

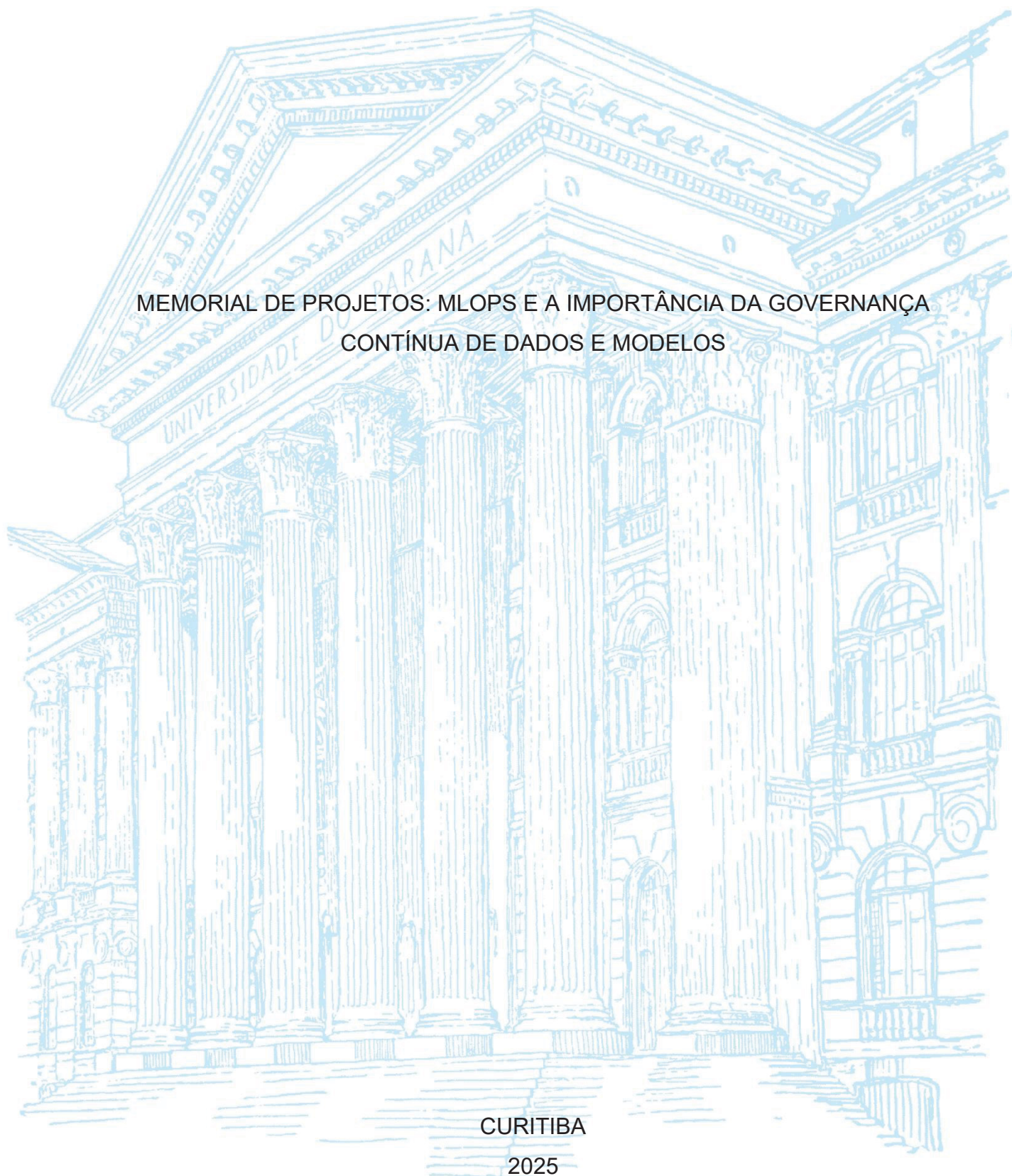
UNIVERSIDADE FEDERAL DO PARANÁ

RAPHAEL PEREIRA RODRIGUES

MEMORIAL DE PROJETOS: MLOPS E A IMPORTÂNCIA DA GOVERNANÇA
CONTÍNUA DE DADOS E MODELOS

CURITIBA

2025



RAPHAEL PEREIRA RODRIGUES

MEMORIAL DE PROJETOS: MLOPS E A IMPORTÂNCIA DA GOVERNANÇA
CONTÍNUA DE DADOS E MODELOS

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

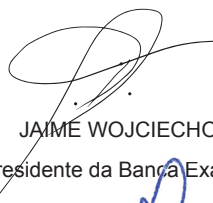
2025

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 **RAPHAEL PEREIRA RODRIGUES**, intitulada: **MEMORIAL DE PROJETOS: MLOPS E A IMPORTÂNCIA DA GOVERNANÇA CONTÍNUA DE DADOS E MODELOS**, 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, 27 de Outubro de 2025.



JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora



RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este memorial de projetos analisa o MLOps (*Machine Learning Operations*) como uma extensão essencial do paradigma DevOps, dedicada ao ciclo de vida completo do Aprendizado de Máquina (ML). O MLOps tem como principal função orquestrar a complexa tríade formada por código, dados e modelos, por meio da implementação de *pipelines* de integração e entrega contínuas (CI/CD). Esse processo exige uma automação rigorosa das etapas de Engenharia de Dados e Pré-Processamento, assegurando a consistência do *Feature Engineering* (construção de características) entre os ambientes de treinamento e inferência. Embora técnicas de automação, como o AutoML, representem ferramentas valiosas, destaca-se a importância de um fluxo de trabalho híbrido, que combine a eficiência dos sistemas automatizados com a expertise humana — indispensável em atividades de maior complexidade, como a limpeza de dados e a interpretação de resultados. Além da operacionalização, o MLOps constitui o alicerce da governança contínua de modelos, demandando monitoramento abrangente para detectar a degradação de desempenho, garantir a confiabilidade das previsões e preservar a segurança da infraestrutura. A adoção responsável de Inteligência Artificial requer, ainda, que o MLOps incorpore princípios éticos que assegurem transparência, mitigação de vieses e proteção de dados em produção. Em síntese, o MLOps representa o caminho para a maturidade da Engenharia de *Machine Learning*, ao transformar protótipos experimentais em soluções de IA aplicada com valor sustentável para o mundo real.

Palavras-chave: mlops; devops; aprendizado de máquina; esteira de produção; governança de dados; automação.

ABSTRACT

This project portfolio analyzes MLOps (Machine Learning Operations) as an essential extension of the DevOps paradigm, dedicated to the complete life cycle of Machine Learning (ML). MLOps plays a central role in orchestrating the complex triad of code, data, and models through the implementation of continuous integration and delivery (CI/CD) pipelines. This process requires rigorous automation of Data Engineering and Preprocessing stages, ensuring consistency in Feature Engineering between training and inference environments. Although automation techniques such as AutoML are valuable tools, this study highlights the importance of a hybrid workflow that combines the efficiency of automated systems with human expertise—indispensable for complex activities such as data cleaning and result interpretation. Beyond operationalization, MLOps serves as the foundation for continuous model governance, requiring comprehensive monitoring to detect performance degradation, ensure prediction reliability, and maintain infrastructure security. Responsible adoption of Artificial Intelligence also demands that MLOps incorporate ethical principles to ensure transparency, bias mitigation, and data protection in production. In summary, MLOps represents the path toward the maturity of Machine Learning Engineering, transforming experimental prototypes into applied AI solutions with sustainable real-world value.

Keywords: mlops; devops; machine learning; production pipeline; data governance; automation.

SUMÁRIO

1	PARECER TÉCNICO.....	7
1.1	MLOPS: DEVOPS PARA MACHINE LEARNING EM PRODUÇÃO	8
1.2	OS COMPONENTES ESSENCIAIS DO PIPELINE	9
1.3	GOVERNANÇA DO DADO E DO MODELO	11
	REFERÊNCIAS	13
	APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL	14
	APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA	21
	APÊNDICE C - LINGUAGEM R.....	36
	APÊNDICE D - ESTATÍSTICA APLICADA I	43
	APÊNDICE E - ESTATÍSTICA APLICADA II.....	57
	APÊNDICE F - ARQUITETURA DE DADOS	79
	APÊNDICE G - APRENDIZADO DE MÁQUINA	97
	APÊNDICE H - DEEP LEARNING	108
	APÊNDICE I - BIG DATA.....	134
	APÊNDICE J - VISÃO COMPUTACIONAL	137
	APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA	157
	APÊNDICE L - GESTÃO DE PROJETOS DE IA.....	164
	APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	167
	APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING.....	186
	APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	194

1 PARECER TÉCNICO

Embora o *Machine Learning* (ML) seja um ramo da Inteligência Artificial (IA) com potencial transformador, ele frequentemente enfrenta uma barreira crítica: a transição de modelos validados em laboratório para sistemas robustos em produção e em escala. O desenvolvimento de modelos altamente precisos — seja em tarefas de classificação, regressão ou *deep learning* — pertence ao domínio da pesquisa. Entretanto, a implantação desses modelos em larga escala introduz desafios operacionais e técnicos que a engenharia de *software* tradicional não é capaz de resolver de forma isolada.

É nesse contexto de crescente complexidade que emerge o MLOps (*Machine Learning Operations*), consolidando-se como uma extensão do paradigma DevOps. O MLOps adapta os princípios de automação, *Continuous Integration* (CI) e *Continuous Delivery* (CD) para atender ao ciclo de vida singular do *Machine Learning*. Para o profