. O *Deep Learning*, por meio de arquiteturas como as *CNNs* (*Convolutional Neural Networks*), consegue auxiliar no reconhecimento de imagens, pois essa arquitetura usa camadas convolucionais para fazer a extração de padrões locais (como bordas, texturas e formas) e camadas de *pooling* para redução da dimensionalidade, o que preserva as informações mais relevantes (Goodfellow; Bengio; Courville, 2016). Essa técnica impacta diretamente no campo da Saúde, pois ajuda os profissionais desta área a terem um diagnóstico mais preciso de exames de imagem (Yadav e Jadhav, 2019). Arquiteturas como as *RNNs (Recurrent Neural Networks)* também apresentam grandes progressos em processamento de linguagem natural e tradução automática,

visto que são projetadas para receber dados sequencialmente e possuem conexões recorrentes, que permitem manter informações sobre as entradas anteriores, possibilitando a preservação do contexto (Goodfellow; Bengio; Courville, 2016). O *Automl* tem ajudado instituições e pesquisadores na automatização de tarefas do fluxo do aprendizado de máquina, contribuindo fortemente na diminuição de tempo e de complexidade para chegar em modelos eficientes. Técnicas como a *NAS (Neural Architecture Search)* são aplicadas para automatizar o projeto de redes neurais, buscando identificar a estrutura de rede mais adequada considerando cada atividade e cada desafio enfrentado (Elsken; Metzen; Hutter, 2019). O Aprendizado Federado está sendo usado em áreas que requerem uma alta segurança de dados e privacidade, como as áreas de Finanças e da Saúde, e tem viabilizado o treinamento distribuído sem a necessidade de centralização de dados. O método *Federated Averaging* (*FedAvg*) está sendo amplamente utilizado para consolidar de forma ponderada as atualizações recebidas de modelos treinados localmente, levando em conta a quantidade de dados de cada cliente (McMahan et al., 2017).

As abordagens modernas de Aprendizado de Máquina refletem uma grande evolução na Inteligência Artificial, pois superam algumas limitações dos métodos clássicos e ampliam as possibilidades de uso, promovendo avanços nas capacidades de generalização, adaptação e autonomia. As novas técnicas trazem grandes contribuições no desenvolvimento de soluções mais eficientes, colaborando na resolução de problemas de diversas áreas e mostrando que a Inteligência Artificial é uma ferramenta indispensável em qualquer contexto contemporâneo.

REFERÊNCIAS

CAMPOS, R. S. **Desmistificando a inteligência artificial: uma breve introdução conceitual ao aprendizado de máquina**. Aoristo: International Journal of Phenomenology, Hermeneutics and Metaphysics, v. 3, n. 1, p. 106-123, 2020. Disponível em: https://doi.org/10.48075/aoristo.v3i1.24880. Acesso em: 07 out. 2025.

CHAGAS, E. T. de O. **Deep Learning e suas aplicações na atualidade**. Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano 04, Ed. 05, Vol. 04, pp. 05-26, Maio de 2019. Disponível em: https://doi.org/10.32749/nucleodoconhecimento.com.br/administracao/deep-learning. Acesso em: 08 out. 2025.

ELSKEN, T.; METZEN, J. H.; HUTTER, F. **Neural Architecture Search: A Survey.** Journal of Machine Learning Research, v. 20, n. 55, p. 1–21, 2019.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. de L. F. **Inteligência Artificial: uma abordagem de aprendizado de máquina**. Rio de Janeiro: LTC, 2011.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge: MIT Press, 2016.

HALDER, R. K.; UDDIN, M. N.; UDDIN, Md. A.; ARYAL, S.; KHRAISAT, A. **Enhancing K-nearest neighbor algorithm: a comprehensive review and performance analysis of modifications**. Journal of Big Data, v. 11, n. 113, 2024. Disponível em: https://doi.org/10.1186/s40537-024-00973-y. Acesso em: 07 out. 2025.

LUDERMIR, T. B. **Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências**. Estudos Avançados, São Paulo, v. 35, n. 101, p. 85–94, 2021. Disponível em: https://doi.org/10.1590/s0103-4014.2021.35101.007. Acesso em: 05 out. 2025.

MCMAHAN, B.; MOORE, E.; RAMAGE, D.; HAMPSON, S.; AGÜERA Y ARCAS, B. **A. Communication-Efficient Learning of Deep Networks from Decentralized Data.** In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017). Fort Lauderdale, FL, USA: Proceedings of Machine Learning Research, v. 54, p. 1273–1282, 2017.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997.

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, 2021, Yokohama, Japan. New York: ACM, 2021. p. 1–16.

YADAV, S. S.; JADHAV, S. M. **Deep convolutional neural network based medical image classification for disease diagnosis**. Journal of Big Data, v. 6, n. 113, 2019. Disponível em https://doi.org/10.1186/s40537-019-0276-2. Acesso em: 10 out. 2025.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

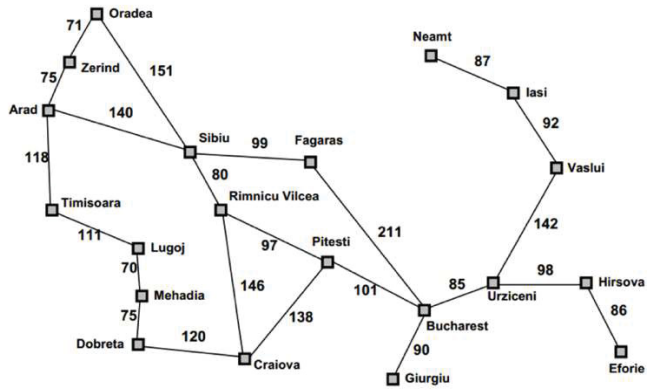
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

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

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

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

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

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

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

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

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

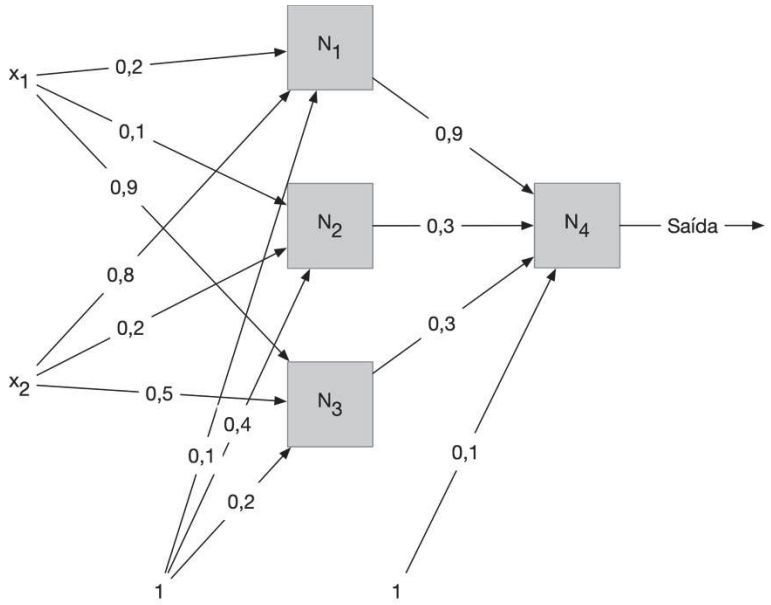
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

Questão 1:

a)

Model GPT 3.5

Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e máquinas capazes de realizar tarefas que normalmente requerem inteligência humana. Esses sistemas são projetados para simular processos cognitivos humanos, como aprendizado, raciocínio, resolução de problemas, percepção e compreensão da linguagem natural. A IA abrange uma ampla gama de técnicas e abordagens, incluindo aprendizado de máquina, redes neurais, processamento de linguagem natural, visão computacional e muito mais. Seu objetivo é criar sistemas que possam aprender e se adaptar a novas situações, tornando-se cada vez mais eficientes e autônomos em suas operações. A IA tem aplicações em diversos setores, como saúde, finanças, automotivo, jogos, assistentes virtuais, entre outros.

Model GPT 4

Inteligência Artificial (IA) refere-se à simulação da inteligência humana em máquinas que são programadas para pensar como humanos e imitar suas ações. O termo também pode ser aplicado a qualquer máquina que exiba traços associados à mente humana, como aprendizado e resolução de problemas. A capacidade ideal de uma máquina de IA é realizar tarefas que normalmente requerem inteligência humana, incluindo reconhecimento de padrões, interpretação de linguagem natural, raciocínio, aprendizado a partir da experiência, adaptação a novas situações e a capacidade de manipular objetos no mundo físico.

A IA é um campo amplo com muitas subáreas, incluindo:

1. **Aprendizado de Máquina (Machine Learning)**: Uma abordagem de IA que envolve o treinamento de algoritmos para aprender padrões em dados e fazer previsões ou decisões baseadas em dados novos, sem ser explicitamente programado para realizar a tarefa.
2. **Processamento de Linguagem Natural (PLN)**: Uma subárea da IA focada em dar às máquinas a capacidade de entender, interpretar e responder à linguagem humana de forma útil.
3. **Visão Computacional**: A capacidade de máquinas e sistemas para derivar informações significativas de imagens digitais, vídeos e outros inputs visuais e tomar ações ou fazer recomendações baseadas nessa informação.
4. **Robótica Inteligente**: A integração da IA em robôs, permitindo que eles realizem tarefas complexas, aprendam com suas experiências e interajam com o ambiente de maneira mais sofisticada.
5. **Sistemas de Recomendação**: Algoritmos de IA que analisam o comportamento do usuário e padrões de dados para sugerir itens ou ações que possam ser de interesse do usuário, como visto em plataformas de streaming e lojas online.

A IA pode variar desde sistemas simples, como uma calculadora ou um aplicativo de reconhecimento de voz, até sistemas extremamente complexos, como carros autônomos ou robôs com capacidade de aprendizado profundo. O desenvolvimento da IA promete transformar

muitosaspectos da vida cotidiana e tem aplicação em diversos campos, incluindo medicina, educação, finanças, logística, arte, entretenimento e muito mais.

b)

Conforme a resultado acima classifico a resposta do ChatGPT em AGIR RACIONALMENTE, pois o objetivo GPT é sempre fornecer uma resposta com o melhor resultado e rapidamente conforme as informações fornecidas, ou seja, ele adapta da situação (no caso o usuário), sempre se preocupando em tomar uma ação racional.

Ao contrário da abordagem AGIR COMO HUMANO, onde o objetivo é imitar o comportamento humano, algo que não é esperado de uma IA, no qual o objetivo é ter resultados precisos, racionais e rápidos diante das mais diversas situações. O mesmo podemos desconsiderar a abordagem de PENSAR COMO OS HUMANOS, cujo objetivo é implementar fielmente o processo de pensamento, e PENSAR RACIONALMENTE, que visa sempre fornecer um resultado logicamente correto.

c)

ChatGPT é uma inteligência artificial baseada em Generative Pre-Trained Transformer, traduzido do inglês significa **Transformador Generativo Pré-Treinado**, disponibilizado via chatbot por meio de uma interface web e via API REST, permitindo a integração com diversas aplicações.

O ChatGPT foi treinado com uma vasta gama de informações (dados), permitindo que ele compreenda e responda a diversas perguntas dos mais variados temas, por exemplo, matemática, saúde, pontos turísticos e entre outros. Por isso, ele também é considerado um modelo de IA conversional e linguagem natural.

Os modelos de redes neurais utilizados na criação do ChatGPT são baseados na técnica transformer mencionada no artigo Vaswani et al. *(2017). Com isso, modelos do ChatGPT conseguem realizar geração de textos, interpretação de áudios, criação de imagens, análise de imagens se baseando nas informações fornecidas pelo usuário via plataforma web ou API rest*.

Referências

1. Achiam, Josh, et al. "Gpt-4 technical report." arXiv preprint arXiv:2303.08774 (2023).

2. Alawida, Moatsum, et al. "A comprehensive study of ChatGPT: advancements, limitations, and ethical considerations in natural language processing and cybersecurity." Information 14.8 (2023): 462.

3. Chen, Mark, et al. "Generative pretraining from pixels." International conference on machine learning. PMLR, 2020.

4. Deng, Jianyang, and Yijia Lin. "The benefits and challenges of ChatGPT: An overview." Frontiers in Computing and Intelligent Systems 2.2 (2022): 81-83.

5. OpenAI, R. "Gpt-4 technical report. arxiv 2303.08774." View in Article 2 (2023): 13.

6. Vaswani, Ashish, et al. "Attention is all you need."*Advances in neural information processing systems* 30 (2017).

7. Wu, Tianyu, et al. "A brief overview of ChatGPT: The history, status quo and potential future development." IEEE/CAA Journal of Automatica Sinica 10.5 (2023): 1122-1136.

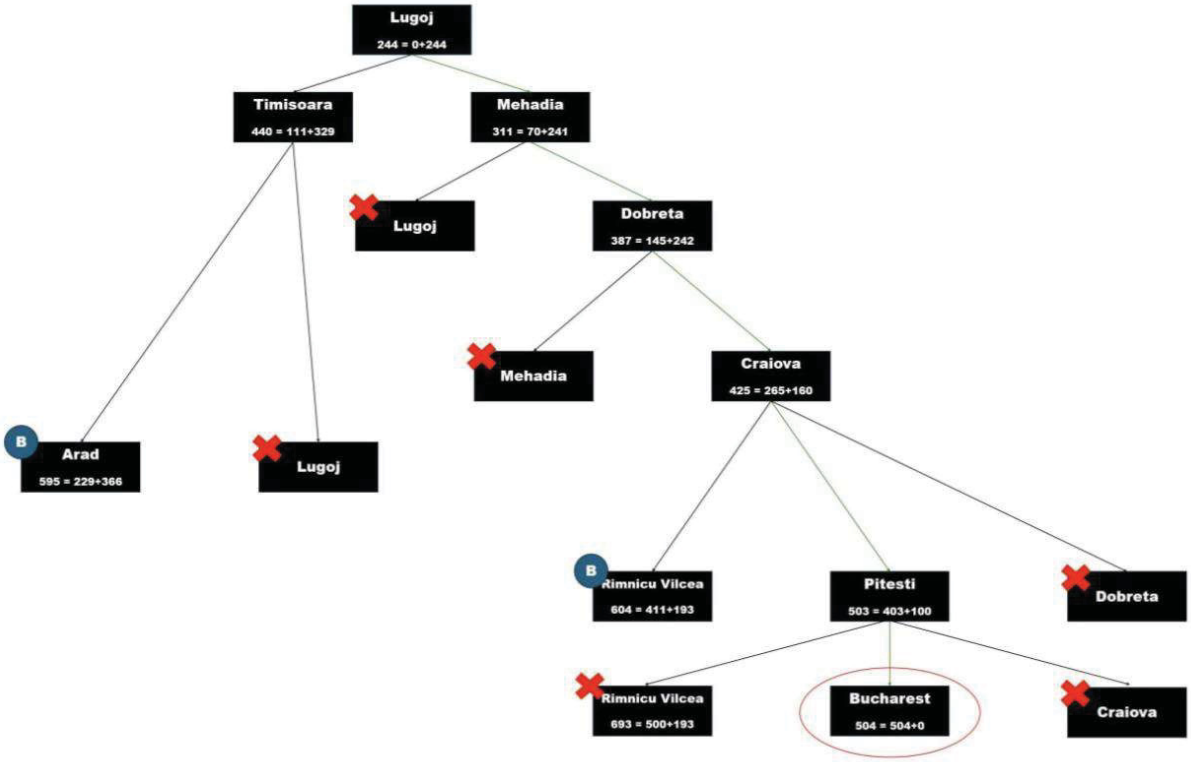
d)

Agir racionalmente (classificação definida)

O ChatGPT se alinha mais estreitamente com a abordagem de agir racionalmente. Ele é projetado para realizar tarefas de maneira lógica e eficiente, produzindo respostas que atendem aos requisitos de uma dada solicitação de acordo com seu treinamento e as diretrizes predefinidas. Essa capacidade de gerar respostas apropriadas e úteis num vasto domínio de conhecimento, otimizando para objetivos específicos (como responder a perguntas, fornecer informações, criar imagens, interpretar áudio, etc.), exemplifica a ação racional conforme definido na inteligência artificial.

Questão 2:

a)



Questão 3:

Premissas:

P1: $p \rightarrow q$
P2: $q \rightarrow r$
P3: $\neg r \vee p$

Conclusão: Q: q ↔ p

Provar que: p →q, q →r, ~r v p ⊢ q ↔ p

R1 : p → q

R2 : q → r

R3 : ~r v p

R4 : r → p (EI R3)

R5 : q → p (SH R2 e R4)

R6 : (p → q) ^(q → p) (CONJ R1 e R5)

R7 : p ↔ q (BICOND R6)

O argumento é VÁLIDO

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

Referência:

Monard, M.C., Nicoletti, M.C., Noguchi.R.H., O Cálculo Proposicional: Uma abordagem voltada à compreensão da linguagem Prolog, Notas Didáticas do ICMC-USP, 1992

Questão 4:

a)

N1 = (0,2x-3) + (0,8x1) + (0,1x1)

N1 = (-0,6) + 0,8 + 0,1

N1 = 0,3

b)

N2 = (0,1x-3) + (0,2x1) + (0,4x1)

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

N2 = 0,3

c)

N3 = (0,9x-3) + (0,5x1) + (0,2x1)

N3 = -(2,7) + 0,5 + 0,2

N3 = -2,00

d)

N4 = (0,9x0,3) + (0,3x0,3) + (0,3x-2,0) + (0,1x1)

N4 = 0,27 + 0,09 + (-0,6) + 0,1

N4 = -0,14

N4 = tahn(-0,14) = -0,13909

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

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

B - RESOLUÇÃO

Questão 1:

a)

```
# Função read_csv para importar os dados da pasta do computador
dados = pd.read_csv('precos_carros_brasil.csv')

#Quantidade de linhas e colunas antes do pré-processamento dos dados
dados.shape
(267542, 11)
```

b)

```
#Verificando se existem valores faltantes nos dados
dados.isna().any()
year_of_reference True
month_of_reference True
fipe_code True
authentication True
brand True
model True
fuel True
gear True
engine_size True
year_model True
avg_price_brl True
dtype: bool

# No resultado do comando acima, verificamos que existem valores fal
tantes em todas as colunas/variáveis


# Verificando a quantidade de valores faltantes por coluna
dados.isna().sum()
year_of_reference     65245
month_of_reference    65245
fipe_code         65245
authentication        65245
brand                 65245
model                 65245
fuel                  65245
gear                  65245
engine_size           65245
year_model            65245
avg_price_brl         65245
dtype: int64


#Optou-se por remover os valores de linhas faltantes
dados = dados.dropna(axis=0)
dados.shape
```

```
#Quantidade de linhas e colunas após remoção de linhas faltantes
(202297, 11)
```

c)

```
# Verificação se existem dados duplicados
dados.duplicated().sum()

# Remoção dos dados duplicados no dataframe original
dados = dados.drop_duplicates()

# Imprimir o tamanho do dataframe após remoção das duplicações
dados.shape
(202295, 11)
```

d)

```
# Imprimir os tipos de dados do dataframe para identificação da quan
tidade de colunas de cada tipo
dados.dtypes

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

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

# Resumo das variáveis numéricas - Imprime alguns valores de medidas de
tendências centrais
dados[numericas_cols].describe()
```

	year_of_reference	year_model	avg_price_brl
count	202297.000000	202297.000000	202297.000000
mean	2021.564694	2011.271527	52756.909153
std	0.571903	6.376234	51628.677716
min	2021.000000	2000.000000	6647.000000
25%	2021.000000	2006.000000	22855.000000
50%	2022.000000	2012.000000	38027.000000
75%	2022.000000	2016.000000	64064.000000
max	2023.000000	2023.000000	979358.000000

```
# Resumo das variáveis categóricas - Imprime alguns valores de estatística
descritiva
dados[categoricas_cols].describe()
```

count	202295	202295	202295	202295	202295	202295	202295	202295
unique	12	2091	202295	6	2112	3	2	29
top	January	003281-6	cfzlctzfwrcp	Fiat	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6
freq	24260	425	1	44962	425	168684	161883	47420

```
# Converter os dados relativos a anos de float para int
dados['year_of_reference'] = dados['year_of_reference'].astype(int)
dados['year_model'] = dados['year_model'].astype(int)
```

e)

```
# Contagem de valores por modelo
contagem_modelos = dados['model'].value_counts()
total_modelos_unicos = len(contagem_modelos)
print("Total de modelos únicos:", total_modelos_unicos)

Total de modelos únicos: 2112

# Contagem de valores por marca
contagem_por_marca = dados['brand'].value_counts()
total_marcas_unicas = len(contagem_por_marca)
print("Total de marcas únicas:", total_marcas_unicas)

Total de marcas únicas: 6
```

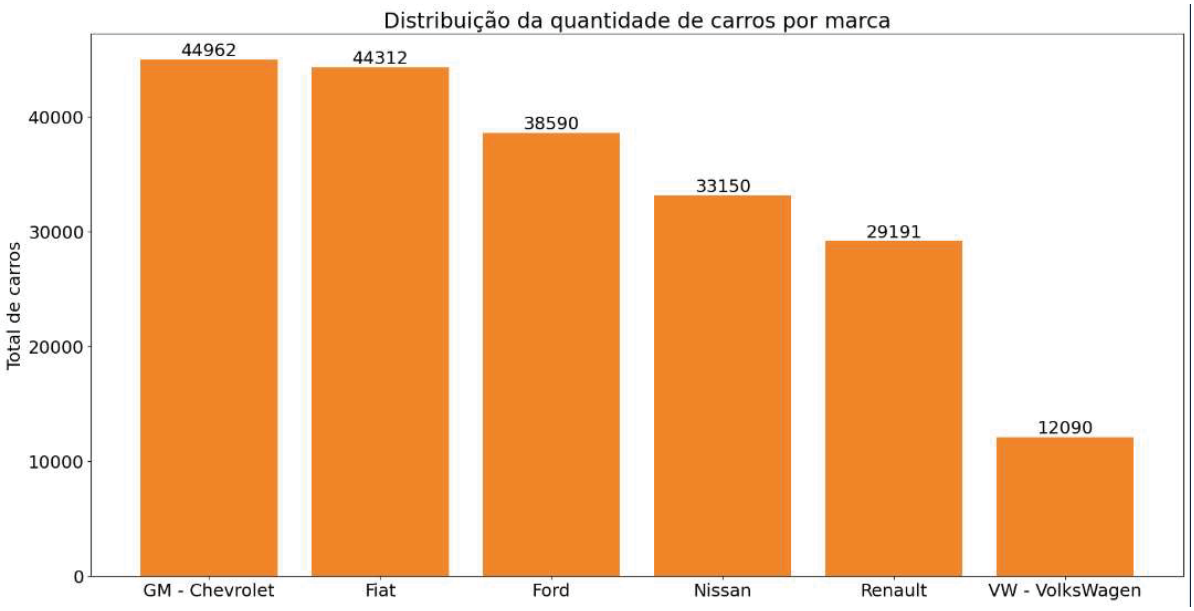
f)

Ao carregar o arquivo, identificou-se um total de 267.542 linhas e 11 colunas. Posteriormente, observou-se que 65.245 linhas continham valores faltantes, as quais foram removidas, resultando em 202.297 linhas. Além disso, foram detectadas e excluídas 2 linhas duplicadas, resultando em 202.295 linhas únicas. Foi realizada uma contagem de valores por modelo, totalizando 2.112, e por marca, totalizando 6.

Questão 2:

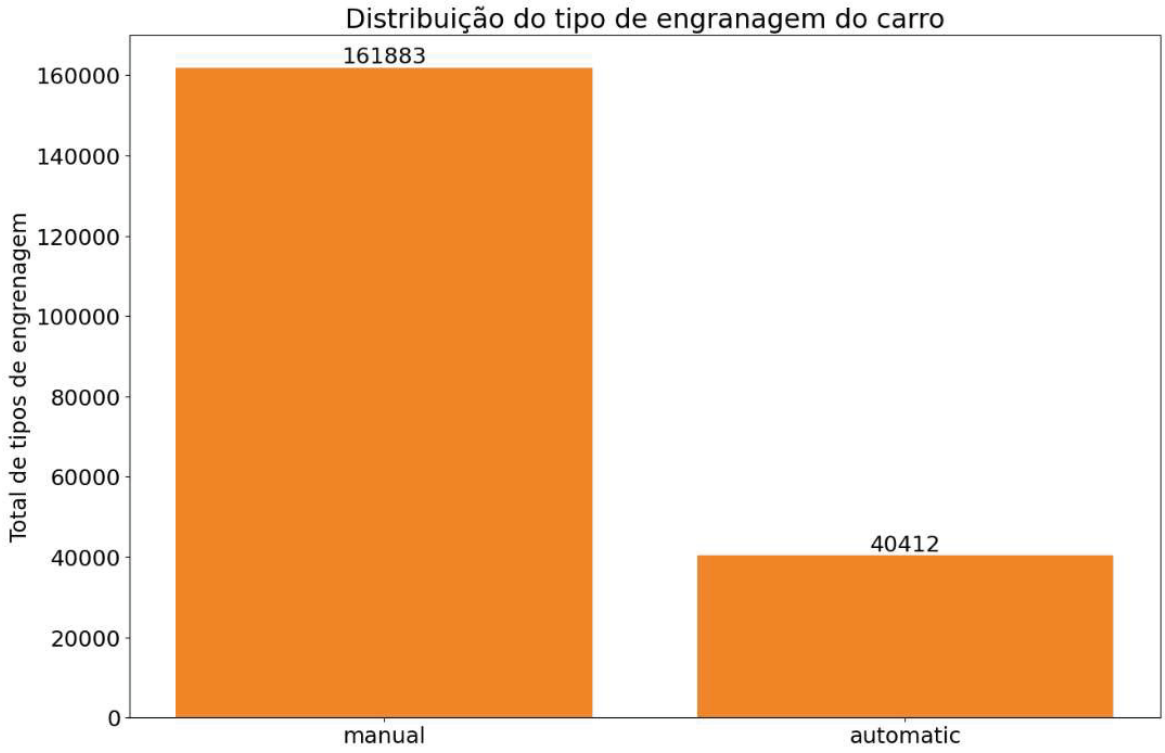
a)

```
# Gráfico da distribuição por marca
plt.figure(figsize=(20,10))
grafico=plt.bar(dados['brand'].unique(),dados['brand'].value_counts())#plt.
bar para gráfico de barras
plt.bar(dados['brand'].unique(), dados['brand'].value_counts()) # plt.bar
para gráfico de barras. Variáveis nos eixos X e Y
plt.title('Distribuição da quantidade de carros por marca') # plt.title
para inserir título no gráfico
plt.ylabel('Total de carros') # # plt.ylabel para inserir título no gráfico
plt.bar_label(grafico,fmt="%.00f",size=18, label_type="edge");
```



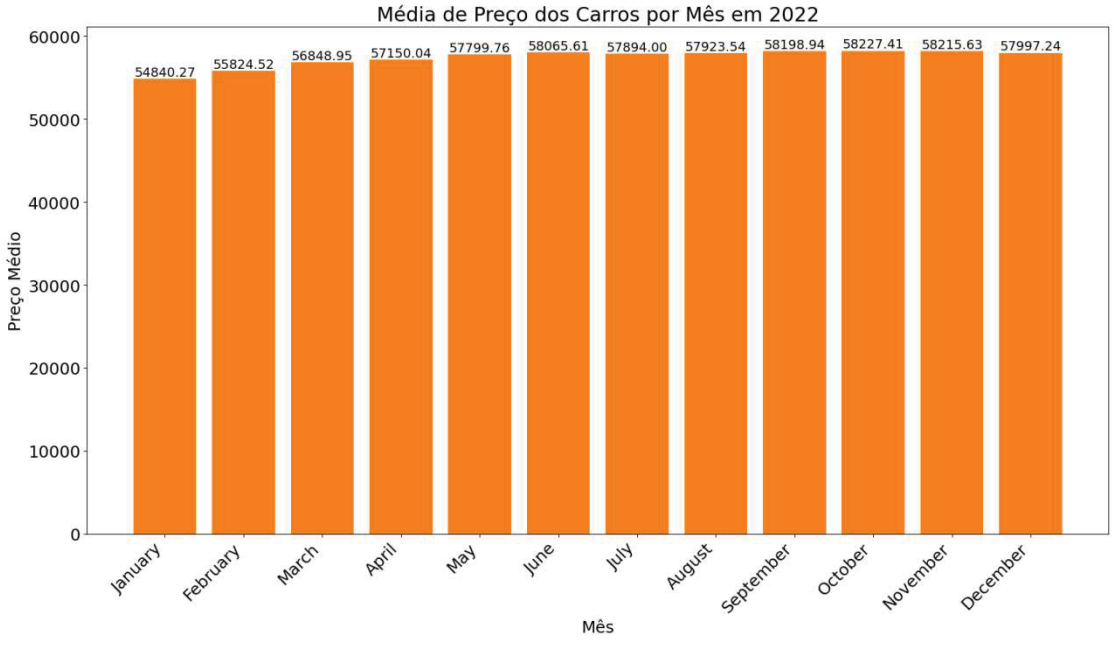
b)

```
# Gráfico da distribuição por tipo de engranagem do carro
plt.figure(figsize=(15,10))
#plt.bar para gráfico de barras
grafico1=plt.bar(dados['gear'].unique(),dados['gear'].value_counts())
# plt.bar gráfico de barras.Var nos eixos X e Y
plt.bar(dados['gear'].unique(), dados['gear'].value_counts())
# plt.title para inserir título no gráfico
plt.title('Distribuição do tipo de engranagem do carro')
# plt.ylabel para inserir título no gráfico
plt.ylabel('Total de tipos de engrenagem')
plt.bar_label(grafico1,fmt="%.00f", label_type="edge");
plt.rcParams.update({'font.size': 18})
```



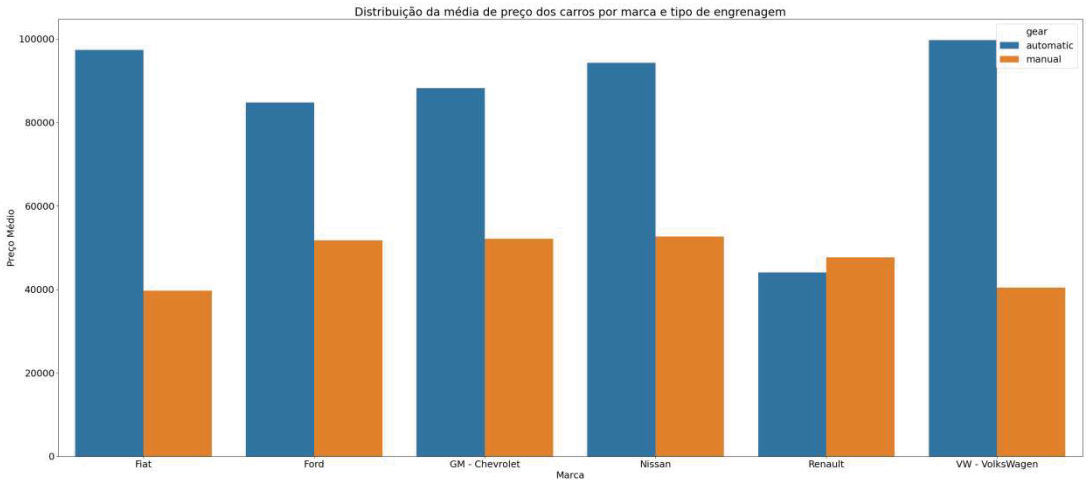
c)

```
# Filtrar os dados para o ano de 2022
dados_2022 = dados[dados['year_of_reference'] == 2022].copy()

# Converter a coluna 'year_of_reference' para valores inteiros
dados_2022['year_of_reference'] =
dados_2022['year_of_reference'].astype(int)

# Calculando a média por mês
media_preco_mes_2022 = dados_2022.groupby(
['year_of_reference', 'month_of_reference'])['avg_price_brl'].mean()

# Ordenando os resultados pelo mês
media_preco_mes_formatada = media_preco_mes_2022.reindex(
pd.date_range(start='2022-01-01', end='2022-12-01',
freq='MS').strftime('%B'), level=1)

# Utilizando a função reset_index para criar uma ordem e facilitar a cria-
ção do gráfico
media_preco_mes_formatada =
media_preco_mes_formatada.reset_index(name='avg_price_brl')

# Gráfico da Média de Preço dos Carros por Mês em 2022
plt.figure(figsize=(20,10))
grafico2=plt.bar(media_preco_mes_formatada['month_of_reference'],media_pre-
co_mes_formatada['avg_price_brl'])#plt.bar para gráfico de barras
plt.bar(media_preco_mes_formatada['month_of_reference'],media_preco_mes_for
matada['avg_price_brl']) # plt.bar para gráfico de barras. Variáveis nos
eixos X e Y
plt.title('Média de Preço dos Carros por Mês em 2022')
plt.xlabel('Mês')
plt.ylabel('Preço Médio')
plt.bar_label(grafico2, fmt="{:.2f}", size=14, label_type="edge")
plt.xticks(rotation=45, ha='right')
plt.rcParams.update({'font.size': 18})
```



d)

```
# colocar em um dataframe os dados que serão usados nos gráficos
mediaPrecoMarcaEngrenagem = dados.groupby(['brand', 'gear'])
['avg_price_brl'].mean().round(2)

# Nomeando a coluna do preço médio
mediaPrecoMarcaEngrenagem =
mediaPrecoMarcaEngrenagem.reset_index(name='Preço Médio')
mediaPrecoMarcaEngrenagem.head(15)

# Criando o gráfico
plt.figure(figsize=(35,15))
plt.xlabel("Marca") # Título do eixo X
plt.ylabel("Preço Médio") # Título do eixo Y
plt.title("Distribuição da média de preço dos carros por marca e tipo de
engrenagem") # Aumento do tamanho da fonte
sns.barplot(x='brand', y='Preço Médio', hue='gear', data=mediaPrecoMarcaEn-
grenagem,hue_order=['automatic', 'manual']);
plt.rcParams.update({'font.size': 20})
```



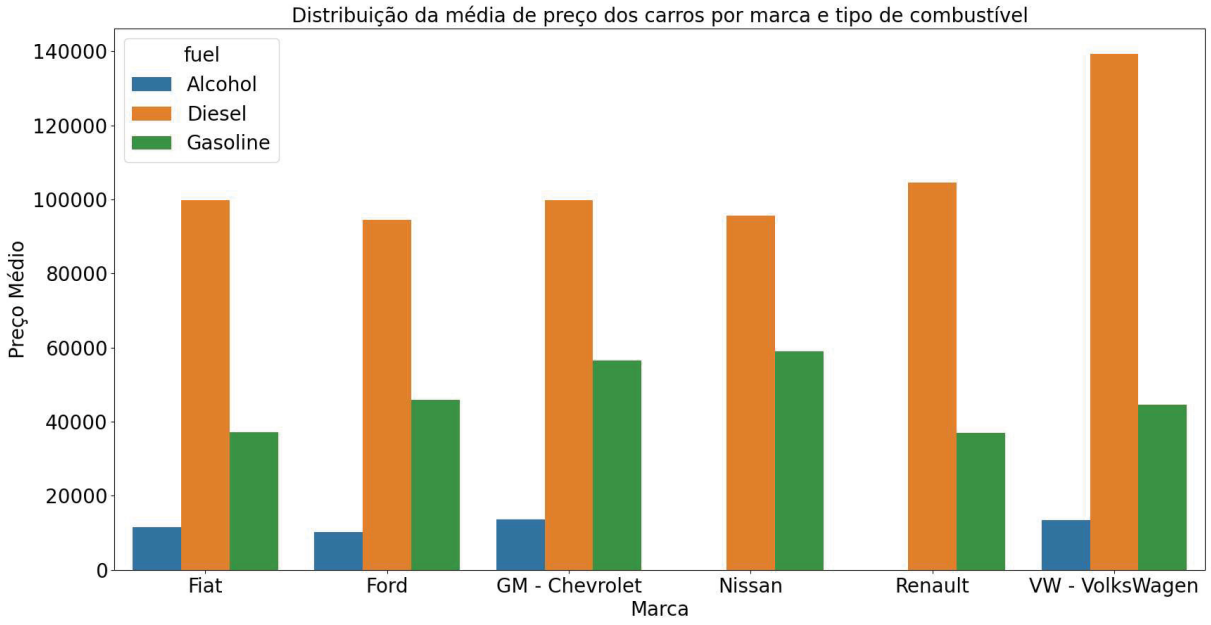
e)

No gráfico da média de preço dos carros por marca e tipo de engrenagem, destaca-se que a Volkswagen (VW) lidera com o valor médio mais alto em carros automáticos, seguida pela Nissan em carros manuais. Enquanto isso, a Renault e a Fiat registram os valores médios mais baixos, respectivamente, nessas categorias. Essa análise evidencia as variações nos preços médios de acordo com a marca e o tipo de transmissão.

f)

```
# Colocar em um dataframe os dados que serão usados nos gráficos
mediaPrecoMarcaCombustivel = dados.groupby(['brand', 'fuel'])
['avg_price_brl'].mean().round(2)

# Nomeando a coluna do preço médio
mediaPrecoMarcaCombustivel =
mediaPrecoMarcaCombustivel.reset_index(name='Preço Médio')


# Criando o gráfico
plt.figure(figsize=(20,10))
```

```
plt.xlabel("Marca", fontsize=20) # Título do eixo X
plt.ylabel("Preço Médio", fontsize=20) # Título do eixo Y
plt.title("Distribuição da média de preço dos carros por marca e tipo de
combustível", fontsize=20) # Aumento do tamanho da fonte
sns.barplot(x='brand', y='Preço Médio', hue='fuel', data=mediaPrecoMarca-
Combustivel,hue_order=['Alcohol', 'Diesel','Gasoline']);
```



g)

No gráfico de distribuição da média de preço dos carros por marca e tipo de combustível, destaca-se que o Diesel é o tipo mais caro, seguido por Gasolina e Álcool, respectivamente. As marcas com os valores mais elevados nas respectivas categorias são Volkswagen (VW), Nissan e GM - Chevrolet. Por outro lado, as marcas com os valores mais baixos são Ford, Renault e Ford, respectivamente. Essa análise evidencia as variações nos preços médios conforme o tipo de combustível e a marca do carro.

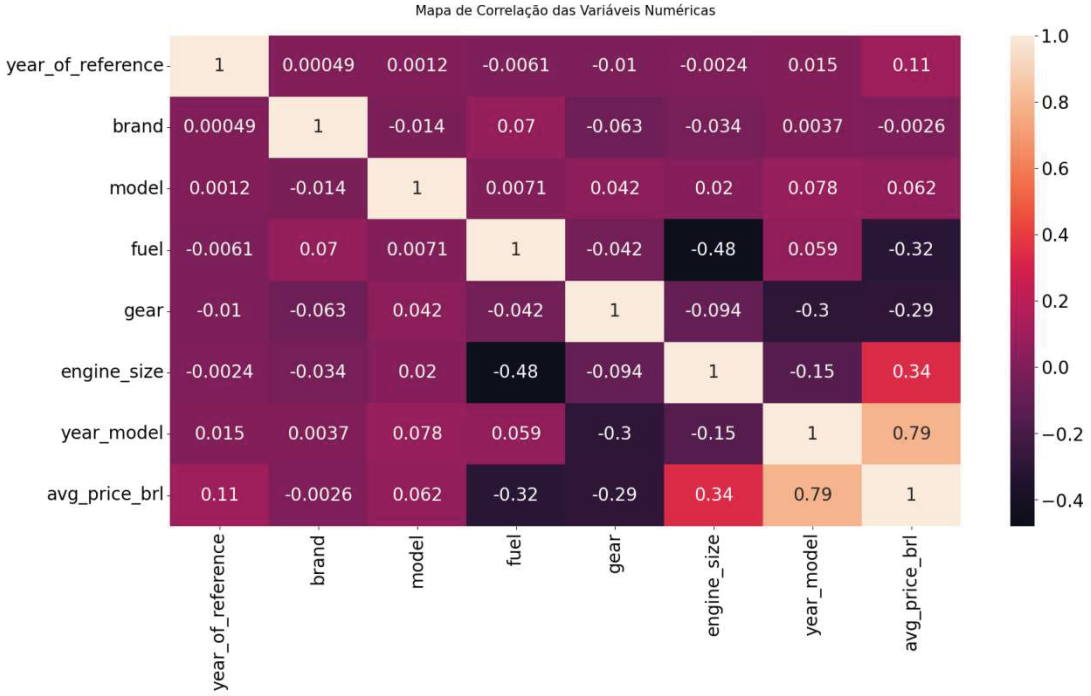
Questão 3:

a)

```
# Mostrar o tipo de dados
dados.dtypes
```

```
year_of_reference       int32
month_of_reference     object
fipe_code              object
authentication         object
brand                  object
model                  object
fuel                   object
gear                   object
engine_size            object
year_model              int32
avg_price_brl         float64
dtype: object
```

```
# Exibindo a quantidade de valores únicos por cada variável
dados.nunique()

year_of_reference          3
month_of_reference        12
fipe_code               2091
authentication        202295
brand                      6
model                   2112
fuel                       3
gear                       2
engine_size               29
year_model                24
avg_price_brl          88510
dtype: int64

# Atribuir os valores para outro dataframe
dadosPredicao = dados

# Remoção das colunas desnecessárias
dadosPredicao = dadosPredicao.drop('month_of_reference', axis=1)
dadosPredicao = dadosPredicao.drop('fipe_code', axis=1)
dadosPredicao = dadosPredicao.drop('authentication', axis=1)
dadosPredicao.head(10)

# Alterar valores categóricos em numéricos
#brand                  object
#model                  object
#fuel                   object
#gear                   object
#engine_size            object

dadosPredicao['brand'] =
LabelEncoder().fit_transform(dadosPredicao['brand'])
dadosPredicao['model'] = LabelEncoder().fit_transform(dadosPredicao['mo-
del'])
dadosPredicao['fuel'] = LabelEncoder().fit_transform(dadosPredicao['fuel'])
dadosPredicao['gear'] = LabelEncoder().fit_transform(dadosPredicao['gear'])
dadosPredicao['engine_size'] =
LabelEncoder().fit_transform(dadosPredicao['engine_size'])

# Verificar quantidade de marcas
dadosPredicao['brand'].value_counts()

brand
0    44962
5    44312
2    38590
1    33150
4    29191
3    12090
Name: count, dtype: int64

# Mapa de correlação das variáveis numéricas com variável Target
plt.figure(figsize=(20,10))
sns.heatmap(dadosPredicao.corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```



b)

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

# Variável Y contém apenas a variável target
Y = dadosPredicao['avg_price_brl']

# Divisão dos parâmetros para treino (75%) e teste (25%)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
random_state = 42)

# Verificação dos dados do treinamento
print(X_train.shape)
X_train.head(10)

(151721, 7)

# Verificação dos dados do teste
print(X_test.shape)
X_test.head(10)

(50574, 7)
```

c)

```
### RandomForest

# Algoritmo Random Forest, sem especificar nenhum parâmetro (número de ár-
vores, número de ramificações, etc)
model_rf = RandomForestRegressor()

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

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

# Valores preditos
valores_preditos_rf

array([ 42187.96172948,  12079.22055913,   8922.48170197, ...,
109823.1116493 ,   9643.05146144,  22960.47166006])

### XGBoost

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

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

array([ 44367.266,  12109.644,  10512.041, ..., 110412.94 ,  11347.781,
24380.184], dtype=float32)

### Random Forest com alteração de parâmetros

# Com parâmetros iguais aos do exercício da disciplina
model_rf_parametros = RandomForestRegressor(max_depth=29,
min_samples_leaf=32, min_samples_split=28,
n_estimators=208, random_state=43)

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

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

d)

```
# Valores preditos no Random Forest sem inclusão de parâmetros
valores_preditos_rf = model_rf.predict(X_test)


# Valores preditos no XGboost
valores_preditos_xgboost = model_xgboost.predict(X_test)

# Valores preditos no Random Forest com alteração de parâmetros
valores_preditos_rf_parametros = model_rf_parametros.predict(X_test)
```

e)

```
# Análise da Importância das variáveis

# Random Forest

model_rf.feature_importances_
feature_importances = pd.DataFrame(model_rf.feature_importances_, index =
X_train.columns, columns=['importance']).sort_values('importance', ascen-
ding = False)
feature_importances

# XGboost

model_xgboost.feature_importances_
```

```
feature_importances = pd.DataFrame(model_xgboost.feature_importances_, in-
dex = X_train.columns, columns=['importance']).sort_values('importance',
ascending = False)
feature_importances

# Random Forest com alteração de parâmetros

model_rf_parametros.feature_importances_
feature_importances = pd.DataFrame(model_rf_parametros.feature_importan-
ces_, index = X_train.columns, columns=['importance']).sort_values('impor-
tance', ascending = False)
feature_importances
```

f)

Na análise de importância de variáveis, observou-se uma consistência em relação à influência das características do veículo, como o tamanho do motor e o ano do modelo, na predição do preço médio. Além disso, fatores como o tipo de combustível e a transmissão também desempenharam um papel significativo. A marca do veículo e o modelo específico tiveram uma importância menor, indicando que outros atributos podem ser mais relevantes na determinação do preço médio.

g)

```
# Random Forest - MSE - calcula o erro quadrático médio das predições do
nosso modelo. Quanto maior o MSE, pior é o modelo.
mse = mean_squared_error(Y_test, valores_preditos_rf)
mse
20013214.759729225

# Random Forest - O MAE calcula a média da diferença absoluta entre o valor
predito e o valor real. Nesse caso, os erros são penalizados linearmente,
ou seja, todos terão o mesmo peso na média.
mae = mean_absolute_error(Y_test, valores_preditos_rf)
mae
2336.850652570212

# Random Forest - O R² é uma métrica que varia entre 0 e 1 e é uma razão
que indica o quão bom o nosso modelo. Quanto maior seu valor, melhor é o
modelo
r2_score(Y_test, valores_preditos_rf)
0.9925636190237587

# XGBoost - # MSE - calcula o erro quadrático médio das predições do nosso
modelo. Quanto maior o MSE, pior é o modelo.
mse = mean_squared_error(Y_test, valores_preditos_xgboost)
mse
39450171.01302586

# XGBoost
# O MAE calcula a média da diferença absoluta entre o valor predito e o va-
lor real.
# Nesse caso, os erros são penalizados linearmente, ou seja, todos terão o
mesmo peso na média.
mae = mean_absolute_error(Y_test, valores_preditos_xgboost)
mae
3669.041110367435
```

```
# XGBoost
# O R² é uma métrica que varia entre 0 e 1 e é uma razão que indica o quão
bom o nosso modelo.
# Quanto maior seu valor, melhor é o modelo
r2_score(Y_test, valores_preditos_xgboost)

0.9853413604584382

# Random Forest com alteração de parâmetros
# MSE - calcula o erro quadrático médio das predições do nosso modelo.
Quanto maior o MSE, pior é o modelo.
mse = mean_squared_error(Y_test, valores_preditos_rf_parametros)
mse
146317605.61954153

# Random Forest com alteração de parâmetros
# O MAE calcula a média da diferença absoluta entre o valor predito e o va-
lor real.
# Nesse caso, os erros são penalizados linearmente, ou seja, todos terão o
mesmo peso na média.
mae = mean_absolute_error(Y_test, valores_preditos_rf_parametros)
mae
4557.4308940321735

# Random Forest com alteração de parâmetros
# O R² é uma métrica que varia entre 0 e 1 e é uma razão que indica o quão
bom o nosso modelo.
# Quanto maior seu valor, melhor é o modelo
r2_score(Y_test, valores_preditos_rf_parametros)
0.9456322498918177
```

h)

O modelo que apresentou melhor desempenho foi o **XGBoost com métrica R^2** (0.97324), considerando também que apresenta um bom equilíbrio entre precisão (MAE) e explicação da variabilidade (R^2).

Modelo	Métrica	Resultado
Random Forest	MAE	4217.2601416528505
Random Forest	R^2	0.9797074555930972
Random Forest	MSE	54612458.73956774
Random Forest com alteração de parâmetros	MAE	5390.8614339574015
Random Forest com alteração de parâmetros	R^2	0.9389028768223593
Random Forest com alteração de parâmetros	MSE	164428080.17264596
XGBoost	MSE	71996968.4208215
XGBoost	MAE	4913.739671481548
XGBoost	R^2	0.9732478318581291

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

    - Coeficiente de determinação: R^2

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

onde y_i é o valor observado, $\widehat{y_i}$ é o valor predito e $\overline{y}$ 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%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

Questão 1:

```
# Carregar pacotes
library(mlbench)      # Para carregar a base de dados Satellite
library(caret)        # Para treinar os modelos de aprendizado de máquina
library(randomForest) # Para treinar o modelo Random Forest
library(kernlab)      # Para treinar o modelo SVM
library(e1071)        # Para treinar o modelo SVM
library(neuralnet)      # Para treinar o modelo Rede Neural

# Carregar a base de dados Satellite
data(Satellite)

# Carregar e selecionar apenas as colunas x.17, x.18, x.19, x.20 e a coluna
de classes
dados_selecionados <- Satellite[, c("x.17", "x.18", "x.19", "x.20", "clas-
ses")]

# Visualizar a estrutura inicial dos dados
cat("\nEstrutura inicial dos dados:\n")
str(dados_selecionados)

# Criação de partições contendo 80% para treino e 20% para teste
set.seed(123)
indice <- createDataPartition(dados_selecionados$classes, p = 0.8, list =
FALSE)

# Criar conjuntos de treinamento e teste
dados_treino <- dados_selecionados[indice, ]
dados_teste <- dados_selecionados[-indice, ]
```

```
# Exibir a quantidade de dados nos conjuntos de treino e teste
cat(paste("\nQuantidade de dados no conjunto de treino:", nrow(dados_trei-
no), "\n"))
cat(paste("Quantidade de dados no conjunto de teste:", nrow(dados_teste),
"\n\n"))

#Treinamento de modelos RandomForest, SVM e RNA para predição dos dados
cat("Treinando Modelo Random Forest:\n")
modelo_rf <- randomForest(classes ~ ., data = dados_treino)

cat("\nTreinando Modelo SVM:\n")
modelo_svm <- svm(classes ~ .,   data = dados_treino,  kernel = "radial")

cat("\nTreinando Modelo RNA:\n")
modelo_rna <- caret::train(classes ~ ., data = dados_treino, method =
"nnet")

# Avaliação dos modelos com base em suas matrizes de confusão
pred_rf <- predict(modelo_rf, newdata = dados_teste)
confusao_rf <- confusionMatrix(pred_rf, dados_teste$classes)

pred_svm <- predict(modelo_svm, newdata = dados_teste)
confusao_svm <- confusionMatrix(pred_svm, dados_teste$classes)

pred_rna <- predict(modelo_rna, newdata = dados_teste)
confusao_rna <- confusionMatrix(pred_rna, dados_teste$classes)

# Comparação das métricas de desempenho dos modelos
metricas <- data.frame(Modelo = c("Random Forest", "SVM", "RNA"),
                       Acuracia = c(confusao_rf$overall['Accuracy'], confu-
sao_svm$overall['Accuracy'], confusao_rna$overall['Accuracy'])
)
cat("\n\nResumo dos Modelos")
print(metricas)
```

As tarefas de número 1 a 3 serão detalhadas em outro documento e no arquivo R. Em relação as tarefas 4 e 5, após o treinamento dos modelos RandomForest, SVM e RNA(nnet), o modelo SVM foi identificado como o melhor dos 3 modelos, pois leva um tempo de execução semelhante ao RandomForest, porém tem uma acurácia levemente superior de acordo com a comparação entre as matrizes de confusão. O RNA(nnet) teve um acurácia inferior ao SVM e ao RandomForest e ainda levou muito mais tempo de execução.

Complementando a escolha pelo SVM, a equipe fez testes usando o pacote caret para o treinar os modelos RandomForest e SVM, porém o treinamento levava vários minutos, o que dificulta a execução de testes e por isso optamos por utilizar os pacotes “randomForest” para o RandomForest e os pacotes “e1071” e “kernlab” para usar no SVM. Estes pacotes deixaram a execução muitos mais rápida, pois levava segundos, enquanto usando o pacote caret, levava alguns minutos para os mesmos algoritmos. Adicionalmente, além do ganho de tempo, os resultados foram semelhantes em relação a acurácia dos modelos.

As métricas utilizadas para escolher o SVM como o melhor modelo para usar neste conjunto de dados foram a acurácia apresentada nas matrizes de confusão em conjunto com o tempo de execução do treinamento.

Abaixo constam os resultados das acurácias da matriz de confusão de cada modelo que foram usadas para a escolha do melhor modelo.

```
Resumo dos Modelos> print(metricas)
          Modelo  Acuracia
1 Random Forest 0.8512461
2           SVM 0.8551402
3           RNA 0.8045171
```

Questão 2:

Abaixo é exibido o código completo do exercício. Após o código, é mostrada a resposta discursiva da questão em seguida a saída dos últimos comandos do código.

```
# Carregar pacotes
library(mlbench)        # Para carregar o conjunto de dados Volumes.csv
library(caret)          # Para treinar os modelos de aprendizado de máquina
library(randomForest)   # Para treinar o modelo Random Forest
library(kernlab)        # Para treinar o modelo SVM
library(e1071)          # Para treinar o modelo SVM
library(neuralnet)      # Para treinar o modelo Rede Neural

# Carregar o arquivo Volumes.csv
volumes <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv", sep =
";", header = TRUE)

# Visualizar a estrutura inicial dos dados
cat("\nEstrutura inicial dos dados:\n")
str(volumes)

# Eliminar a coluna NR
volumes <- volumes[, -1]  # Exclui a primeira coluna

# Visualizar a estrutura inicial dos dados
cat("\nDados após exclusão da coluna NR e conversão para númerico:\n")
str(volumes)

# Criação de partições contendo 80% para treino e 20% para teste
set.seed(123)
indice <- createDataPartition(volumes$VOL, p = 0.8, list = FALSE)

# Criar conjuntos de treinamento e teste
dados_treino <- volumes[indice, ]
dados_teste <- volumes[-indice, ]

cat(paste("\nQuantidade de dados no conjunto de treino:", nrow(dados_trei-
no)))
cat(paste("\nQuantidade de dados no conjunto de teste:",
nrow(dados_teste)))

# Treinar os modelos
# Random Forest
cat("\n\nTreinando Modelo Random Forest:\n\n")
modelo_rf <- train(VOL ~ ., data = dados_treino, method = "rf")
```

```
# SVM
cat("Treinando Modelo SVM\n\n")
modelo_svm <- train(VOL ~ ., data = dados_treino, method = "svmRadial")

# Redes Neurais
cat("Treinando Modelo Redes Neurais\n\n")
modelo_neuralnet <- neuralnet(VOL ~ ., data = dados_treino)

cat("Treinando Modelo alométrico de SPURR \n\n")
# Modelo alométrico de SPURR
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, data = dados_treino, start = list(b0
= 0.5, b1 = 0.5))

# Efetuar as predições nos dados de teste
pred_rf <- predict(modelo_rf, newdata = dados_teste)
pred_svm <- predict(modelo_svm, newdata = dados_teste)
pred_neuralnet <- predict(modelo_neuralnet, newdata = dados_teste)
pred_alom <- predict(alom, newdata = dados_teste)

# Criar funções para calcular as métricas

#  Coeficiente de determinação: R2
calcular_R2 <- function(y_real, y_predito) {
  media_y_real <- mean(y_real)
  ss_tot <- sum((y_real - media_y_real)^2)
  ss_res <- sum((y_real - y_predito)^2)
  r2 <- 1 - (ss_res / ss_tot)
  return(r2)
}

#  Erro padrão da estimativa: Syx
calcular_Syx <- function(y_real, y_predito) {
  n <- length(y_real)
  syx <- sqrt(sum((y_real - y_predito)^2) / (n - 2))
  return(syx)
}

# Syx%
calcular_Syx_percent <- function(y_real, y_predito) {
  syx <- calcular_Syx(y_real, y_predito)
  media_y_real <- mean(y_real)
  syx_percent <- (syx / media_y_real) * 100
  return(syx_percent)
}

# Calculando as métricas para cada modelo
R2_rf <- calcular_R2(dados_teste$VOL, pred_rf)
Syx_rf <- calcular_Syx(dados_teste$VOL, pred_rf)
Syx_percent_rf <- calcular_Syx_percent(dados_teste$VOL, pred_rf)

R2_svm <- calcular_R2(dados_teste$VOL, pred_svm)
Syx_svm <- calcular_Syx(dados_teste$VOL, pred_svm)
Syx_percent_svm <- calcular_Syx_percent(dados_teste$VOL, pred_svm)

R2_neuralnet <- calcular_R2(dados_teste$VOL, pred_neuralnet)
Syx_neuralnet <- calcular_Syx(dados_teste$VOL, pred_neuralnet)
Syx_percent_neuralnet <- calcular_Syx_percent(dados_teste$VOL, pred_neural-
net)

R2_alom <- calcular_R2(dados_teste$VOL, pred_alom)
```

```
Syx_alom <- calcular_Syx(dados_teste$VOL, pred_alom)
Syx_percent_alom <- calcular_Syx_percent(dados_teste$VOL, pred_alom)

# Escolher o melhor modelo
# Comparando as métricas R2, Syx e Syx_percent para decidir o melhor modelo
resultados <- data.frame(Modelo = c("Random Forest", "SVM", "Redes Neu-
rais", "Alométrico de SPURR"),
                         R2 = c(R2_rf, R2_svm, R2_neuralnet, R2_alom),
                         Syx = c(Syx_rf, Syx_svm, Syx_neuralnet, Syx_alom),
                         Syx_percent = c(Syx_percent_rf, Syx_percent_svm,
Syx_percent_neuralnet, Syx_percent_alom))



# Ordena o dataframe com base na coluna "R2"
resultados_ordenados <- resultados[order(-resultados$R2), ]

#cat("Resultados dos modelos")
# Exibe os resultados ordenados
print(resultados_ordenados)
```

As tarefas de número 1 a 6 serão detalhadas em outro documento e no arquivo R. Em relação a tarefa 7, após o treinamento dos modelos Random Forest (rf), SVM (svmRadial), Redes Neurais (neuralnet) e modelo alométrico de SPURR, o modelo usando Redes Neurais foi considerado o melhor, pois foi superior aos demais em todas as métricas de comparação, pois foi o que ficou mais próximo a 1 na métrica R2 e nas métricas Syx e Syx% foi o que ficou mais próximo de zero. A escolha se baseou nos resultados da tabela abaixo que foram geradas através do script em R desenvolvido pela equipe e que será explicado no outro arquivo.

	Modelo	R2	Syx	Syx_percent
3	Redes Neurais	0.9099191	0.1208231	8.976552
4	Alométrico de SPURR	0.8694429	0.1454567	10.806705
1	Random Forest	0.8486654	0.1566040	11.634890
2	SVM	0.7899082	0.1845178	13.708744

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

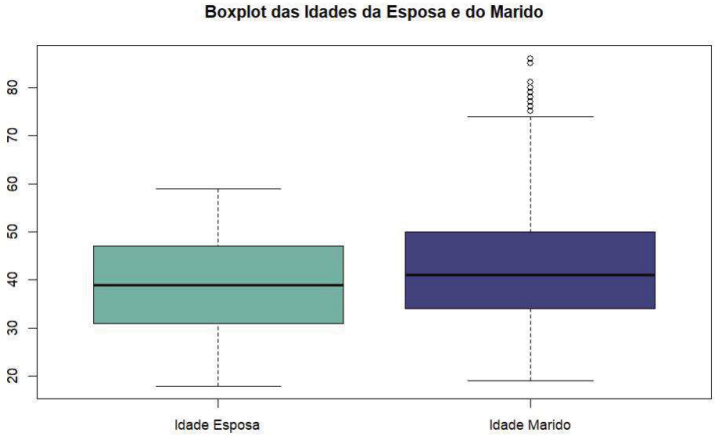
Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

B – RESOLUÇÃO

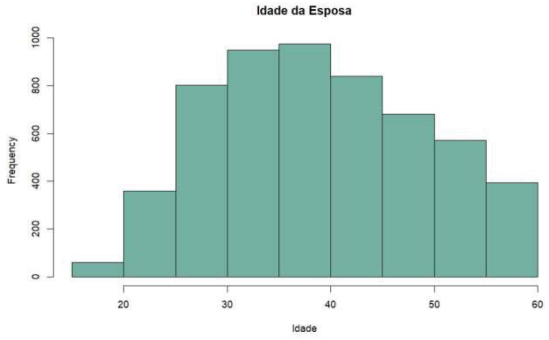
Questão 1:

a)

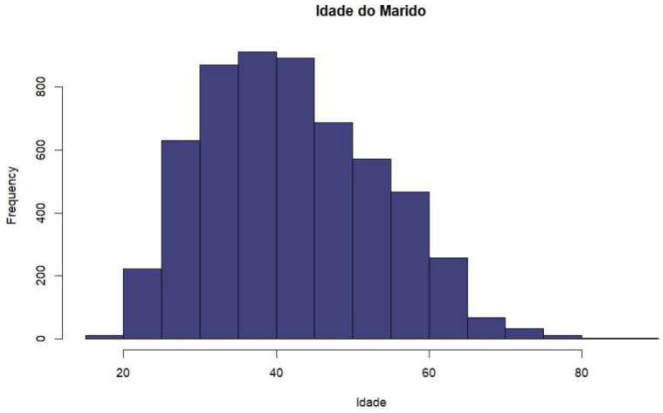
```
# Boxplot
> boxplot(salarios$age, salarios$husage,
+ names = c("Idade Esposa", "Idade Marido"),
+ col = c("blue", "red"),
+ main = "Boxplot das Idades da Esposa e do Marido")
```



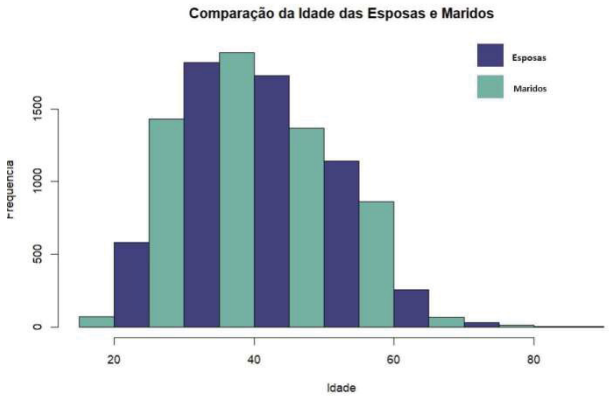

```
#Histograma 1
> hist(salarios$age,
+ main = "Idade da Esposa",
+ xlab = "Idade",
+ ylab = "Frequency",
+ col = "#69b3a2")
```




```
#Histograma 2
> hist(salarios$husage,
+ main = "Idade do Marido",
+ xlab = "Idade",
+ ylab = "Frequency",
+ col = "#404080")
```



```
# Gerando histograma comparativo
hist(c(salarios$age, salarios$husage),
main = "Comparação da Idade das Esposas e Maridos",
xlab = "Idade", ylab = "Frequência",
col = c("#69b3a2", "#404080"),
breaks = 10,
)
```



Comparação dos Histogramas

No histograma com as idades dos maridos existe uma distribuição maior dos dados, visto que existe uma maior varidade de valores observados. Os valores mais elevados podem ser consideradas outliers devido a distância que possuem da média porém a grande parte dos dados tem maior proximidade com a média.

No histograma com as idades das esposas existe uma concentração ainda maior em relação a média dos dados, além disso existem uma menor varidade de valores observados.

Comparação dos gráficos box plot.

O limite inferior do gráfico com as idades dos maridos é por volta de 20 anos e o limite superior é por volta de 80 anos. Nos valores próximos a 80 anos existem alguns outliers, que já foram identificados anteriormente na análise do histograma.

O limite superior do gráfico das esposas é menor em relação ao gráfico dos maridos, chegando apenas a 60 anos. O limite inferior também é por volta de 20 anos, ficando semelhante com o limite inferior dos maridos. Nessa amosta das idades das esposas não constam outliers.

b)

```
# Calculando a distribuição de frequência
> tabela_age <- fdt(salarios$age)

> tabela_husage <- fdt(salarios$husage)

# Mostrando a distribuicao de frequencia
```

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

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

Comparação das Tabelas de Frequência

A maior frequência das idades das esposas está entre 35,721 e 38,705 (624 ocorrências) e em seguida o intervalo entre 32,738 e35,721 (571 ocorrências). A menor ocorrência está no intervalo entre 17,82 e 20,804, contado com apenas 61 ocorrências.

Na frequência das idades dos maridos constam 917 ocorrências nos intervalos de 38,253 e 43,114, portanto esse intervalo tem a maior frequência. A menor frequência está no intervalo entre 81,999 e 86,86, contando com apenas 2 ocorrências. Outra observação são os intervalos entre 72.278 e 77.139 e entre 77.139 e 81.999 que constam com apenas 6 e 2 ocorrências respectivamente.

Esses dados citados por último se caracterizam como os outliers da amostra.

Questão 2:

a)

```
media_age <- mean(salarios$age)
mediana_age <- median(salarios$age)
moda_age <- names(sort(table(salarios$age), decreasing = TRUE)[1])
media_husage <- mean(salarios$husage)
mediana_husage <- median(salarios$husage)
moda_husage <- names(sort(table(salarios$husage), decreasing = TRUE)[1])

# Comparação dos resultados
resultadoA <- data.frame(
Variavel = c("Idade da Esposa", "Idade do Marido"),
Media = c(media_age, media_husage),
Mediana = c(mediana_age, mediana_husage),
Moda = c(moda_age, moda_husage)
)
print(resultadoA)
```

```
        Variavel    Media Mediana Moda
1 Idade da Esposa 39.42758      39   37
2 Idade do Marido 42.45296      41   44
>
```

Ao analisar as idades de esposas e maridos, percebemos que os maridos geralmente são mais velhos com idade máxima de 86 e para as esposas a idade máxima é 59.

```
> summary(salarios$age)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   31.00   39.00   39.43   47.00   59.00
> summary(salarios$husage)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  19.00   34.00   41.00   42.45   50.00   86.00
```

Com uma média de 42,45 anos para os maridos e 39,43 anos para as esposas, a diferença é clara. A mediana também suporta isso, mostrando maridos com 41 anos e esposas com 39.

Isso indica que, mesmo removendo outliers, a tendência de os maridos serem mais velhos se mantém. Além disso, a moda nos dá uma diferença maior ainda, com 44 anos para maridos e 37 para esposas, o que pode sugerir uma maior dispersão nas idades dos maridos.

Esses números mostram uma tendência consistente dos maridos serem mais velhos nas relações.

b)

```
var_age <- var(salarios$age)
desvio_padrao_age <- sd(salarios$age)
coef_variacao_age <- desvio_padrao_age / media_age * 100

var_husage <- var(salarios$husage)
desvio_padrao_husage <- sd(salarios$husage)
coef_variacao_husage <- desvio_padrao_husage / media_husage * 100

# Comparação dos resultados
resultadoB <- data.frame(
Variavel = c("Idade da Esposa", "Idade do Marido"),
Variancia = c(var_age, var_husage),
Desvio_Padrao = c(desvio_padrao_age, desvio_padrao_husage),
Coeficiente_de_Variacao = c(coef_variacao_age, coef_variacao_husage)
)
print(resultadoB)
```

```
resultadoB <- data.frame(
    Variavel = c("Idade da Esposa", "Idade do Marido"),
    Variancia = c(var_age, var_husage),
    Desvio_Padrao = c(desvio_padrao_age, desvio_padrao_husage),
    Coeficiente_de_Variacao = c(coef_variacao_age, coef_variacao_husage)
)
print(resultadoB)
        Variavel Variancia Desvio_Padrao Coeficiente_de_Variacao
Idade da Esposa  99.75234       9.98761              25.33153
Idade do Marido 126.07173      11.22817              26.44849
```

É notável que os maridos são mais velhos, mas também exibem maior variabilidade em suas idades. A análise das variâncias e desvios padrões das idades dá uma ideia de como esses valores se espalham ao redor da média, e os resultados são bem interessantes.

A variância das idades dos maridos é de 126.07, enquanto a das esposas é de 99.75. Isso mostra que as idades dos maridos estão mais espalhadas, indicando uma diversidade maior. O desvio padrão, dá uma ideia da dispersão em relação à média, segue o mesmo padrão: 11.23 para maridos contra 9.99 para esposas.

Além disso, o coeficiente de variação, nos ajuda a comparar a dispersão entre conjuntos de dados com médias diferentes. Os maridos têm um coeficiente de 26.44%, enquanto as esposas têm 25.33%. Isso significa que, relativamente, a idade dos maridos varia mais em torno da média do que a das esposas.

Esses números revelam a heterogeneidade dentro dos grupos. Uma maior variância e coeficiente de variação nos maridos sugerem que existe uma gama mais ampla de idades entre eles, talvez refletindo normas sociais ou culturais que influenciam a idade em que os homens se casam em comparação com as mulheres.

Questão 3:

a)

As informações abaixo contemplam os itens 1 e 2 do exercício.

```
# Verificar o tamanho da amostra
n <- nrow(salarios)

# Como a amostra possui mais de 5000 registros utilizaremos os testes lil-
lie.test e JarqueBeraTest para verificar a normalidade

# Testar normalidade das variáveis "age" e "husage" usando o teste de Lil-
liefors
>lillie.test(salarios$age)
```

```
> lillie.test(salarios$age)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  salarios$age
D = 0.058909, p-value < 0.00000000000000022
```

```
>lillie.test(salarios$husage)
```

```
> lillie.test(salarios$husage)

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  salarios$husage
D = 0.059662, p-value < 0.00000000000000022

>
>
```

```
# Testar normalidade das variáveis "age" e "husage" usando o teste de Jar-
que-Bera
>JarqueBeraTest(salarios$age, robust = TRUE)
>JarqueBeraTest(salarios$husage, robust = TRUE)
# Ambos deram p-value < 0.00000000000000022 então utilizaremos um teste não
paramétrico
# Mann-Whitney "U" test

# Criando uma data frame com o nome "Idade_emp"
Idade_emp <- data.frame(
Grupo = rep (c("husage ", "age"), each = nrow(salarios)),
Idade = c (salarios$age, salarios$husage)
)

# Calculando um sumário estatístico
sumario_estatistico <- Idade_emp %>%
group_by (Grupo) %>%
summarise(
count = n(),
median = median(Idade, na.rm = TRUE),
```

```
IQR = IQR(Idade, na.rm = TRUE)
)

# Vamos fazer o teste se a idade mediana dos homens é igual a
# idade mediana das mulheres

# Hipóteses do teste:

# H0: A idade mediana dos homens é igual estatisticamente a idade
# mediana das mulheres;
# Ha: A idade mediana dos homens NÃO é igual estatisticamente a idade
# mediana das mulheres;

# Teste de Wilcoxon
test_result <- wilcox.test(Idade ~ Grupo, data = Idade_emp, exact = FALSE,
conf.int = TRUE)
print(test_result)

# O p-value do teste é 0,0.00000000000000022, que é menor que o nível de
# significância 0,05. Podemos concluir que a idade mediana dos
# homens eh estatisticamente diferente da idade mediana das
# mulheres (rejeitamos H0).
# O intervalo de confiança da diferença entre as medianas esta
# entre 2.000033 e 3.000024
```

Após análise de normalidade das variáveis “husage” e “age” utilizando o teste Lilliefors e Jaquer-Bera, no qual ambos tiveram resultado de p-value < 0.00000000000000022, que é menor que o nível de significância 0,05. Podemos concluir que a idade mediana dos homens é estatisticamente diferente da idade mediana das mulheres (rejeitamos a hipótese nula de que as localizações das distribuições são iguais). Então, optou-se pela escolha do teste não paramétrico Mann-Whitney "U" test. O intervalo de confiança da diferença entre as medianas está entre 2.000033 e 3.000024, isso significa que há uma confiança de 95% de que as localizações das distribuições estão entre esse intervalo.

A estimativa da diferença nas localizações das distribuições é de 2,999966. Isso indica que a média da variável “age” (idade das esposas) é aproximadamente 3 unidades menor que a média da variável “husage” (idade dos maridos).

Por isso, infere-se que, há uma diferença estatisticamente significativa na distribuição da variável Idade entre os grupos de Maridos (“husage”) e Esposas (“age”).

Segue abaixo o resultado do teste:

```
> print(test_result)

        Wilcoxon rank sum test with continuity correction

data:  Idade by Grupo
W = 18122044, p-value < 0.00000000000000022
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 2.000033 3.000024
sample estimates:
difference in location
              2.999966
```

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

Os dados foram carregados, guardados em uma variável e separados para treino e teste. Após isso, os dados de variáveis não binárias foram padronizados para terem a mesma escala. A padronização foi aplicada nos conjuntos de treinamento e de teste.

Depois da separação em treino e teste, as variáveis categóricas foram convertidas em variáveis dummies, aplicadas nos dois conjuntos e armazenadas em 4 matrizes separadas contendo

em cada uma delas as variáveis explicativas de treinamento e de teste e a variável dependente, também de treinamento e de teste. Antes da criação dos modelos, foi efetuado o cálculo do lamba da quando necessário. Abaixo serão destacados os códigos para criação de cada um dos modelos de regressão:

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

# Modelo Lasso
lasso_reg <- glmnet(x, y_train, alpha = 1,
                    lambda = best_lambda_lasso,
                    standardize = TRUE)

# Modelo ElasticNet
elastic_reg <-
train(lwage~husage+husunion+husearns+huseduc+husblck+hushisp+hushrs+
                       kidge6+age+black+educ+hispanic+union+exper+kidlt6,
                     data = train,
                     method = "glmnet",
                     tuneLength = 10,
                     trControl = train_cont)
```

Para armazenar as métricas de performance (R2 e RMSE), foi criada uma função e a mesma foi aplicada nas predições para comparação entre os métodos usados. O exemplo abaixo mostra parte do código usado nos conjuntos de treinamento e teste na regressão Ridge, que sofreu pequenas alterações para se adequar às regressões Lasso e ElaticNet.

```
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

  # As metricas de performace do modelo:
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}

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

# As metricas da base de treinamento sao:
eval_results(y_train, predictions_train, train)

# Predicao e avaliacao nos dados de teste:
predictions_test <- predict(ridge_reg,
 s = best_lambda_ridge,
 newx = x_test)
```

```
# As metricas da base de teste sao:
eval_results(y_test, predictions_test, test)
```

Após a criação dos modelos, um mesmo conjunto de dados foi testado nos 3 modelos, permitindo a comparação entre os resultados. Adicionalmente, foram calculados os intervalos de confiança para possibilitar uma análise mais adequada dos modelos.

Análise

Foi efetuado o particionamento do dataset “trabalhosalarios.RData” em 80% dos dados para treinamento e 20% para teste.

Considerando a aplicação dos modelos Ridge, Lasso e ElasticNet, após criação de nova matriz com os valores da Tabela 1, o melhor desempenho para a estimativa do valor hora de salário para a esposa foi o do modelo Ridge, pois este modelo apresentou um RMSE (Erro Quadrático Médio) levemente inferior aos demais e também apresentou um R2 (R Quadrado) mais próximo de 1, indicando que se ajusta melhor aos dados observados.

Outra métrica considerada para a escolha do modelo Ridge foi o tempo de execução, pois o modelo Ridge levou menos tempo e ainda apresentou resultados melhores. O modelo Lasso foi levemente superior em relação aos intervalos de confiança, porém em um contexto geral o Ridge se mostrou melhor na maioria das métricas avaliadas. Os dados da tabela abaixo foram extraídos dos processamentos dos scripts em R que foram anexados no arquivo .zip da entrega do trabalho e os mesmos foram usados para a análise e escolha do melhor modelo.

Modelo	RMSE	R^2	Cllwr (Intervalo Inferior)	Clupr (Intervalo Superior)	Predição var. “lwage” (salário-hora esposa)
Ridge Time:1.37s Lambda: 0.0316	Treino: 0.8411446 Teste: 0.9893328	Treino: 0.292132 Teste: 0.259084	8.493945	8.87351	8.681654
Lasso Time: 1.55s Lambda: 0.01	Treino: 0.8420298 Teste: 0.9894877	Treino: 0.2906413 Teste: 0.258852	7.84636	8.196938	8.01971
ElasticNet Time: 7.91s Lambda: 0.0125	Treino: 0.8410462 Teste: 0.9894186	Treino: 0.2922976 Teste: 0.2589555	8.587557	8.971305	8.777334

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

Questão 1:

Para resolução do exercício, primeiramente foram fornecidas amostras de frases previamente classificadas em diferentes idiomas. Posteriormente, essas amostras foram transformadas em padrões, onde cada padrão seria um conjunto de características representado por um vetor e um conjunto de padrões no formato de tabela. Cada linha é um padrão, onde as colunas são as características e a última coluna é a classe.

Para a construção de atributos foram criadas funções para calcular a quantidade de bigramas, a quantidade de trigramas e a frequência de palavras mais comuns em cada idioma que constam no texto de entrada. Outras duas funções vão extrair as características e gerar os padrões identificados. Para a geração do modelo, os padrões serão submetidos a um treinamento usando SVM e este modelo será usado para a identificação do idioma.

Abaixo consta a parte final do código usado para identificar o idioma de um texto. Após o código, consta a saída exibida pelo código após receber a frase de entrada “trabalho de arquitetura de dados”.

```
def identifica_idioma(texto):
"""
Identifica o idioma de um texto dado. Se o texto tiver mais de 500 caracte-
res, será truncado.
"""
if len(texto) > 500:
texto = texto[:500]

# Extrai características do texto
caracteristicas = extraiCaracteristicas([texto, 'desconhecido'])[:-1]  #
Remove a classe ('desconhecido')

# Prediz o idioma usando o modelo treinado
predicao = modelo.predict([caracteristicas])[0]

return predicao

# Solicitar ao usuário a entrada de um texto
texto_usuario = input("Por favor, insira um texto (máximo de 500 caracte-
res): ")

# Identificar o idioma do texto
idioma_identificado = identifica_idioma(texto_usuario)

# Exibir o resultado
if idioma_identificado in ['inglês', 'espanhol', 'português']:
print(f"O seu texto está escrito no idioma {idioma_identificado}.")
else:
print("Não foi possível identificar o idioma.")

def avalia_acuracia_identificacao(textos, idiomas_reais):
"""
Avalia a acurácia da identificação de cada idioma.
"""
predicoes = [identifica_idioma(texto) for texto in textos]
```

```
acuracia = accuracy_score(idiomas_reais, predicoes)
print("Acurácia geral: {:.2f}%".format(acuracia * 100))
cm = confusion_matrix(idiomas_reais, predicoes, labels=['inglês', 'espa-
nhol', 'português'])
print("Matriz de Confusão:")
print(cm)
print("Relatório de Classificação:")
print(classification_report(idiomas_reais, predicoes, labels=['inglês',
'espanhol', 'português']))

# Textos de teste e seus idiomas reais
textos_teste = [
"I love reading books.", "Me gusta bailar salsa.", "Adoro praticar espor-
tes.",
"The cat is sleeping.", "¿Dónde está el restaurante?", "A festa começa às
20h.",
"She sells seashells by the seashore.", "El perro ladra fuerte.", "O céu
está azul hoje.",
"Can you help me with this?", "Estoy muy cansado hoy.", "Eles foram ao mer-
cado.",
"We need to finish this project by tomorrow.", "Necesito una taza de ca-
fé.", "Eu vou viajar amanhã."
]
idiomas_reais = ['inglês', 'espanhol', 'português', 'inglês', 'espanhol',
'português',
'inglês', 'espanhol', 'português', 'inglês', 'espanhol', 'português',
'inglês', 'espanhol', 'português']

# Avaliar a acurácia da identificação
avalia_acuracia_identificacao(textos_teste, idiomas_reais)re_padroes = []
for frase in ingles:
pre_padroes.append( [frase, 'inglês'])

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

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

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

```
Por favor, insira um texto (máximo de 500 caracteres): trabalho de arquitetura de dados
O seu texto está escrito no idioma português.
Acurácia geral: 80.00%
Matriz de Confusão:
[[4 1 0]
 [0 5 0]
 [0 2 3]]
Relatório de Classificação:
              precision    recall  f1-score   support

      inglês       1.00      0.80      0.89         5
    espanhol       0.62      1.00      0.77         5
   português       1.00      0.60      0.75         5

    accuracy                           0.80        15
   macro avg       0.88      0.80      0.80        15
weighted avg       0.88      0.80      0.80        15
```

Questão 2:

A base usada no trabalho foi a base do Titanic disponível na biblioteca Seaborn do Python. Após a identificação das features e do target, as colunas numéricas foram transformadas em variáveis categóricas usando LabelEncoder. Nos próximos passos, as linhas com dados faltantes foram removidas, os dados foram divididos para os conjuntos de treinamento e teste e o treinamento foi feito usando SVM. Os códigos estão detalhados abaixo:

```
# Itera sobre todas as colunas não numéricas e aplica o LabelEncoder
for column in titanic.select_dtypes(include=['object', 'bool',
'category']).columns:
    label_encoder = LabelEncoder()
    titanic[column] = label_encoder.fit_transform(titanic[column])

# Remover linhas com valores ausentes do DataFrame titanic
titanic.dropna(inplace=True)

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

# Inicializar e treinar o classificador SVM
svm_classifier = SVC()
svm_classifier.fit(X_train, y_train)

# Fazer previsões no conjunto de teste
y_pred = svm_classifier.predict(X_test)


# Calcular a acurácia do modelo em porcentagem
accuracy_percentage = accuracy_score(y_test, y_pred) * 100
print("Acurácia do modelo:", accuracy_percentage, "%")

Acurácia do modelo: 62.93706293706294 %
```

Após o primeiro processamento dos dados, foram aplicadas técnicas de agrupamento, codificação e construção de atributos derivados, além do cálculo da prevalência que resultou em cerca de 40,62%, considerado um valor razoavelmente balanceado. Antes de aplicar novo treinamento, foram selecionadas somente as variáveis mais relevantes. Após o novo treinamento na base melhorada, a acurácia chegou a 79,72%, conforme os códigos abaixo:

```
# Calcular a acurácia do novo modelo em porcentagem
accuracy_percentage = accuracy_score(y_test, y_pred) * 100
print("Acurácia do modelo:", accuracy_percentage, "%")

Acurácia do modelo: 79.72027972027972 %
```

APÊNDICE 7 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

CLASSIFICAÇÃO

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – Hold-out	C=1 Sigma=0.06	0.784	Confusion Matrix and Statistics Reference Prediction bus opel saab van bus 41 2 1 0 opel 0 20 9 0 saab 0 19 32 1 van 2 1 1 38
RNA – CV	size=5decay=0.1	0.7545	Reference Prediction bus opel saab van bus 38 0 1 0 opel 1 26 17 1 saab 4 16 25 1 van 0 0 0 37
RF – CV	mtry=2	0.7485	Confusion Matrix and Statistics Reference Prediction bus opel saab van bus 42 3 0 0 opel 0 21 15 0 saab 0 18 25 2 van 1 0 3 37
RF – Hold-out	mtry=2	0.7365	Confusion Matrix and Statistics Reference Prediction bus opel saab van bus 42 3 1 0 opel 0 20 15 0 saab 0 19 24 2 van 1 0 3 37
SVM – CV	C=0.5 Sigma=0.06	0.713	Confusion Matrix and Statistics Reference Prediction bus opel saab van bus 40 2 1 0 opel 0 12 10 0 saab 0 27 30 2 van 3 1 2 37
KNN	k=1	0.647	Confusion Matrix and Statistics Reference Prediction bus opel saab van bus 39 2 4 4 opel 2 18 15 0 saab 3 18 19 0 van 2 5 5 34
RNA – Hold-out	size=3decay=0.1	0.479	Reference Prediction bus opel saab van bus 42 41 42 1 opel 0 0 0 0 saab 0 0 0 0 van 1 1 1 38

Após cálculos

Melhor técnica:SVM – Hold-out

3 Casos Desconhecidos:

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis
1	95	48	83	178	72	10	162	42	20	100	176	379
2	100	41	84	141	57	9	149	45	19	143	170	330
3	104	50	100	209	66	10	300	32	23	158	223	635

RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
184	70	6	16	187	200	?
158	72	9	14	189	199	?
220	73	14	9	188	196	?

Resultados:

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis
1	95	48	83	178	72	10	162	42	20	100	176	379
2	100	41	84	141	57	9	149	45	19	143	170	330
3	104	50	100	209	66	10	300	32	23	158	223	635

RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	predict.svm
184	70	6	16	187	200	opel
158	72	9	14	189	199	van
220	73	14	9	188	196	opel

Lista de comandos do RStudio:

```
### Criar bases de Treino e Teste
set.seed(202470)
indices<- createDataPartition(dados$tipo, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinar SVM com a base de Treino
set.seed(202470)
svm<- train(tipo~., data=treino, method="svmRadial")
svm

### Aplicar modelos treinados na base de Teste
predict.svm<- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))
###  Aplicar modelos treinados na base de Teste
predict.svm<- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

#### Cross-validation SVM
ctrl<- trainControl(method = "cv", number = 10)
set.seed(202470)
svm<- train(tipo~., data=treino, method="svmRadial", trControl=ctrl)
svm

### Matriz de confusão
predict.svm<- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))
```

```
#### Vários C e sigma
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))

set.seed(202470)
svm<- train(tipo~., data=treino, method="svmRadial", trControl=ctrl, tune-
Grid=tuneGrid)
svm

### Matriz de confusão
predict.svm<- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos<- read.csv("6 - Veiculos - Novos Casos.csv")
dados_novos_casos$a<- NULL
View(dados_novos_casos)

predict.svm<- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
resultado$tipo<- NULL
View(resultado)
```

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – CV	mtry=2	0.7451	Reference Prediction neg pos neg 82 21 pos 18 32
SVM – Hold-out	C=0.25 Sigma=0.13	0.739	Confusion Matrix and Statistics Reference Prediction neg pos neg 87 27 pos 13 26
RF – Hold-out	mtry=2	0.7386	Reference Prediction neg pos neg 83 23 pos 17 30
SVM – Hold-out	C=0.25 Sigma=0.13	0.732	Confusion Matrix and Statistics Reference Prediction neg pos neg 84 25 pos 13 26
SVM – CV	C=0.25 Sigma=0.13	0.732	Confusion Matrix and Statistics Reference Prediction neg pos neg 84 25 pos 16 28
RNA – Hold-out	size=1 decay=0.1	0.7255	Reference Prediction neg pos neg 81 23 pos 19 30
KNN	k=9	0.721	Confusion Matrix and Statistics Reference Prediction neg pos neg 89 27 pos 16 22

Após os cálculos

Melhor técnica: RF – CV

3 Casos Desconhecidos:

	num	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
1	1	10	200	74	0	0	38.0	0.937	34	?
2	2	10	200	80	0	0	27.1	1.441	57	?
3	3	1	60	60	23	300	30.1	0.396	59	?

Resultados:

	num	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes	predict.rf_cv
1	1	10	200	74	0	0	38.0	0.937	34	?	pos
2	2	10	200	80	0	0	27.1	1.441	57	?	neg
3	3	1	60	60	23	300	30.1	0.396	59	?	neg

Lista de comandos do RStudio:

```
#Diabetes

### Cria um arquivo com treino com 80% e teste com 20% das linhas de forma
randomizada
set.seed(202470)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino<- dados[indices,]
teste <- dados[-indices,]

### Gerar um novo modelousandoRandonForest, predições e matriz de confusão
set.seed(202470)
rf <- train(diabetes~., data=treino, method="rf")
rf

### Predições com o arquivo de teste
predicoes.rf<- predict(rf, teste)
confusionMatrix(predicoes.rf,as.factor(teste$diabetes))

#### Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202470)
rf_cv<- train(diabetes~., data=treino, method="rf", trControl=ctrl)
rf_cv

### Matriz de confusao
predict.rf_cv<- predict(rf_cv, teste)
confusionMatrix(predict.rf_cv, as.factor(teste$diabetes))

### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos<- read.csv("10 - Diabetes - Novos Casos.csv")
View(dados_novos_casos)

dados_novos_casos$Volume<- NULL
predict.rf_cv<- predict(rf_cv, dados_novos_casos)
resultado<- cbind(dados_novos_casos, predict.rf_cv)
View(resultado)
```

REGRESSÃO

Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – Hold-out	C=1 Sigma=0.11	0.8492141	0.05683194	0.9285196	0.05654124	0.03801399
SVM – CV	C=1 Sigma=0.11	0.8492141	0.05683194	0.9285196	0.05654124	0.03801399
RF – Hold-out	mtry=2	0.8206227	0.06230834	0.9065653	0.06166926	0.04253831
RF – CV	mtry=2	0.8153338	0.06322024	0.9035653	0.06257181	0.04345472
RNA – Hold-out	size=3 decay=0.1	0.8139129	0.06346299	0.9068822	0.06281207	0.04790741
RNA – CV	size=5 decay=0.1	0.8107109	0.06400666	0.9025809	0.06335017	0.04759592
KNN	K=7	0.089	0.09	0.8	0.62	0.068

Após os cálculos

Melhor técnica: SVM – Hold-out

3 Casos Desconhecidos:

	num	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	ChanceOfAdmit
1	1	412	99	3	4	2	7.9	1	?
2	2	324	114	4	4	2	10.4	0	?
3	3	358	113	5	5	9	9.5	1	?

Resultados:

	num	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predict.svm
1	1	412	99	3	4	2	7.9	1	0.6910434
2	2	324	114	4	4	2	10.4	0	0.7658169
3	3	358	113	5	5	9	9.5	1	0.7104633

Lista de comandos do RStudio:

```
### Criar bases de Treino e Teste
set.seed(202470)
indices <- createDataPartition(dados$ChanceOfAdmit, p=0.80,list=FALSE)
treino<- dados[indices,]
teste <- dados[-indices,]

### Treinar SVM com a base de Treino
set.seed(202470)
svm<- train(ChanceOfAdmit~., data=treino, method="svmRadial")
svm

### Aplicarmodelostreinadosna base de Teste
predicoes.svm<- predict(svm, teste)

### Calcular as métricas
rmse(teste$ChanceOfAdmit, predicoes.svm)
```

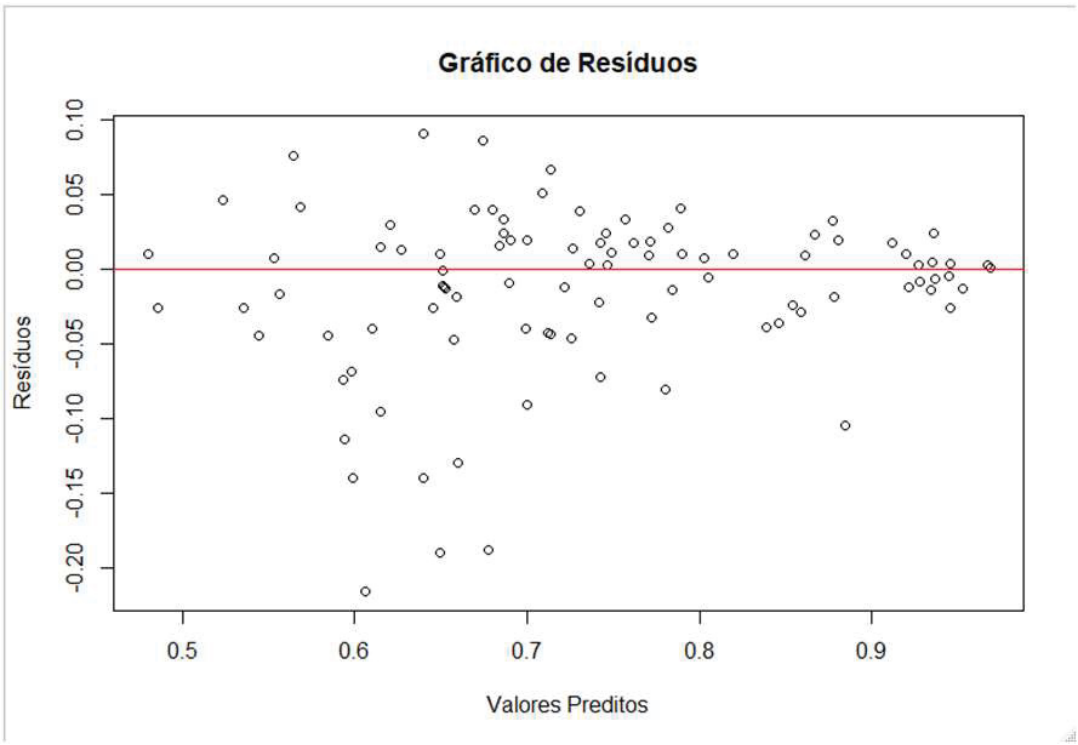
```
##calculo do r2
r2 <- function(predito, observado) {
return(1 - (sum((predito-observado)^2) / sum((observado-
mean(observado))^2)))
}
r2(predicoes.svm,teste$ChanceOfAdmit)

## Cálculo de syx
syx<- function(observado, predito) {
n <- length(observado)
p <- 1
sqrt(sum((observado - predito)^2) / (n - p))
}
syx(teste$ChanceOfAdmit, predicoes.svm)


## Cálculo do coeficiente de Pearson
pearson<- function(observado, predito) {
cor(observado, predito)
}
pearson(teste$ChanceOfAdmit, predicoes.svm)

## Cálculo do MAE
mae<- function(observado, predito) {
mean(abs(observado - predito))
}
mae(teste$ChanceOfAdmit, predicoes.svm)


#### Cross-validation SVM
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202470)
svm<- train(ChanceOfAdmit~., data=treino, method="svmRadial",
trControl=ctrl)
svm

### Aplicarmodelostreinadosna base de Teste
predicoes.svm<- predict(svm, teste)

### Calcular as métricas
rmse(teste$ChanceOfAdmit, predicoes.svm)

##calculo do r2
r2 <- function(predito, observado) {
return(1 - (sum((predito-observado)^2) / sum((observado-
mean(observado))^2)))
}
r2(predicoes.svm ,teste$ChanceOfAdmit)

### Calcular as métricas
syx(teste$ChanceOfAdmit, predicoes.svm)
pearson(teste$ChanceOfAdmit, predicoes.svm)
mae(teste$ChanceOfAdmit, predicoes.svm)


### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos<- read.csv("9 - Admissao - Novos Casos.csv")
View(dados_novos_casos)

dados_novos_casos$ChanceOfAdmit<- NULL
predict.svm<- predict(svm, dados_novos_casos)
resultado<- cbind(dados_novos_casos, predict.svm)
```

```
View(resultado)
```

Gráfico de Resíduos

Lista de comandos do RStudio:

```
### Gráfico de Resíduos
residuos <- teste$ChanceOfAdmit - predicoes.svm
plot(predicoes.svm, residuos, main="Gráfico de Resíduos", xlab="Valores Preditos", ylab="Resíduos")
abline(h=0, col="red")
```



Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0.8709691	532.4229	0.9797789	523.474	157.3707
RF – CV	mtry=2	0.8679476	538.6207	0.9805176	529.5676	165.2688
KNN	K=1	0.67	850	0.91	836	232
SVM – Hold-out	C=1 Sigma=0.91	0.34	1205	0.73	1184	362
SVM – CV	C=1 Sigma=0.91	0.34	1205	0.73	1184	362
RNA – Hold-out	size=3 decay=0.1	0.1884613	1335.256	0.8232358	1312.813	570.528
RNA – CV	size=3 decay=0.4	-0.09940084	1554.132	0.8481104	1528.01	502.1445

Após os cálculos

Melhor técnica: RF – Hold-out

3 Casos Desconhecidos:

	dap	h	Me	biomassa
1	11.3	10.7	0.9	?
2	10.3	13.0	0.9	?
3	17.5	11.0	0.5	?

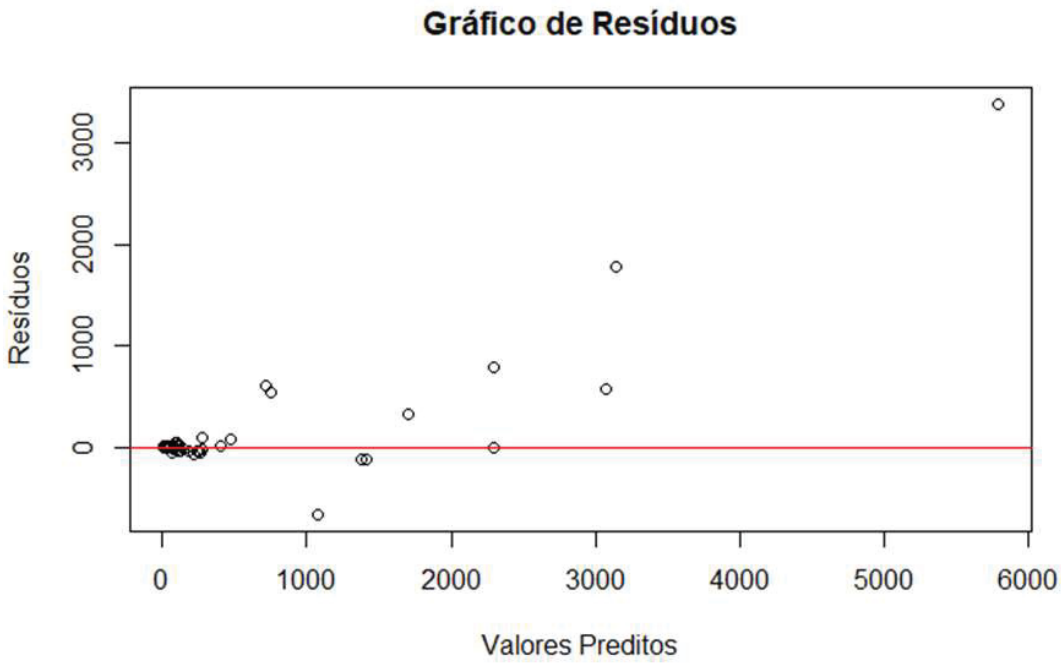
Resultados:

	dap	h	Me	biomassa	predict.rf
1	11.3	10.7	0.9	?	77.56292
2	10.3	13.0	0.9	?	52.75713
3	17.5	11.0	0.5	?	89.92782

Lista de comandos do RStudio:

```
#######################################
#REGRESSÃO - RF
#######################################
##2. Biomassa

## Cria arquivos de treino e teste
set.seed(202470)
indices<- createDataPartition(dados$biomassa, p=0.80, list = FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinar Randon Forest com a base de Treino
set.seed(202470)
rf<- train(biomassa~., data=treino, method="rf")
rf

### Aplicar modelos treinados na base de Teste
predicoes.rf<- predict(rf, teste)
predicoes.rf

### Calculo rmse
rmse(teste$biomassa, predicoes.rf)

#Função para calcular o R2
r2 <- function(predito, observado) {
return(1 - (sum((predito-observado)^2) / sum((observado-
mean(observado))^2)))
}
r2(predicoes.rf,teste$biomassa)

# Função para calcular o syx (erro padrão da estimativa)
syx<- function(observado, predito) {
sqrt(sum((observado - predito)^2) / (length(observado) - 2))
}
syx(teste$biomassa, predicoes.rf)
```

```
# Função para calcular o coeficiente de Pearson
pearson<- function(observado, predito) {
cor(observado, predito)
}
pearson(teste$biomassa, predicoes.rf)

# Função para calcular o MAE (Mean Absolute Error)
mae<- function(observado, predito) {
mean(abs(observado - predito))
}
mae(teste$biomassa, predicoes.rf)

#### Cross-validation
ctrl<- trainControl(method = "cv", number = 10)
set.seed(202470)
rf_cv<- train(biomassa~., data=treino, method="rf", trControl=ctrl)
rf_cv
predict.rf_cv<- predict(rf_cv, teste)


### Calculo rmse
rmse(teste$biomassa, predict.rf_cv)

#Função para calcular o R2
r2 <- function(predito, observado) {
return(1 - (sum((predito-observado)^2) / sum((observado-
mean(observado))^2)))
}
r2(predict.rf_cv,teste$biomassa)


# Função para calcular o syx (erro padrão da estimativa)
syx<- function(observado, predito) {
sqrt(sum((observado - predito)^2) / (length(observado) - 2))
}
syx(teste$biomassa, predict.rf_cv)

# Função para calcular o coeficiente de Pearson
pearson<- function(observado, predito) {
cor(observado, predito)
}
pearson(teste$biomassa, predict.rf_cv)


# Função para calcular o MAE (Mean Absolute Error)
mae<- function(observado, predito) {
mean(abs(observado - predito))
}
mae(teste$biomassa, predict.rf_cv)


### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos<- read.csv("5 - Biomassa - Novos Casos.csv")
View(dados_novos_casos)


dados_novos_casos$Volume<- NULL
predict.rf<- predict(rf, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.rf)
View(resultado)
```



AGRUPAMENTO

VEÍCULO

Lista completa dos comandos emitidos no Rstudio:

```
### Pacotes necessários
#install.packages("mlbench")
library(mlbench)

#install.packages("mice")
library(mice)

## para o kmodes
#install.packages("klaR")
library(klaR)

### Leitura dos dados
setwd("C:\\iaa\\Trabalho")
dados <- read.csv("4 - Veiculos - Dados.csv")
View(dados)

# Remoção da coluna desnecessária (id)
dados$a <- NULL

## Executa o cluster - Usar 10 Clusters
set.seed(202470)
cluster.results <- kmodes(dados, 10, weighted = FALSE )
cluster.results

### Resultado do agrupamento
resultado <- cbind(dados, cluster.results$cluster)

### Exibição do resultado
head(resultado, 10)
```

As saídas dos principais comandos serão mostradas abaixo.

Após a leitura dos dados e remoção da coluna de ID, é lançada a seed e depois o comando com 10 clusters é executado:

```
## Executa o cluster - Usar 10 Clusters
set.seed(202470)
cluster.results <- kmodes(dados, 10, weighted = FALSE )
cluster.results
```

O resultado é este abaixo, onde consta a Lista de Clusters gerados:

```
K-modes clustering with 10 clusters of sizes 63, 109, 84, 79, 75, 57, 93, 97, 76, 113

Cluster modes:
    a Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis ScVarmaxis RaGyr SkewMaxis Skewmaxis KurtMaxis KurtMaxis HollRa Class
1   8   89   36    69   127       64      8    140    48         18       143        161        281   123        66         7         6       193      201  opel
2   1  101   53   108   203       63     10    217    31         24       162        226        712   214        70         0        24       189      197  saab
3   7   86   42    71   141       54      7    153    44         19       144        175        341   172        69         3         7       179      182   bus
4   9   85   37    66   121       55      6    122    57         17       128        137        209   127        64         4        14       180      183   van
5   2   90   46    85   169       56     11    157    43         20       160        170        373   185        74         1        11       183      192   van
6   3   89   45    82   136       63     10    155    43         19       148        173        351   174        71         1         9       188      196   van
7  15  104   55   105   197       61     11    218    31         24       173        221        706   216        71         0         4       187      196  opel
8  16  100   52   100   197       62     10    201    33         23       161        214        602   178        67         0        11       192      198  opel
9  27   91   38    66   130       59      7    128    52         18       127        148        246   139        72         1        20       183      187  saab
10  5   85   43    68   120       56      7    150    45         19       145        169        330   171        85         4         1       181      183   bus

Clustering vector:
  [1]  2  5  6  6 10 10  3  1  4  4  1  4  5  6  7  8  4 10  7  2  3  4 10  8  7  6  9  7  3  9  6  1  4  2  7  5  3  3  2  3  8  4  5  2  2  5  4  3  3  6  9  2  7  3  4  1  2  5  7
 [60]  9  8  9  9 10 10  7 10  7  5 10  2  8  8  4  3  2  9 10  7 10  3  2  1 10  3 10 10  4  3  4  7  8  8  5  3  2  3 10  8  3 10  3  9  9  6  2  7 10 10 10  6  6  9  4  6  5  8  2
[119] 10  2  9  5  6  4  6  5 10 10  6  8  2 10  9  4  7  5  5  1  9  3  5  3  8 10  8  7  5 10  3  1  2 10 10  2  6 10  4  5  6  1  9  8  2  9  2  7  3  2 10  5  7 10  6  7  5  5  7
[178] 10  3  4  2  1  3  9  8  2  3  6  2  1  8  2  2  9  5 10  2  3  3  6  5  1  7  7  4  6  8  3  5  7  8  2  3  7  1  5  9  7 10  4  7  5  8  3  9  6  7  1  2 10  4  4 10  8  1  5
[237]  4 10  8  3  1  1  3  2  5  1 10  8  7  4  9  7  5 10  6  7  1  3  2  7  9  2  4  4  8 10  3  5  4  4 10  9 10  6 10 10 10  4  2 10  8  3 10  2  5  4  3  6  8  4  9  5  3  1  4
[296]  9  8  9  9  3  2 10  4  1 10 10  2  2  8  8  2  6 10  8 10  9  9  6  4  3  2 10 10  2  9  3  8  6 10  9  6  8  7  3  5  7  5  8  6  3 10  1  4  8  7  2  2  3  6  1  1 10  5  3
[355]  2  2  3  2  6  8  2  8  5  3  3  8  6  2 10  7  8  3  6  4  4  8  2  9  9  7  1  8  3  2  1  5  2  3  6  8 10  8  3  6  2  5 10  3  3  1  2  3  2  1  8  3  9  9  3  9  1  4 10
[414]  1  5  2  1  3 10  9  9  7  9 10  1  1  2  9  7  1  2  8  9 10  2  4  3 10  8  9  8  5  4  1  2  5  8  2  2  6 10  5  8 10  8 10  4  6  7  3  9  8  8  4 10 10 10  7  2  9  9  7
[473]  3  7  3  7  7  5 10  3  7  7  8  9  5  3  7  2  3  2  2  5 10  7  3  3  7  6  4  2  8  7  9  9  2  2  2 10  5  7 10  8  7  5  9  9  3  2  9  1  2  4  1 10  7  9  9  8  5  3  8
[532]  3  7  4  5  6  7  4  8  4  1  9  9  2  3  5  4  1  3  2  4  1  2  3  8  8 10  9 10  2  7  5  7  3  6  2  2  9  3  9  2  8  6  5  7  2  2  8  5  9 10  4  2  7  8  4  7  9  7  7
[591] 10  1  3  6  9  7  7  4 10  9  5  4  8  1  8  2  4  4  1 10  7  5  4  4  4  1  1  4  9  7  2  6  8  2  1  2  1  3  8  9  2  7 10  8  9  1  9 10 10  7  8 10  2 10  6  7  4  9  7
[650]  6  7  6  8  8  2  4  3  5  1  2  9  8 10  7  4  5  8  2  5 10  8  7  8 10  6  8 10  8 10  5  5 10  5 10  7  3  1  4  4  7  2 10  8  5  8  1  6  4  3  5  7  8  1  1  8  7  1  4
[709]  5  8  4  8  8  2  3  8  9  5  8  7  7  2  5  7  6  4  8 10  1  6  7  5  6  4  4  7  2  2  2  6  2 10  1  2  6  3  9 10  5  2  2  4  9 10  1  4  2 10 10  6  8  2  7  7  2 10  9
[768]  3  8  1 10  8  8 10  4  5  2 10  7  9  8  8  8  7  4 10  7  4  2  2 10  1  8  2 10  4  4  7  7 10  6  9  1  3  8 10  6  7  3  4  7  1 10  5 10 10  5  2  2  2  9  7  4  8  7  5
[827]  8  8  3  5 10  3  7  5  9 10  6  5  6  1  1  9  5  7  1  4

Within cluster simple-matching distance by cluster:
 [1] 1062 1785 1400 1360 1204  934 1486 1603 1295 1873

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

Os comandos abaixo combinam os dados originais com os resultados encontrados e depois é feita a exibição das 10 primeiras linhas do arquivo e qual o cluster correspondente de cada linha:

```
### Resultado do agrupamento
resultado <- cbind(dados, cluster.results$cluster)

### Exibição do resultado
head(resultado, 10)
```

Exibição gerada pelo comando acima, com o cluster de cada linha na última coluna a direita:

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

REGRAS DE ASSOCIAÇÃO

Musculação

Lista completa dos comandos emitidos no RStudio

```
### Instalação dos pacotes necessários
#install.packages('arules', dep=T)
library(arules)
library(datasets)

### Leitura dos dados
setwd("C:\\faa\\Trabalho")
dados <- read.transactions(file="2 - Musculacao - Dados.csv",format="basket",sep=";")

# Exibição dos 5 primeiros itens
inspect(dados[1:5])

# Gráfico dos 10 primeiros itens mais frequentes
itemFrequencyPlot(dados, topN=10, type='absolute')

# Visão geral dos dados
summary(dados)

### EXTRAÇÃO DAS REGRAS
# TESTES EFETUADOS COM SUPORTE ENTRE 0,1 A 0,5 E CONFIANÇA ENTRE 0,5 E 0,9
# A ESCOLHA FINAL FOI SUPORTE = 0,3 E CONFIANÇA = 0,7
# FORAM GERADAS 18 REGRAS

set.seed(202470)
rules <- apriori(dados, parameter = list(supp=0.3, conf = 0.7, target = "rules")) # > 18 regras

# visão geral das regras
summary(rules)

# EXIBIÇÃO DAS REGRAS POR ORDEM DE CONFIANÇA
inspect(sort(rules,by="confidence"))
```

As saídas dos principais comandos serão mostradas abaixo.

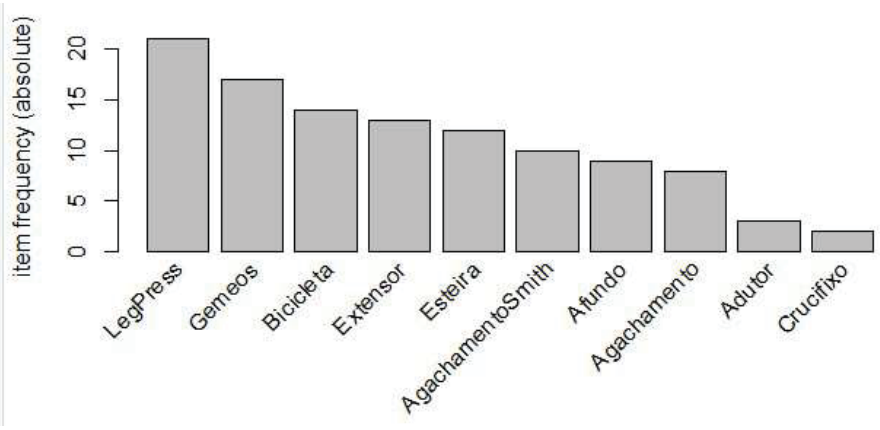
Exibição dos primeiros itens:

```
> # Exibição dos 5 primeiros itens
> inspect(dados[1:5])
    items
[1] {Afundo, Crucifixo, Gemeos, LegPress}
[2] {Agachamento, Gemeos, LegPress}
[3] {Afundo, Agachamento, Gemeos, LegPress}
[4] {Adutor, Agachamento, LegPress}
[5] {Afundo, Bicicleta, Gemeos, LegPress}
```

O comando abaixo gera o gráfico apresentando na sequência do documento:

```
# Gráficos dos 10 primeiros itens mais frequentes
itemFrequencyPlot(dados, topN=10, type='absolute')
```

Exibição do gráfico gerado com o comando anterior:



A execução do código com o apriori exibe o resultado abaixo:

```
> set.seed(202470)
> rules <- apriori(dados, parameter = list(supp=0.3, conf = 0.7, target = "rules")) # > 18 regras
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.7    0.1    1 none FALSE            TRUE       5     0.3      1
 maxlen target  ext
     10  rules TRUE

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

Absolute minimum support count: 7

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[11 item(s), 26 transaction(s)] done [0.00s].
sorting and recoding items ... [8 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [18 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Comando para exibir uma visão geral das regras encontradas:

```
> summary(rules)
set of 18 rules

rule length distribution (lhs + rhs):sizes
 1  2  3
 1 12  5

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   2.000   2.222   2.750   3.000

summary of quality measures:
    support         confidence        coverage          lift            count
 Min.   :0.3077   Min.   :0.7059   Min.   :0.3077   Min.   :0.8739   Min.   : 8.00
 1st Qu.:0.3173   1st Qu.:0.8141   1st Qu.:0.3846   1st Qu.:1.5340   1st Qu.: 8.25
 Median :0.3846   Median :0.8730   Median :0.4423   Median :1.7013   Median :10.00
 Mean   :0.3996   Mean   :0.8742   Mean   :0.4658   Mean   :1.6109   Mean   :10.39
 3rd Qu.:0.4231   3rd Qu.:0.9215   3rd Qu.:0.5000   3rd Qu.:1.8042   3rd Qu.:11.00
 Max.   :0.8077   Max.   :1.0000   Max.   :1.0000   Max.   :2.0000   Max.   :21.00

mining info:
  data ntransactions support confidence                                                                        call
 dados            26     0.3        0.7 apriori(data = dados, parameter = list(supp = 0.3, conf = 0.7, target = "rules"))
```

Exibição das regras encontradas por ordem de confiança:

```
> # EXIBIÇÃO DAS REGRAS POR ORDEM DE CONFIANÇA
> inspect(sort(rules,by="confidence"))
     lhs                                rhs          support   confidence coverage  lift      count
[1]  {Agachamento}                   => {LegPress}   0.3076923 1.0000000  0.3076923 1.2380952  8
[2]  {Afundo}                        => {Gemeos}     0.3461538 1.0000000  0.3461538 1.5294118  9
[3]  {AgachamentoSmith, Bicicleta}   => {Extensor}   0.3076923 1.0000000  0.3076923 2.0000000  8
[4]  {Bicicleta, Esteira}            => {Extensor}   0.3846154 1.0000000  0.3846154 2.0000000 10
[5]  {Extensor}                      => {Bicicleta}  0.4615385 0.9230769  0.5000000 1.7142857 12
[6]  {Esteira}                       => {Extensor}   0.4230769 0.9166667  0.4615385 1.8333333 11
[7]  {Esteira, Extensor}             => {Bicicleta}  0.3846154 0.9090909  0.4230769 1.6883117 10
[8]  {AgachamentoSmith}              => {Extensor}   0.3461538 0.9000000  0.3846154 1.8000000  9
[9]  {AgachamentoSmith, Extensor}    => {Bicicleta}  0.3076923 0.8888889  0.3461538 1.6507937  8
[10] {Bicicleta}                     => {Extensor}   0.4615385 0.8571429  0.5384615 1.7142857 12
[11] {Extensor}                      => {Esteira}    0.4230769 0.8461538  0.5000000 1.8333333 11
[12] {Esteira}                       => {Bicicleta}  0.3846154 0.8333333  0.4615385 1.5476190 10
[13] {Bicicleta, Extensor}           => {Esteira}    0.3846154 0.8333333  0.4615385 1.8055556 10
[14] {}                              => {LegPress}   0.8076923 0.8076923  1.0000000 1.0000000 21
[15] {AgachamentoSmith}              => {Esteira}    0.3076923 0.8000000  0.3846154 1.7333333  8
[16] {AgachamentoSmith}              => {Bicicleta}  0.3076923 0.8000000  0.3846154 1.4857143  8
[17] {Bicicleta}                     => {Esteira}    0.3846154 0.7142857  0.5384615 1.5476190 10
[18] {Gemeos}                        => {LegPress}   0.4615385 0.7058824  0.6538462 0.8739496 12
```

APÊNDICE 8 – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

Abaixo serão exibidos alguns dos principais trechos de código usados na resolução dos exercícios.

Classificação de Imagens (CNN)

```
# Carga da base
cifar10 = tf.keras.datasets.cifar10
# Já está separado em dados de treino e teste
# Não precisa separar
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalização os dados

# Imagens em pixels de 0 - 255
# / 255.0 transforma em 0 - 1
x_train, x_test = x_train / 255.0, x_test / 255.0
# O dado y é a classe a qual faz parte
# O flattem torna os dados vetorizados
y_train, y_test = y_train.flatten(), y_test.flatten()
```

```
# Dimensão dos dados
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)

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

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

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

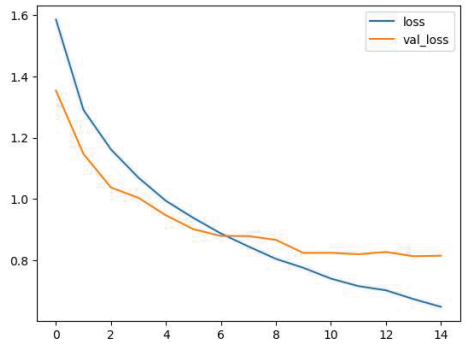
Model: "functional_1"

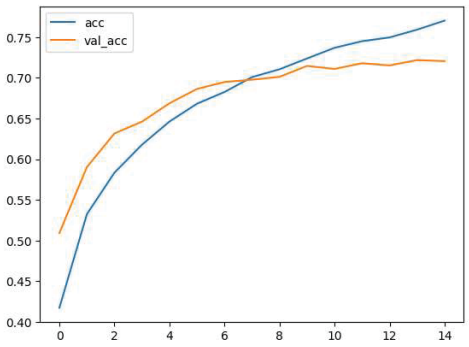
Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 32, 32, 3)	0
conv2d_3 (Conv2D)	(None, 15, 15, 32)	896
conv2d_4 (Conv2D)	(None, 7, 7, 64)	18,496
conv2d_5 (Conv2D)	(None, 3, 3, 128)	73,856
flatten_1 (Flatten)	(None, 1152)	0
dropout_2 (Dropout)	(None, 1152)	0
dense_2 (Dense)	(None, 1024)	1,180,672
dropout_3 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 10)	10,250

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

```
# Compilar o modelo
model.compile(optimizer="adam",
loss="sparse_categorical_crossentropy", metrics=["accuracy"])
# Treinar o modelo
r = model.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=15)

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

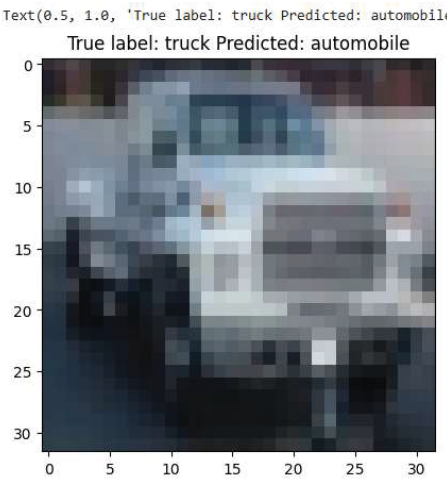
```
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
plt.show()
```






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

```
Text(0.5, 1.0, 'True label: truck Predicted: automobile')
```



Classificação de Imagens (CNN)

```
Acerta a tokenização
# Número máximo de palavras para considerar
# São consideradas as mais frequentes, as demais são
# ignoradas
num_words = 20000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train)
sequences_test = tokenizer.texts_to_sequences(x_test)
word2index = tokenizer.word_index
V = len(word2index)
print("%s tokens" % V)

7202 tokens

Acerta o tamanho das sequências (padding)
# Acerta o tamanho das sequências (padding)
data_train = pad_sequences(sequences_train) # usa o tamanho da maior seq.
T = data_train.shape[1] # tamanho da sequência
data_test = pad_sequences(sequences_test, maxlen=T)
print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)

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

Definição do modelo

# Define o modelo
D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido
M = 5 # tamanho do hidden state, quantidade de unidades LSTM
i = Input(shape=(T,)) # Entra uma frase inteira
x = Embedding(V+1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation="sigmoid")(x) # Sigmoide pois só tem 2 valores
model = Model(i, x)

Sumário do modelo
model.summary()
```

Model: "functional_1"

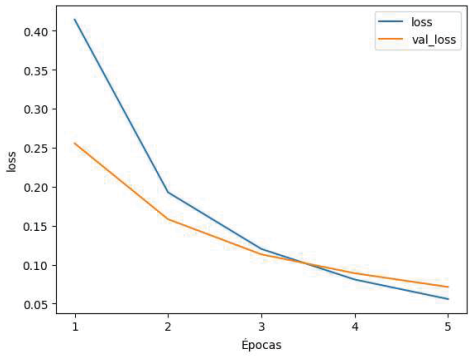
Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 189)	0
embedding_1 (Embedding)	(None, 189, 20)	144,060
lstm_1 (LSTM)	(None, 5)	520
dense_1 (Dense)	(None, 1)	6

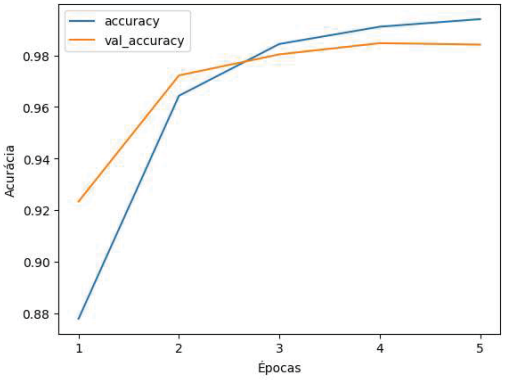
Total params: 144,586 (564.79 KB)
Trainable params: 144,586 (564.79 KB)
Non-trainable params: 0 (0.00 B)

```
Compila e treina o modelo

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

```
Epoch 1/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 11s 67ms/step - accuracy: 0.8475 - loss: 0.5297 - val_accuracy: 0.9233 - val_loss: 0.2555
Epoch 2/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 10s 69ms/step - accuracy: 0.9589 - loss: 0.2163 - val_accuracy: 0.9723 - val_loss: 0.1583
Epoch 3/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 7s 56ms/step - accuracy: 0.9854 - loss: 0.1252 - val_accuracy: 0.9804 - val_loss: 0.1132
Epoch 4/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 8s 71ms/step - accuracy: 0.9892 - loss: 0.0907 - val_accuracy: 0.9848 - val_loss: 0.0890
Epoch 5/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 10s 71ms/step - accuracy: 0.9917 - loss: 0.0622 - val_accuracy: 0.9842 - val_loss: 0.0714
```

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



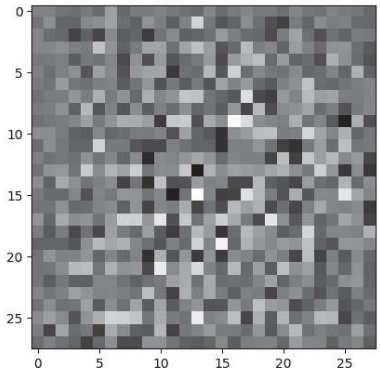



```
Predição de um novo texto

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

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

1/1 ------------------ 0s 28ms/step
[[0.01949643]]
OK
```

Gerador de Dígitos Fake (GAN)

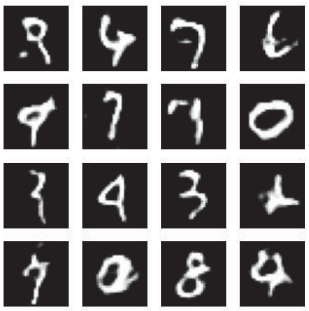
```
# Carregar a base de dados
(train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()
# Normalização
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).asty-
pe('float32')
train_images = (train_images - 127.5) / 127.5 # Normaliza entre [-1, 1]
# Gera o banco em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

def make_generator_model():
model = tf.keras.Sequential()
model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU())
model.add(layers.Reshape((7, 7, 256)))
assert model.output_shape == (None, 7, 7, 256)  # Note: None is the batch
size

model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
padding='same', use_bias=False))
assert model.output_shape == (None, 7, 7, 128)
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU())
model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
padding='same', use_bias=False))
assert model.output_shape == (None, 14, 14, 64)
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU())
model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same',
use_bias=False, activation='tanh'))
assert model.output_shape == (None, 28, 28, 1)

return model

# Exemplo de uso da função
model = make_generator_model()

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




```
# Cria o DISCRIMINADOR
def make_discriminator_model():
model = tf.keras.Sequential()
model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
input_shape=[28, 28, 1]))
model.add(layers.LeakyReLU())
model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
model.add(layers.LeakyReLU())
model.add(layers.Dropout(0.3))
model.add(layers.Flatten())
model.add(layers.Dense(1))

return model

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

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

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

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

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

# Função que faz um passo de treinamento
```

```
# É uma `tf.function`, que compila essa função
@tf.function
def train_step(images):
noise = tf.random.normal([BATCH_SIZE, noise_dim])
with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
generated_images = generator(noise, training=True)
real_output = discriminator(images, training=True)
fake_output = discriminator(generated_images, training=True)
gen_loss = generator_loss(fake_output)
disc_loss = discriminator_loss(real_output, fake_output)

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

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

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

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

# Save the model every 15 epochs
if (epoch + 1) % 15 == 0:
checkpoint.save(file_prefix=checkpoint_prefix)

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

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

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

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

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

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

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

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

display_image(EPOCHS)

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

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

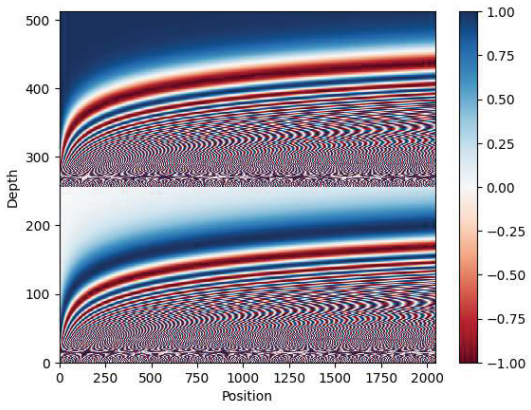


Tradutor de Textos (Transformer)

```
# Tokenização e Destokenização do texto
model_name = "ted_hrlr_translate_pt_en_converter"
tf.keras.utils.get_file(
f"{model_name}.zip",
f"https://storage.googleapis.com/download.tensorflow.org/models/
{model_name}.zip",
cache_dir='.',
cache_subdir='',
extract=True
)

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

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

# Pipeline simples: processa, embaralha, agrupa os dados, prefetch
# Datasets de entrada terminam com prefetch
BUFFER_SIZE = 20000
BATCH_SIZE = 64

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

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

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

def positional_encoding(position, d_model):
angle_rads = get_angles(
np.arange(position)[:, np.newaxis],
np.arange(d_model)[np.newaxis, :],
d_model
)
# sin em índices pares no array; 2i
angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
# cos em índices ímpares no array; 2i+1
angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
# newaxis, aumenta a dimensão [] -> [ [] ]
pos_encoding = angle_rads[np.newaxis, ...]
return tf.cast(pos_encoding, dtype=tf.float32)

# CODIFICAÇÃO POSICIONAL
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)

pos_encoding = pos_encoding[0]
# Arrumar as dimensões
pos_encoding = tf.reshape(pos_encoding, (n, d // 2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))

plt.pcolormesh(pos_encoding, cmap='RdBu')
```

```
plt.ylabel('Depth')
plt.xlabel('Position')
plt.colorbar()
plt.show()
```




```
# Cria uma máscara de 0 e 1, 0 para quando há valor e 1 quando não há
def create_padding_mask(seq):
seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
# Add extra dimensions to add the padding
# to the attention logits.
return seq[:, tf.newaxis, tf.newaxis, :]  # (batch_size, 1, 1, seq_len)

# Máscara futura, usada no decoder
def create_look_ahead_mask(size):
# Zera o triângulo inferior
mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
return mask  # (seq_len, seq_len)

# Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):
# Q K^T
matmul_qk = tf.matmul(q, k, transpose_b=True)  # (..., seq_len_q,
seq_len_k)

# Converte matmul_qk para float32
dk = tf.cast(tf.shape(k)[-1], tf.float32)

# Divide por sqrt(d_k)
scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)

# Soma a máscara, e os valores faltantes serão um número próximo a -inf se
mask for fornecida
if mask is not None:
scaled_attention_logits += (mask * -1e9)

# Softmax normaliza os dados, somando 1. // (..., seq_len_q, seq_len_k)
attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)

# Calcula a saída
output = tf.matmul(attention_weights, v)  # (..., seq_len_q, depth_v)

return output, attention_weights

class MultiHeadAttention(tf.keras.layers.Layer):
def __init__(self, d_model, num_heads):
super(MultiHeadAttention, self).__init__()
```

```
self.num_heads = num_heads
self.d_model = d_model
assert d_model % self.num_heads == 0
self.depth = d_model // self.num_heads

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

def split_heads(self, x, batch_size):
"""Separa a última dimensão em (num_heads, depth).
Transpõe o resultado para o shape (batch_size, num_heads, seq_len, depth).
"""
x = tf.reshape(x, (batch_size, -1, self.num_heads, self.depth))
return tf.transpose(x, perm=[0, 2, 1, 3])

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

q = self.wq(q)  # (batch_size, seq_len, d_model)
k = self.wk(k)  # (batch_size, seq_len, d_model)
v = self.wv(v)  # (batch_size, seq_len, d_model)

q = self.split_heads(q, batch_size)  # (batch_size, num_heads, seq_len_q,
depth)
k = self.split_heads(k, batch_size)  # (batch_size, num_heads, seq_len_k,
depth)
v = self.split_heads(v, batch_size)  # (batch_size, num_heads, seq_len_v,
depth)

# Calcula a atenção para cada cabeça (de forma matricial)
# scaled_attention.shape == (batch_size, num_heads, seq_len_q, depth)
# attention_weights.shape == (batch_size, num_heads, seq_len_q, seq_len_k)
scaled_attention, attention_weights = scaled_dot_product_attention(q, k, v,
mask)

# Troca a dimensão 2 com 1, para acertar o num_heads
# (batch_size, seq_len_q, num_heads, depth)
scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])

# Concatena os valores em: (batch_size, seq_len_q, d_model)
concat_attention = tf.reshape(scaled_attention, (batch_size, -1, self.d_mo-
del))

output = self.dense(concat_attention)  # (batch_size, seq_len_q, d_model)

return output, attention_weights

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

class EncoderLayer(tf.keras.layers.Layer):
def __init__(self, d_model, num_heads, dff, rate=0.1):
super(EncoderLayer, self).__init__()
self.mha = MultiHeadAttention(d_model, num_heads)
self.ffn = point_wise_feed_forward_network(d_model, dff)
```

```
self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
self.dropout1 = tf.keras.layers.Dropout(rate)
self.dropout2 = tf.keras.layers.Dropout(rate)

def call(self, x, training, mask):
attn_output, _ = self.mha(x, x, x, mask)  # (batch_size, input_seq_len,
d_model)
attn_output = self.dropout1(attn_output, training=training)
out1 = self.layernorm1(x + attn_output)  # (batch_size, input_seq_len,
d_model)

ffn_output = self.ffn(out1)  # (batch_size, input_seq_len, d_model)
ffn_output = self.dropout2(ffn_output, training=training)
out2 = self.layernorm2(out1 + ffn_output)  # (batch_size, input_seq_len,
d_model)

return out2

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

def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
# enc_output.shape == (batch_size, input_seq_len, d_model)
# (batch_size, target_seq_len, d_model)
attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
attn1 = self.dropout1(attn1, training=training)
out1 = self.layernorm1(attn1 + x)
# (batch_size, target_seq_len, d_model)
attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1, pad-
ding_mask)
attn2 = self.dropout2(attn2, training=training)
out2 = self.layernorm2(attn2 + out1) # (batch_size, target_seq_len, d_mo-
del)
ffn_output = self.ffn(out2) # (batch_size, target_seq_len, d_model)
ffn_output = self.dropout3(ffn_output, training=training)
out3 = self.layernorm3(ffn_output + out2) # (batch_size, target_seq_len,
d_model)
return out3, attn_weights_block1, attn_weights_block2

class Encoder(tf.keras.layers.Layer):
def __init__(self, num_layers, d_model, num_heads, dff,
input_vocab_size, maximum_position_encoding, rate=0.1):
super(Encoder, self).__init__()

self.d_model = d_model
self.num_layers = num_layers

self.embedding = tf.keras.layers.Embedding(input_vocab_size, d_model)
self.pos_encoding = positional_encoding(maximum_position_encoding,
self.d_model)
```

```
self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate)
for _ in range(num_layers)]

self.dropout = tf.keras.layers.Dropout(rate)

def call(self, x, training, mask):
seq_len = tf.shape(x)[1]

# adding embedding and position encoding.
x = self.embedding(x)  # (batch_size, input_seq_len, d_model)
x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
x += self.pos_encoding[:, :seq_len, :]

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

for i in range(self.num_layers):
x = self.enc_layers[i](x, training, mask)

return x  # (batch_size, input_seq_len, d_model)

class Decoder(tf.keras.layers.Layer):
def __init__(self, num_layers, d_model, num_heads, dff, target_vocab_size,
maximum_position_encoding, rate=0.1):
super(Decoder, self).__init__()

self.d_model = d_model
self.num_layers = num_layers

self.embedding = tf.keras.layers.Embedding(target_vocab_size, d_model)
self.pos_encoding = positional_encoding(maximum_position_encoding, d_model)

self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate)
for _ in range(num_layers)]

self.dropout = tf.keras.layers.Dropout(rate)

def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
seq_len = tf.shape(x)[1]
attention_weights = {}

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

for i in range(self.num_layers):
x, block1, block2 = self.dec_layers[i](x, enc_output, training,
look_ahead_mask, padding_mask)
attention_weights[f'decoder_layer{i+1}_block1'] = block1
attention_weights[f'decoder_layer{i+1}_block2'] = block2

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


class Transformer(tf.keras.Model):
def __init__(self, num_layers, d_model, num_heads, dff, input_vocab_size,
target_vocab_size, pe_input, pe_target, rate=0.1):
super().__init__()
```

```
self.encoder = Encoder(num_layers, d_model, num_heads, dff,
input_vocab_size,
pe_input, rate)
self.decoder = Decoder(num_layers, d_model, num_heads, dff,
target_vocab_size,
pe_target, rate)
self.final_layer = tf.keras.layers.Dense(target_vocab_size)

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

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

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

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

def create_masks(self, inp, tar):
# Encoder padding mask
enc_padding_mask = create_padding_mask(inp)

# Used in the 2nd attention block in the decoder.
# This padding mask is used to mask the encoder outputs.
dec_padding_mask = create_padding_mask(inp)

# Used in the 1st attention block in the decoder.
# It is used to pad and mask future tokens in the input received by
# the decoder.
look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])
dec_target_padding_mask = create_padding_mask(tar)
look_ahead_mask = tf.maximum(dec_target_padding_mask, look_ahead_mask)

return enc_padding_mask, look_ahead_mask, dec_padding_mask


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

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

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

# Define a função de perda
```

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

def loss_function(real, pred):
mask = tf.math.logical_not(tf.math.equal(real, 0))
loss_ = loss_object(real, pred)
mask = tf.cast(mask, dtype=loss_.dtype)
loss_ *= mask
return tf.reduce_sum(loss_) / tf.reduce_sum(mask)

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

# Define as métricas
train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy = tf.keras.metrics.Mean(name='train_accuracy')

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

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

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

EPOCHS = 20
train_step_signature = [
tf.TensorSpec(shape=(None, None), dtype=tf.int64),
tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]

@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
tar_inp = tar[:, :-1]
tar_real = tar[:, 1:]

with tf.GradientTape() as tape:
predictions, _ = transformer([inp, tar_inp], training=True)
loss = loss_function(tar_real, predictions)
gradients = tape.gradient(loss, transformer.trainable_variables)
optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))
```

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

for epoch in range(EPOCHS):
start = time.time()
train_loss.reset_state()
train_accuracy.reset_state()

# inp -> português, tar -> inglês
for batch, (inp, tar) in enumerate(train_batches):
train_step(inp, tar)

if batch % 50 == 0:
print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f} Ac-
curacy {train_accuracy.result():.4f}')

if (epoch + 1) % 5 == 0:
ckpt_save_path = ckpt_manager.save()
print(f'Saving checkpoint for epoch {epoch + 1} at {ckpt_save_path}')

translator = Translator(tokenizers, transformer)
sentence = "Eu li sobre triceratops na enciclopédia."
translated_text, translated_tokens, attention_weights = translator(tf.cons-
tant(sentence))
print(f'{"Prediction":15s}: {translated_text}')

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

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

def __call__(self, sentence, max_length=20):
# Verifica se o input é um tensor
assert isinstance(sentence, tf.Tensor)

if len(sentence.shape) == 0:
sentence = sentence[tf.newaxis]

# Tokeniza a sentença de entrada em português
sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
encoder_input = sentence

# Obtém os tokens de início e fim em inglês
start_end = self.tokenizers.en.tokenize([''])[0]
start = start_end[0][tf.newaxis]
end = start_end[1][tf.newaxis]

# Inicializa o array de saída
output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
output_array = output_array.write(0, start)

for i in tf.range(max_length):
output = tf.transpose(output_array.stack())

# Gera previsões com o transformer
predictions, _ = self.transformer([encoder_input, output], training=False)
```

```
predictions = predictions[:, -1:, :]  # (batch_size, 1, vocab_size)
predicted_id = tf.argmax(predictions, axis=-1)

# Adiciona o token previsto ao array de saída
output_array = output_array.write(i + 1, predicted_id[0])

if predicted_id == end:
break

# Converte a saída em texto e tokens
output = tf.transpose(output_array.stack())
text = self.tokenizers.en.detokenize(output)[0]
tokens = self.tokenizers.en.lookup(output)[0]

# Obtém os pesos de atenção
_, attention_weights = self.transformer([encoder_input, output[:, :-1]],
training=False)

return text, tokens, attention_weights

translator = Translator(tokenizers, transformer)
sentence = "Eu li sobre triceratops na enciclopédia."
translated_text, translated_tokens, attention_weights = translator(tf.cons-
tant(sentence))
print(f'{"Prediction":15s}: {translated_text}')

Prediction     : b'i read about trivias in enclos encyclopedia .'
```

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

Contexto

Este estudo de caso baseia-se na rotina de um integrante da equipe que atua no departamento de TI de uma rede de supermercados, composta por 29 lojas no estado do Paraná. Diante da necessidade de entender melhor o comportamento dos clientes, foi proposto o desenvolvimento de um escopo de Big Data para realizar uma análise detalhada do perfil dos clientes cadastrados via aplicativo da rede. O objetivo principal é segmentar os clientes de acordo com seus hábitos de compra, utilizando dados extraídos de sistemas CRM (Customer Relationship Management) e ERP (Enterprise Resource Planning), como histórico de compras, cadastro de clientes e produtos. Com esses dados, será realizada uma análise comportamental para identificar padrões de consumo e auxiliar na personalização de campanhas de marketing.presentar a resolução (somente o resultado) das questões do trabalho.

Ferramentas e Tecnologias utilizadas

- **Sistemas de Log Management**

IBM Qradar para correlação de eventos de segurança em tempo real.

- **Plataforma de Big Data**

Apache Hadoop núcleo do processamento de dados devido à sua capacidade de lidar com grandes volumes de dados distribuídos. Utilizando o MapReduce, os dados serão processados de forma distribuída, permitindo análises rápidas e eficientes, como a classificação de perfis de clientes.

- **NoSql**

Cassandra e HBase, serão empregadas para armazenar os dados não estruturados e semiestruturados, como o histórico de compras e as preferências dos clientes. Estes sistemas são ideais para lidar com grandes volumes de dados com velocidade e flexibilidade

- **Machine Learning**

Para a análise comportamental dos clientes, serão aplicadas técnicas de Machine Learning, utilizando bibliotecas como Mahout e Spark MLlib. Permitindo a criação de algoritmos de clustering, que serão aplicados para segmentar os clientes em três grupos distintos: clientes que gastam muito, clientes com gastos moderados e clientes que gastam pouco. Modelos preditivos também serão desenvolvidos para recomendar produtos com base no perfil de cada cliente, auxiliando nas campanhas de marketing personalizadas.

Organização dos dados

- **Modelagem dos Dados**

Conforme os tipos de dados existentes no CRM e ERP, diferentes abordagens de modelagem serão utilizadas, tais como:

1. Modelo Colunar: O HBase, como banco de dados NoSQL colunar, será utilizado para armazenar dados do histórico de compras e cadastros de clientes, proporcionando um modelo eficiente para leituras rápidas.
2. Modelo Chave-Valor: Sistemas como o Redis podem ser usados para armazenar preferências instantâneas dos clientes, como produtos visualizados recentemente.
3. Spark; será empregado para processar grandes volumes de dados, rodar algoritmos de clustering (K-means) para segmentar os clientes, e aplicar modelos de aprendizado de máquina para prever comportamentos de consumo e recomendar produtos. Essa arquitetura permite um fluxo contínuo de dados entre diferentes componentes.

Captura de dados em tempo real do PDV (ponto de venda)

1. Será utilizado Kafka e Storm. Pois o Kafka terá o papel de gestão em tempo real dos dados de CRM/ERP, consolidando o fluxo de transações e comportamento dos clientes em tempo real. A tecnologia Storm pode ser usada para processar esses dados conforme chegam, com uma arquitetura distribuída que garante alta disponibilidade, semelhante ao que é feito no pipeline de análise mostrado no exemplo da figura 1.

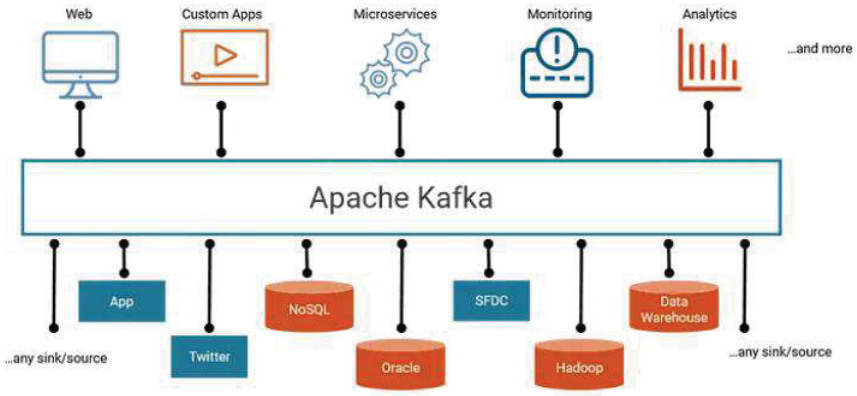


Figura 1: Exemplo de um fluxo de armazenamento e processamento de dados usando Kafka. Fonte: https://medium.com/trainingcenter/apache-kafka-838882261e83.

Visualização dos dados

Grafana para visualização em tempo real das requisições realizadas e informações relacionadas a performances do produto no geral. Além do Power BI para criação de dasboard personalizados de acordo com a necessidade da equipe de marketing, e-commerce e demais que compõem o setor administrativo da rede de supermercado.

Arquitetura Resumida

1. Ingestão de Dados: Kafka para coleta de dados em tempo real dos sistemas CRM/ERP.
2. Processamento de Dados: Storm processando streams de dados, seguido de persistência no Hadoop HDFS e bancos NoSQL (HBase ou Cassandra).
3. Análise: Spark para análise distribuída e segmentação de clientes com aprendizado de máquina.
4. Visualização: Dashboards interativos com Grafana e Power BI para análise do comportamento dos clientes.
5. Recomendação de Produtos: Sistemas baseados em aprendizado de máquina para recomendar produtos em tempo real de acordo com o comportamento.

Especificação dos Vs

Volume: É esperado um volume consideravel de dados que serão processados, provenientes do histórico de compras dos clientes das 29 lojas, cadastros no app.

Variedade: O conjunto de dados é altamente heterogêneo, incluindo dados estruturados (como cadastros de clientes e produtos) e não estruturados (como logs de navegação(app e e-commerce) e preferências de produtos). A utilização de sistemas NoSQL é crucial para lidar com essa variedade.

Velocidade: A análise em tempo real é necessária para recomendações instantâneas de produtos e personalização de campanhas de marketing. Para isso, será utilizadoprocessamento de dados em movimento com ferramentas como Apache Storm e Kafka (possui otimá performance em envio de mensageria que serão consumidos via tópicos), que fornecem capacidades de streaming.

Veracidade: Garantir a integridade e a qualidade dos dados é fundamental para todo o processo executado. Sistemas distribuídos, como o Hadoop, têm mecanismos para garantir a consistência dos dados, mesmo em grandes volumes e alta diversidade,.Desde modo enfatiza ainda mais a escolha do Hadoop.

Valor: A análise comportamental trará valor ao negócio ao identificar os padrões de compra, permitindo a segmentação precisa dos clientes e gerando insights valiosos para campanhas de marketing personalizadas e auxilio nas compras de produtos no geral.

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

Questão 1)

Após o carregamento da base e particionamento dos conjuntos de treinamento e validação, as características foram extraídas de ambas as bases usando o mesmo princípio, conforme exemplificado abaixo para o conjunto de treinamento, tanto usando LPB quanto VGG16.

```
def calcular_lbp(imagem):
# Converter a imagem para escala de cinza
imagem_gray = cv2.cvtColor(imagem, cv2.COLOR_BGR2GRAY)

# Calcular o LBP usando 8 pontos em um raio de 1 pixel
# 'uniform' considera apenas padrões uniformes, o que reduz a complexidade
lbp = feature.local_binary_pattern(imagem_gray, P=8, R=1, method='uniform')

# Criar um histograma dos valores LBP
# ravel() transforma a matriz LBP em um vetor unidimensional
# bins cria 10 bins para o histograma, de 0 a 10
(hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, 11), range=(0, 10))

# Converter o histograma para float para normalização
hist = hist.astype("float")

# Normalizar o histograma para que a soma total seja 1
hist /= hist.sum()

# Retornar o histograma normalizado
return hist

# Extrair características LBP conjunto 80% - Treino
lbp_caracteristicas = []
for img in X_treino:
lbp_hist = calcular_lbp(img)
lbp_caracteristicas.append(lbp_hist)

df_lbp = pd.DataFrame(lbp_caracteristicas)
df_lbp['classe'] = y_treino
df_lbp.to_csv('caracteristicas_lbpTreino.csv', index=False)


# Carregar o modelo VGG16 sem a última camada de classificação
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(250,
250, 3))
```

```
# Adicionar uma camada de pooling global para transformar as saídas em ve-
tores
x = base_model.output
x = GlobalAveragePooling2D()(x)

# Criar um novo modelo que retorna as características da penúltima camada
model = Model(inputs=base_model.input, outputs=x)

# Normalizar as imagens para o intervalo [0, 1]
X_treino_normalizado = X_treino / 255.0

# Extrair características usando a CNN VGG16
features_vgg = model.predict(X_treino_normalizado)  # Aqui os valores da
penúltima camada são armazenados

# Criar um DataFrame com as características extraídas
df_vgg = pd.DataFrame(features_vgg)  # Os valores estão agora em df_vgg
df_vgg['classe'] = y_treino  # Adicionar rótulos às características
df_vgg.to_csv('caracteristicas_vgg16Treino.csv', index=False)  # Salvar em
um CSV
```

Depois dos treinamentos com Random Forest, SVM e RNA, os modelos foram aplicados na base de teste e os resultados foram estes abaixo:

Resultados obtidos com a Base de dados: Test_4cl_amostra

Desempenho dos Modelos com Características LBP

Classe	Modelo	Acurácia	Sensibilidade	Especificidade	F1-Score
0	Random Forest	0.66	0.50	0.80	0.56
1	Random Forest	0.66	0.63	0.81	0.56
2	Random Forest	0.66	0.56	0.87	0.55
3	Random Forest	0.66	0.97	0.98	0.97
0	SVM	0.51	0.02	0.75	0.04
1	SVM	0.51	0.72	0.98	0.51
2	SVM	0.51	0.38	0.84	0.37
3	SVM	0.51	0.98	0.98	0.90
0	RNA	0.50	0.00	0.65	0.00
1	RNA	0.50	0.77	0.68	0.51
2	RNA	0.50	0.31	0.83	0.34
3	RNA	0.50	1.00	0.97	0.87

Desempenho dos Modelos com Características VGG16

Classe	Modelo	Acurácia	Sensibilidade	Especificidade	F1-Score
0	SVM	0.91	0.97	0.92	0.93
1	SVM	0.91	0.91	0.88	0.90
2	SVM	0.91	0.84	0.83	0.90
3	SVM	0.91	0.92	0.97	0.93
0	Random Forest	0.87	0.95	0.85	0.91
1	Random Forest	0.87	0.83	0.88	0.88
2	Random Forest	0.87	0.81	0.85	0.82
3	Random Forest	0.87	0.86	0.93	0.84
0	RNA	0.89	0.94	0.93	0.89
1	RNA	0.89	0.79	0.88	0.82
2	RNA	0.89	0.90	0.96	0.91
3	RNA	0.89	0.93	0.96	0.95

Melhor Modelo por Característica

- LBP: O Random Forest apresentou uma Acurácia de 66% com um F1-Score elevado na classe 3 (0.97).

- VGG16: O SVM teve a melhor Acurácia de 91% com um F1-Score alto em todas as classes.

Conclusão

O modelo SVM com características VGG16 teve o melhor desempenho geral, superando o Random Forest e o RNA em termos de Acurácia e F1-Score, evidenciando sua eficácia na classificação.

Questão 2)

O trecho de código abaixo mostra como foram configuradas as funções com data augmentation para o gerador de treino e sem data augmentation para o gerador de teste.

```
# Criar geradores de treino e validação a partir dos arrays X e y
train_generator = ImageDataGenerator(
rescale=1./255,
rotation_range=90,  # Rotação das imagens em até 90 graus
brightness_range=[0.1, 0.7],  # Ajuste do brilho
width_shift_range=0.5,  # Deslocamento horizontal
height_shift_range=0.5,  # Deslocamento vertical
horizontal_flip=True,  # Inversão horizontal
vertical_flip=True,  # Inversão vertical
preprocessing_function=preprocess_input) # Função de normalização

test_generator = ImageDataGenerator(preprocessing_function=preprocess_in-
put)
```

Os treinamentos com data augmentation e sem data augmentation (exemplo somente do VGG16) foram configurados conforme os códigos que serão exibidos abaixo:

```
# Para o treinamento prevenir o overfitting
early_stop = EarlyStopping(monitor='val_loss',
patience=10,
restore_best_weights=True,
mode='min')


history_0 = vgg_aug.fit(traingen,
epochs=n_epochs,
steps_per_epoch=steps_per_epoch,
validation_data=validgen,
validation_steps=val_steps,
verbose=True)

history_vgg_no_aug = vgg_no_aug.fit(no_aug_train_generator,
epochs=n_epochs,
steps_per_epoch=steps_per_epoch,
validation_data=no_aug_val_generator,
validation_steps=val_steps,
verbose=True)
```

As predições foram calculadas conforme o exemplo do VGG16 exibido abaixo:

```
# VGG COM DATA AUGMENTATION
pred_vgg_aug = vgg_aug.predict(testgen)
pred_vgg_aug_class = np.argmax(pred_vgg_aug, axis=1)


# VGG SEM DATA AUGMENTATION
pred_vgg_no_aug = vgg_no_aug.predict(testgen)
pred_vgg_no_aug_class = np.argmax(pred_vgg_no_aug, axis=1)
```

Os resultados foram compilados na tabela baixo:

Desempenho dos Modelos RESNET50 e VGG16

Classe	Modelo	Acurácia	Sensibilidade	Especificidade	F1-Score
0	Resnet50 COM Data Augmentation	24.26%	0.059406	0.985185	0.112055
1	Resnet50 COM Data Augmentation	24.26%	0.022222	0.971530	0.043451
2	Resnet50 COM Data Augmentation	24.26%	0.155556	0.697509	0.254380
3	Resnet50 COM Data Augmentation	24.26%	0.944444	0.405694	0.567579
0	Resnet50 SEM Data Augmentation	81.13%	0.930693	1.000000	0.964103
1	Resnet50 SEM Data Augmentation	81.13%	0.700000	0.889680	0.783524
2	Resnet50 SEM Data Augmentation	81.13%	0.688889	0.882562	0.773791
3	Resnet50 SEM Data Augmentation	81.13%	0.933333	0.985765	0.958833
0	VGG16 COM Data Augmentation	44.47%	0.029703	1.000000	0.057692
1	VGG16 COM Data Augmentation	44.47%	0.000000	1.000000	0.000000
2	VGG16 COM Data Augmentation	44.47%	0.011111	1.000000	0.021978
3	VGG16 COM Data Augmentation	44.47%	1.000000	0.014235	0.028070
0	VGG16 SEM Data Augmentation	63.07%	0.811881	0.951852	0.876312
1	VGG16 SEM Data Augmentation	63.07%	0.466667	0.882562	0.610515
2	VGG16 SEM Data Augmentation	63.07%	0.466667	0.797153	0.588699
3	VGG16 SEM Data Augmentation	63.07%	0.833333	0.903915	0.867190

Conclusão

Com base nas métricas encontradas em todos os modelos treinados, o modelo ResNet50 sem Data Augmentation é a melhor escolha. Ele combina uma alta acurácia com boas métricas de sensibilidade e F1-Score, sugerindo que ele generaliza bem e é eficaz em classificar corretamente as diferentes classes.

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

No trabalho foram abordados todos os parâmetros elencados no enunciado. Entre as soluções apresentadas foram destacados temas como a criação de ferramentas de auditoria e transparência, implementação de políticas de privacidade rigorosas e fomento a educação e conscientização.

Como complemento, o texto abaixo foi apresentado como conclusão do trabalho.

CONCLUSÃO

O estudo das implicações éticas do uso do ChatGPT mostra a necessidade urgente de abordagens responsáveis na implementação de tecnologias de inteligência artificial. É essencial identificar e resolver dilemas éticos, como privacidade, disseminação de desinformação e discriminação algorítmica, para garantir que o desenvolvimento e uso da IA estejam alinhados com valores de justiça, transparência e respeito aos direitos humanos.

Propostas concretas, como a criação de ferramentas de auditoria e transparência, a implementação de políticas de privacidade rigorosas, o desenvolvimento de práticas de design responsável e a promoção da educação e conscientização, são fundamentais para garantir um uso ético e responsável da IA. A adoção dessas medidas pode ajudar a mitigar riscos, promover a confiança dos usuários e assegurar que a IA seja utilizada de maneira justa e benéfica.

Além disso, é crucial promover a colaboração entre desenvolvedores, pesquisadores, formuladores de políticas e a sociedade em geral. Discussões abertas permitem a troca de diferentes perspectivas, enriquecendo o debate e promovendo soluções mais equilibradas e justas. A reflexão contínua e a adaptação das práticas de desenvolvimento de IA são necessárias para garantir que essas tecnologias beneficiem a sociedade como um todo.

Portanto, integrar princípios éticos no desenvolvimento e uso do ChatGPT e outras tecnologias de IA não é apenas necessário, mas uma responsabilidade coletiva. Somente através de uma abordagem ética e colaborativa podemos garantir que a inteligência artificial seja uma força positiva na sociedade, promovendo o bem-estar e respeitando os direitos de todos.

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Nome do artigo escolhido: Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows

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 artigo é identificar como as ferramentas de auto-ml são realmente exploradas no mercado. O estudo mostra as vantagens e desvantagens do uso dessas ferramentas e também mostra insights sobre a melhor forma de usa-las, indicando, por exemplo, em que fase do projeto elas podem ser aproveitadas de uma forma mais potencializada. Outra questão debatida é o quanto a análise/intervenção humana é importante nos projetos que usam ferramentas de auto-ml.	A situação que motivou o estudo, foi a necessidade de uma melhor compreeensão de como as ferramentas de auto-ml são usadas de fato em projetos reais e quais são os impactos da integração com humanos (ou afalta dela) nos ambientes dos entrevistados, visto que nas análises inciais da literatura, não foram encontrados muitos trabalhos que focavam na experiência concreta do uso dessas ferramentas em projetos. Naturalmente, o estudo revelaria os benefícios e desafios encontrados pelos profissionais que já usaram as ferramentos nos projetos que atuaram.	A metodologia escolhida para o estudo foi a entrevista. Os autores convidaram profissionais com experiências reais no uso de ferramentas de auto-ml a participar de entrevistas semi-estruturadas que duravam cerca de 1 hora (as mesmas foram gravadas em quase todos os casos). As entrevistas buscavam entender a funçao que a pessoa ocupava na organização que atuava, quais eram as experiências no desenvolvimento de fluxos de trabalhos de ml (mesmo sem auto-ml), quais eram os desafios nestes casos de uso, quais ferramentas eram usadas e a percepção sobre cada uma delas, além de indicar como auto-ml se encaixa no fluxo de trabalhodos entrevistados.	Os resultados obtidos pelo estudo foram: - Identificação e confirmação das ferramentas mais usadas. Também foi apontada a necessidade de adaptação das ferramentas para dar mais transparênciados processos internos delas. - O estudo identificou que muitas vezes os resultados obtidos pelas ferramentas de auto-ml eram usados como uma espécie de ponto partidaefuncionavam como referência para adequações que visavam o apromiramento dos modelos. - O estudo mostra que o melhor caminho é utilizar um modelo de projeto no qual a colaboração entre a automação e os ajustes manuais se complementam, visto que os entrevistados consideram a experiência e conhecimento humanos essenciais para que as soluções seja implementadas de forma otimizada. - A principal contribuição do estudo foi identificar que ainda não existe uma confiança plena em usar apenas ferramentas auto-ml dispensando totalmente a intervenção/curadoria humana.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

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

2 Regressão (RNA)

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

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

Informações:

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

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
data = pd.read_csv(url, delimiter=';')
```

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

- Imagem onírica obtida por *Main Loop*;
- Imagem onírica obtida ao levar o modelo até uma oitava;

- Diferenças entre imagens oníricas obtidas com *Main Loop* e levando o modelo até a oitava.

  Informações:

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

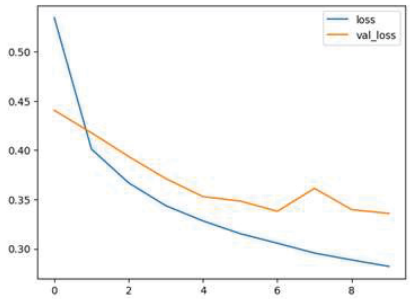
  url = "https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-cat_on_snow.jpg"

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

B – RESOLUÇÃO

Questão 1)

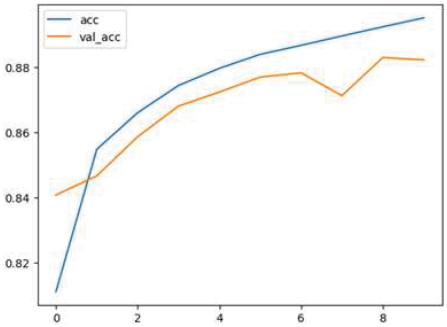
Função de Perda: 0.3329



O gráfico mostra que as perdas de treinamento (loss) e validação (val_loss) diminuem com as épocas, indicando aprendizado do modelo.

- **Primeiras épocas**: Queda acentuada em ambas as perdas, mostrando rápido aprendizado inicial.
- **Últimas épocas**: loss continua caindo, mas val_loss estabiliza com pequenas flutuações.

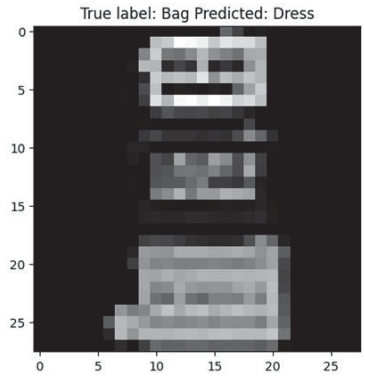
Função de Acurácia: 0.8826



O gráfico mostra que, ao longo das épocas, a acurácia no conjunto de treinamento (acc) cresce de forma consistente, enquanto a acurácia no conjunto de validação (val_acc) também aumenta, mas apresenta pequenas flutuações.

- **Primeiras épocas**: Melhorias rápidas em ambas as acurácias, indicando aprendizado inicial eficiente.
- **Últimas épocas**: acc continua crescendo, mas val_acc se estabiliza e começa a divergir.

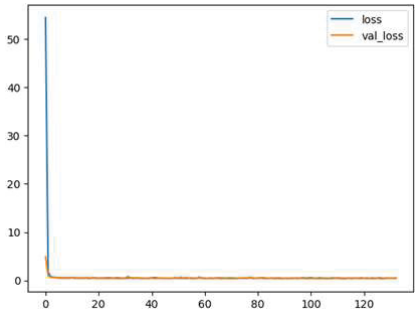
Mostrar algumas classificações erradas: A predição aponta que esta imagem é um Dress mas na verdade é um Bag.



Questão 2)

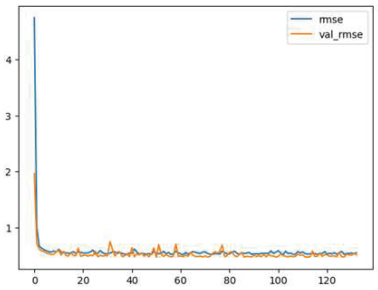
Apresentar a resolução (somente o resultado) das questões do trabalho.

Função de Perda:



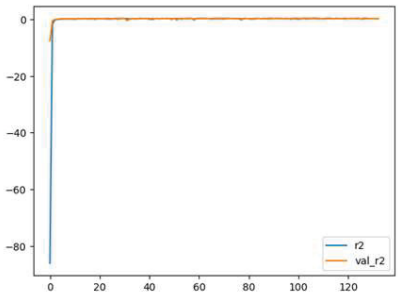
O modelo demonstrou um aprendizado eficiente e consistente, alcançando boa generalização e evitando problemas de sobreajuste ou subajuste. Esses resultados indicam que a arquitetura e os parâmetros utilizados foram adequados para a tarefa de regressão.

Função RMSE:



O gráfico de RMSE indica um bom aprendizado do modelo, com rápida redução do erro inicial e estabilização em valores baixos após cerca de 10-15 épocas. As curvas de treino e validação são próximas, mostrando boa generalização. Não há sinais de overfitting, e o desempenho foi consistente nos dados de validação. O modelo é eficaz para a tarefa proposta.

Função R2:



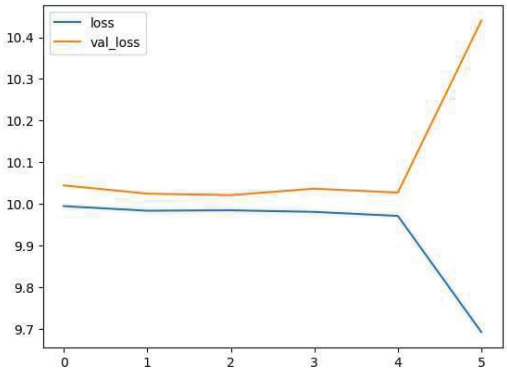
O gráfico do coeficiente R2 mostra uma rápida melhoria inicial, alcançando valores estáveis após poucas épocas. As curvas de treino e validação são muito próximas, indicando boa generalização do modelo. O R2 próximo de 0 ou levemente negativo sugere que o modelo ainda não explica completamente a variação dos dados. Ajustes adicionais podem melhorar a qualidade do ajuste.

Conclusão

Os resultados indicam um MSE de 0.35 e um RMSE de 0.59, representando um erro moderado na previsão. O R2 de 0.43 mostra que o modelo explica cerca de 43% da variância dos dados, sugerindo espaço para melhorias no desempenho preditivo.

Questão 3)

Função de perda:



Análise e Observações

O gráfico mostra que durante as primeiras épocas do treinamento do modelo, o aprendizado ainda é baixo, por isso os valores de loss eval_loss pra□camente não sofrem alteração, porém a partir da quinta época, o valor de loss diminuiu de forma agressiva, mas o val_loss chega a aumentar.

Devido a função early stopping configurada para monitorar o val_loss (função de perda dos dados de validação), o treinamento foi interrompido, pois o val_loss não estava melhorando.

Essa parada antecipada mostra que a continuidade do treinamento resultaria em overfiting, visto que os dados de validação não estavam tendo evolução no aprendizado e por isso o modelo não está generalizando bem os novos dados.

Justificativa dos resultados e sugestão de Intervenção

Alguns dos parâmetros utilizados podem ser ajustados para buscar um modelo mais eficiente, além disso, outras questões podem ser verificadas. Algumas ações que podem tornar o modelo melhor seriam a redução da taxa de aprendizado (foi usado 0,05) visando evitar oscilações e a redução do número de neurônios (foi usado 1024) na camada densa, visto que a redução do número de pesos pode auxiliar na generalização. O número de épocas pode ser aumentado para um número entre 50 e 100, pois desta maneira o modelo poderá ver os dados mais vezes e consequentemente pode ajustar os pesos de uma forma mais gradual para chegar de uma convergência com melhor precisão. A alteração do batch_size também poderá melhorar o modelo, pois o número usado (512) acabou não capturando bem a variação dos dados usados. Outros testes que podem ser feitos são a alteração do algoritmo otimizador para verificação se o resultado final tem um ganho na generalização e testar outros valores mais altos no dropout na camada densa (neste caso, também pode ser necessário ajustar o número de épocas para uma avaliação mais eficiente).

Exemplo de recomendação de livro:

9. Recomendações para o usuário 20584

```
# Gerar o array com o usuário único
# repete a quantidade de livros
input_usuario = np.repeat(a=20584, repeats=M)
livro = np.array(list(set(ISBN_ids)))

preds = model.predict( [input_usuario, livro] )

# descentraliza as predições
rat = preds.flatten() + avg_notas

# índice da maior nota
idx = np.argmax(rat)

print("Recomendação: Livro - ", livro[idx], " / ", rat[idx] , "*")
```

```
4028/4028 ━━━━━━━━━━━━━━━━━━━━ 6s 1ms/step
Recomendação: Livro -  77837  /  5.0071154 *
```

No exemplo acima, é feita uma sugestão de livro para o usuário 20584. Além de apontar o id do livro sugerido, é feita a predição de uma nota aproximada que o usuário deve atribuir ao livro, considerando todos as notas dadas ao livro por outros usuários e também as notas que o próprio usuário tende a atribuir aos livros que ele já fez a leitura.

Questão 4)

Imagem onírica obtida por Main Loop:

```
dream_img = run_deep_dream_simple(img=original_img,
                                  steps=100, step_size=0.01)
```

A imagem onírica obtida no Main Loop resulta do processo de amplificação de padrões que o modelo pré-treinado reconhece nos dados de entrada. Esse efeito é gerado utilizando a técnica conhecida como Deep Dream.

Explicação breve dos resultados

O que é o Deep Dream?

O Deep Dream é uma técnica baseada em redes neurais convolucionais (como InceptionV3) que maximiza as ativações de camadas específicas. Isso amplifica os padrões que a rede reconhece na imagem, criando formas "surreais" e texturas complexas.

Como o Main Loop funciona?

A imagem de entrada é processada iterativamente. O modelo tenta aumentar as ativações dos filtros das camadas especificadas no dream_model. Em cada passo, o modelo ajusta os valores dos pixels da imagem de entrada para amplificar os padrões.

Características da Imagem Resultante

Texturas repetitivas: Elementos visuais, como curvas, olhos ou formas geométricas, são amplificados, criando um aspecto "onírico".

Padrões vibrantes: As cores e os detalhes tornam-se mais pronunciados à medida que os filtros são maximizados.

Integração dos padrões: As texturas não são adicionadas aleatoriamente; elas emergem de áreas da imagem onde os filtros da rede detectaram elementos significativos.

Por que a imagem do gato ficou surreal?

O modelo identificou características específicas do gato (como olhos, pelos e contornos) e amplificou esses padrões. Camadas diferentes da rede contribuem para texturas e formas mais complexas à medida que os filtros interpretam a imagem.

Imagem onírica obtida ao levar o modelo até uma oitava:

A imagem onírica obtida com o uso de oitavas apresenta padrões mais refinados e complexos, pois combina informações em múltiplas escalas de resolução. Detalhes maiores, como o corpo do gato, são destacados em resoluções menores, enquanto texturas finas, como olhos e

contornos, são refinadas em resoluções maiores. Esse processo multiescala reduz o ruído e cria uma integração mais harmoniosa dos padrões amplificados. No código, a imagem é redimensionada iterativamente com o fator OCTAVE_SCALE, passando pelo Deep Dream em cada nível antes de ser retornada ao tamanho original. O resultado é uma obra visual rica, detalhada e surreal.

Diferenças entre imagens oníricas obtidas com Main Loop e levando o modelo até a oitava

As diferenças entre as imagens oníricas obtodas com o Main Loop e levando o modelo até uma oitava são as seguintes:

Detalhamento Multiescala: No Main Loop, os padrões são amplificados em uma única resolução, resultando em detalhes uniformes e, às vezes, caóticos. Com oitavas, o modelo processa a imagem em diferentes escalas, refinando padrões em níveis globais e locais.

Redução de Ruído: A imagem do Main Loop tende a ser mais ruidosa devido à amplificação direta. Usando oitavas, o redimensionamento multiescala suaviza transições e integra melhor os padrões.

Complexidade Visual: No Main Loop, os detalhes podem parecer menos harmoniosos. Já com oitavas, os padrões são mais ricos e detalhados, devido ao refinamento em várias resoluções.

Resolução e Granularidade: A imagem do Main Loop é processada diretamente na resolução original, enquanto com oitavas, a granularidade dos padrões melhora em várias escalas de tamanho.

Harmonia Visual: A técnica com oitavas cria uma composição mais equilibrada, combinando detalhes finos e amplos, enquanto o Main Loop é limitado à resolução única do processamento.

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

- Explicação do **contexto e o publico-alvo** da visualização de dados e do storytelling que será desenvolvido;

- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada**; (**50 pontos**)

B – RESOLUÇÃO

CONTEXTO

O ENEM (Exame Nacional do Ensino Médio) é umas das provas aplicadas no Brasil, cujas notas são usadas em várias instituições como requisito para entrada no Ensino Superior. As notas do ENEM também são usadas em diversos contextos para concessão de bolsas de estudo. Existe uma grande discussão sobre a desigualdade no desempenho dos alunos, considerando o tipo da escola de formação do aluno (pública ou privada), a renda familiar e também a região. Esta análise tem o objetivo de identificar por meio de dados, o quanto os fatores citados são relevantes para a nota final do aluno no exame e entender se existe um nível relevante de disparidade.

VISUALIZAÇÃO DOS DADOS

Antes da apresentação, é preciso informar que base original não foi aproveitada integralmente, pois na avaliação do autor, muitos dados tanto do aluno, como da escola ou da própria

prova seriam desnecessários ou mesmo de pouco proveito no estudo inicial. Dados como o número de inscrição, a nacionalidade do aluno, o município da escola (será usado apenas o estado), o tipo de prova (azul, rosa e etc) e a quantidade de geladeiras na residência foram descartados, porém podem ser usados em um estudo mais aprofundado futuramente. Os dados principais do estudo são o tipo de escola, as notas nas provas, a renda familiar e a quantidade de pessoas que moram na casa do aluno.

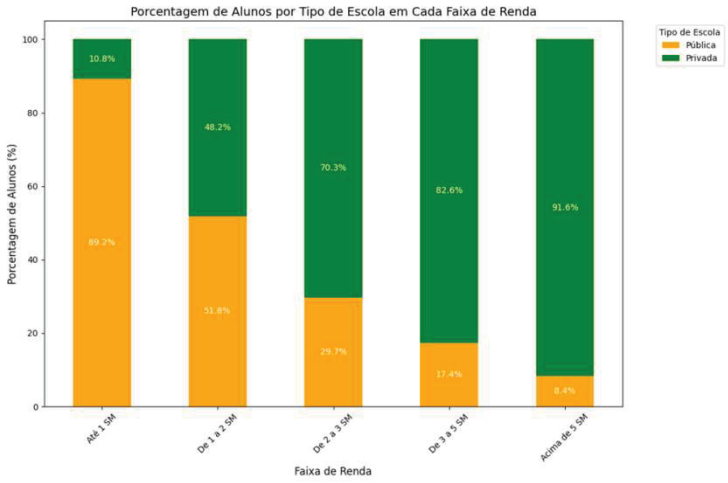
NARRATIVA

O objetivo é mostrar por meio de gráficos como as condições socioeconômicas podem influenciar o desempenho dos alunos na nota final do ENEM, além de verificar se existem diferencias em relação à região (estado) dos alunos.

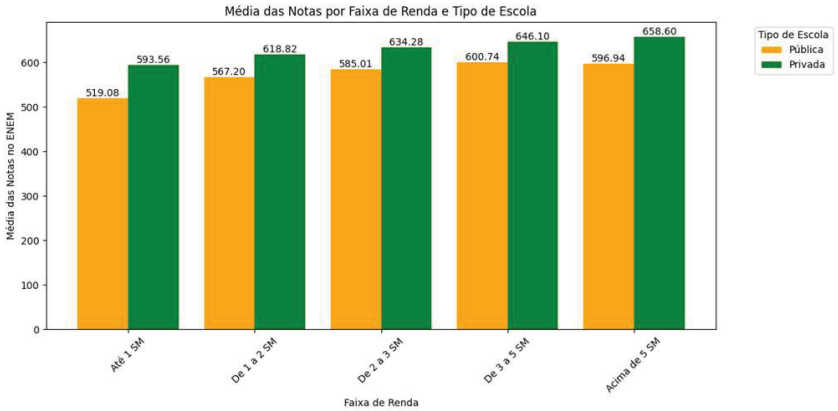
Em relação às faixas de renda per capita, foram separadas 5 categorias. A sigla “SM” é relativa ao salário mínimo e as faixas foram separadas considerando múltiplos do salário mínimo. As categorias são estas abaixo:

- Até 1 SM;
- De 1 a 2 SM;
- De 2 a 3 SM;
- De 3 a 5 SM;
- Acima de 5 SM.

Considerando estas faixas de renda, o gráfico abaixo evidencia que quanto maior é a renda per capita da família, maior é a possibilidade do aluno frequentar uma escola particular.



Podemos ver no gráfico abaixo que mesmo em faixas de renda iguais, alunos de escolas privadas, em média, têm um desempenho superior aos alunos de escolas públicas. Com estes dados, já podemos evidenciar que independentemente do tipo de escola, a renda per capita é um fator que tem um peso grande no desempenho dos alunos.



CONCLUSÃO

Pelas informações desta visualização de dados, fica evidente que as condições socioeconômicas são um fator determinante para um bom desempenho no ENEM, pois de acordo com os dados apresentados, os estados com um melhor desempenho tem uma maior concentração de alunos nas faixas superiores de renda per capita, o que consequentemente possibilita uma maior de chance destes alunos frequentarem escolas particulares, que por sua vez, demonstraram um desempenho melhor em relação a escola pública. Essa disparidade pode ser considerada uma alerta para que políticas públicas devem ser criadas ou modificadas para que essas diferenças sejam minimizadas e o para que o Brasil possa ter uma equidade na aprendizagem nas escolas, sejam elas privadas ou públicas.

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

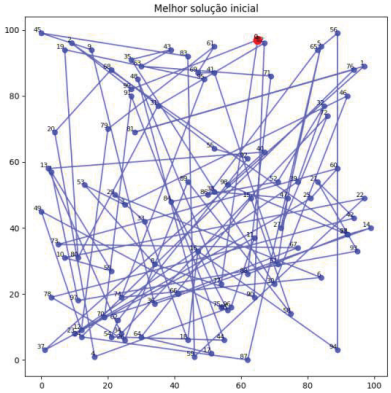
A seguir serão destacados os trechos de código usados para criação do indivíduo e para criação da população inicial, assim como os trechos para avaliação da melhor solução inicial e da melhor solução final. Também serão apresentadas as plotagens geradas com os resultados destas soluções.

Questão 1)

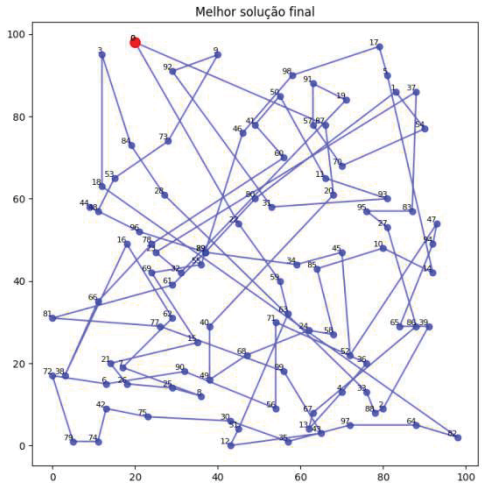
```
# Criar um indivíduo (permutação das cidades, sempre começando na cidade 0)
def criar_individuo():
percurso = list(range(1, NUM_CIDADES)) # Lista de cidades sem a cidade 0
random.shuffle(percurso)  # Embaralha as cidades
return [0] + percurso + [0]  # Inclui a cidade 0 no início e no final

# Criar população inicial com indivíduos aleatórios
def populacao_inicial():
return [criar_individuo() for _ in range(POPULACAO_SIZE)]

# Função de avaliação (fitness): distância total do percurso
def avaliacao(percurso):
dist = 0  # Inicializa a variável dist para armazenar a soma das distâncias
# Loop para percorrer todas as cidades no percurso (exceto a última)
for i in range(len(percurso) - 1):
# Calcula a distância entre a cidade atual (percurso[i]) e a próxima cidade
(percurso[i + 1])
dist += distancia(cidades[percurso[i]], cidades[percurso[i + 1]])
# Retorna o inverso da distância total, pois queremos um valor maior para
percursos mais curtos
return 1 / dist  # Quanto menor a distância, melhor o fitness

# Seleção dos pais por torneio (escolhe o melhor entre 5 indivíduos aleató-
rios)
def selecao(populacao):
return sorted(random.sample(populacao, 5), key=avaliacao, reverse=True)[0]

# Cruzamento OX (Order Crossover) entre dois pais
```

```
def cruzamento(pai1, pai2):
tamanho = len(pai1)
inicio, fim = sorted(random.sample(range(1, tamanho - 1), 2))  # Ponto de
corte

filho = [-1] * tamanho  # Inicializa filho com valores inválidos
filho[inicio:fim] = pai1[inicio:fim]  # Copia segmento do primeiro pai

p2_idx = fim  # Índice do segundo pai
for i in range(fim, tamanho - 1):
while pai2[p2_idx] in filho:
p2_idx = (p2_idx + 1) % (tamanho - 1)
filho[i] = pai2[p2_idx]

for i in range(1, inicio):
while pai2[p2_idx] in filho:
p2_idx = (p2_idx + 1) % (tamanho - 1)
filho[i] = pai2[p2_idx]

filho[0] = filho[-1] = 0  # Mantém a cidade inicial no começo e no fim
return filho

# Mutação (troca de duas cidades aleatórias no percurso)
def mutacao(individuo):
if random.random() < TAXA_MUTACAO:
i, j = random.sample(range(1, NUM_CIDADES - 1), 2)
individuo[i], individuo[j] = individuo[j], individuo[i]

# Evolução da população através de seleção, cruzamento e mutação
def evoluir(populacao):
nova_populacao = [max(populacao, key=avaliacao)]  # Preserva o melhor indi-
víduo
num_filhos = int(TAXA_CRUZAMENTO * POPULACAO_SIZE)

while len(nova_populacao) < num_filhos:
pai1, pai2 = selecao(populacao), selecao(populacao)
filho = cruzamento(pai1, pai2)
mutacao(filho)
nova_populacao.append(filho)

while len(nova_populacao) < POPULACAO_SIZE:
nova_populacao.append(criar_individuo())  # Adiciona novos indivíduos alea-
tórios

return nova_populacao
```

Melhor solução inicial



Melhor solução final



Questão 2)

A seguir serão apresentados os trechos de código usados para vetorizar as sentenças, calcular as similaridades e para a redução PCA:

```
# ============================
# Modelo 1: TF-IDF
# ============================
vectorizer = TfidfVectorizer()
tfidf_vectors = vectorizer.fit_transform(sentencas).toarray()

# ============================
# Modelo 2: Sentence Embedding (BERT)
# ============================
model = SentenceTransformer('all-MiniLM-L6-v2')
bert_vectors = model.encode(sentencas)



# ============================
# Calcular Similaridades
# ============================

def calc_similaridade_tfidf(vetores, limiar=0.6):
matriz_similaridade_tdidf = cosine_similarity(vetores)
np.fill_diagonal(matriz_similaridade_tdidf, 0)  # Zerar diagonal para não
contar similaridade
return (matriz_similaridade_tdidf >= limiar).sum() >= 2  # Pelo menos 2
sentenças com similaridade

def calc_similaridade_bert(vetores, limiar=0.7):
matriz_similaridade_bert = cosine_similarity(vetores)
np.fill_diagonal(matriz_similaridade_bert, 0)  # Zerar diagonal para não
contar similaridade
return (matriz_similaridade_bert >= limiar).sum() >= 2  # Pelo menos 2 sen-
tenças com similaridade

# ============================
# Redução PCA
```

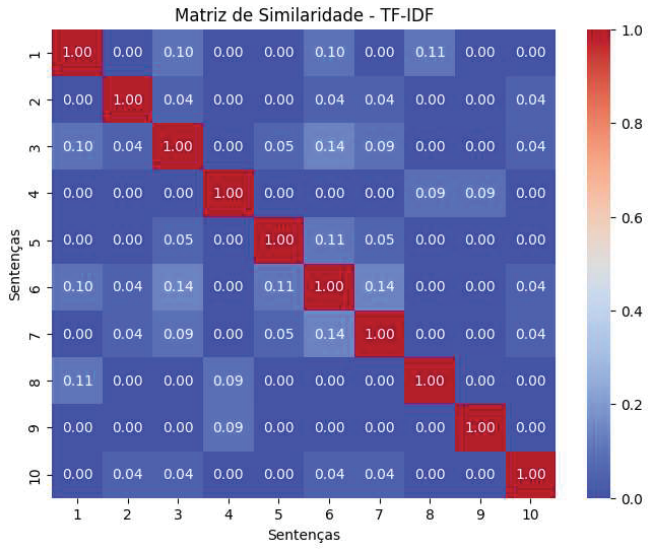
```
# ============================
def reduzir_pca(vetores):
pca = PCA(n_components=2)
return pca.fit_transform(vetores)

tfidf_2d = reduzir_pca(tfidf_vectors)
bert_2d = reduzir_pca(bert_vectors)
```

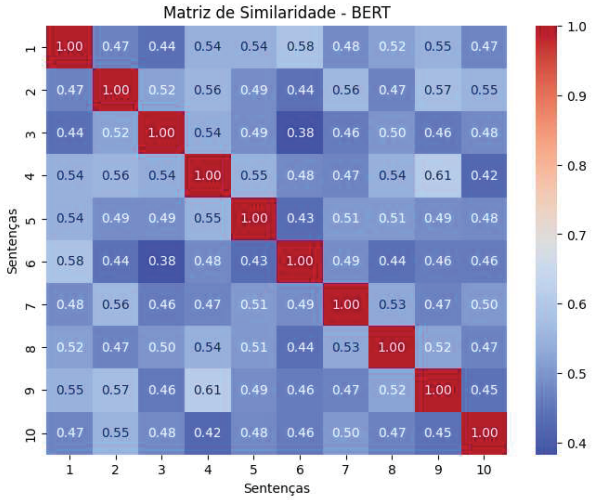
Abaixo serão apresentados os resultados de 2 entradas de dados diferentes:

Entrada de dados 1

```
Digite um texto com até 1000 caracteres (mínimo 6 frases com algumas pare-
cidas):

O pôr do sol na praia era deslumbrante. As ondas quebravam suavemente, cri-
ando um som relaxante. Um grupo de amigos ria enquanto jogava bola na
areia. Crianças corriam perto da água, desenhando formas com os pés. O céu
exibia tons vibrantes de laranja, rosa e roxo. Era o tipo de cenário que
parecia saído de um filme. Mais adiante, um casal caminhava de mãos dadas,
trocando sorrisos. O vento trazia o cheiro do mar e da brisa salgada. Os
pássaros voavam em formação, cruzando o horizonte. Um senhor tocava violão,
cantando baixinho uma melodia nostálgica.

 Sentenças selecionadas:
1. O pôr do sol na praia era deslumbrante.
2. As ondas quebravam suavemente, criando um som relaxante.
3. Um grupo de amigos ria enquanto jogava bola na areia.
4. Crianças corriam perto da água, desenhando formas com os pés.
5. O céu exibia tons vibrantes de laranja, rosa e roxo.
6. Era o tipo de cenário que parecia saído de um filme.
7. Mais adiante, um casal caminhava de mãos dadas, trocando sorrisos.
8. O vento trazia o cheiro do mar e da brisa salgada.
9. Os pássaros voavam em formação, cruzando o horizonte.
10. Um senhor tocava violão, cantando baixinho uma melodia nostálgica.
```

Foram gerados mapas de calor para evidenciar as similaridades entre as sentenças:





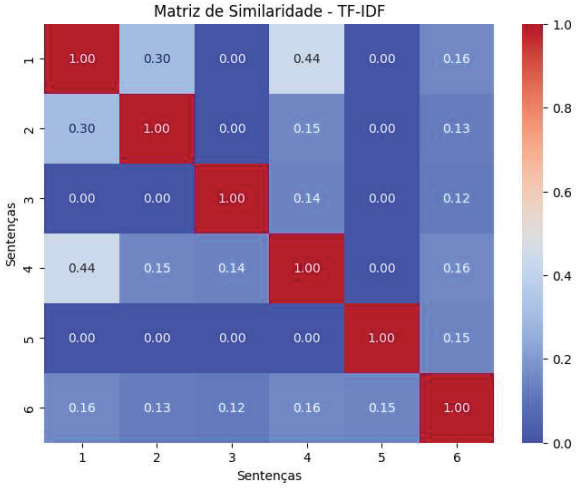
❌ O texto não tem pelo menos 2 sentenças similares (≥ 0.6 para TF-IDF ou ≥ 0.7 para BERT). Tente novamente.

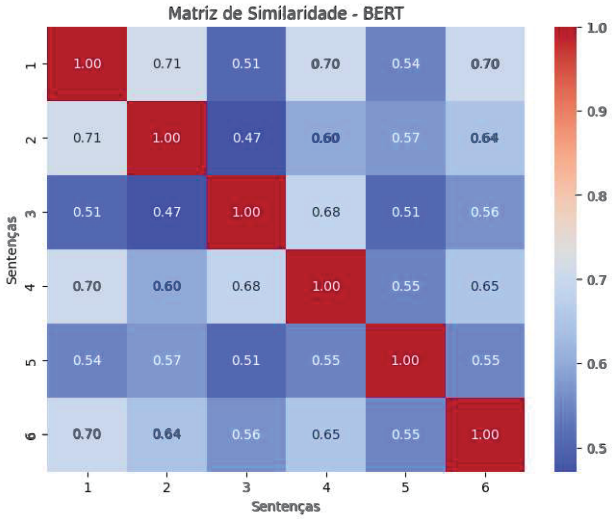
Entrada de dados 2

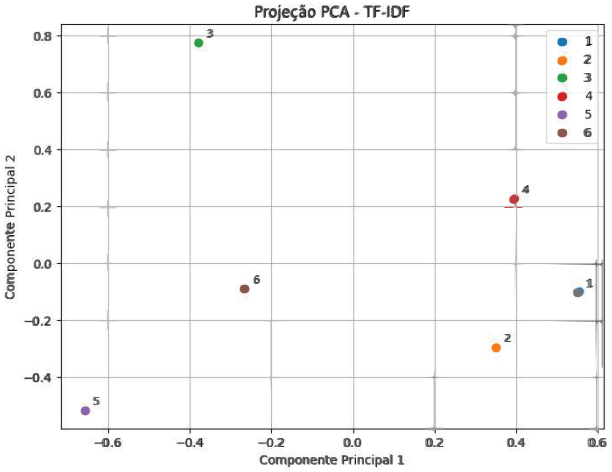
Digite um texto com até 1000 caracteres (mínimo 6 frases com algumas parecidas):

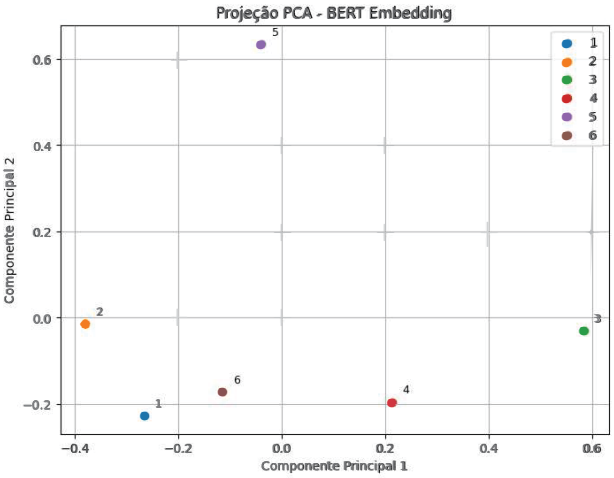
O gato preto pulou o muro. O gato preto saltou sobre a cerca. O cachorro marrom correu pelo jardim. O cachorro preto pulou o portão. O pássaro azul voou alto no céu. O gato e o cachorro brincaram no quintal.

📝 Sentenças selecionadas:
1. O gato preto pulou o muro.
2. O gato preto saltou sobre a cerca.
3. O cachorro marrom correu pelo jardim.
4. O cachorro preto pulou o portão.
5. O pássaro azul voou alto no céu.
6. O gato e o cachorro brincaram no quintal.




Matriz de Similaridade - BERT
1.00 0.71 0.51 0.70 0.54 0.70
0.71 1.00 0.47 0.60 0.57 0.64
0.51 0.47 1.00 0.68 0.51 0.56
0.70 0.60 0.68 1.00 0.55 0.65
0.54 0.57 0.51 0.55 1.00 0.55
0.70 0.64 0.56 0.65 0.55 1.00
Sentenças
Sentenças
1.0
0.9
0.8
0.7
0.6
0.5


Projeção PCA - TF-IDF
Componente Principal 2
Componente Principal 1
1
2
3
4
5
6


Projeção PCA - BERT Embedding
Componente Principal 2
Componente Principal 1
1
2
3
4
5
6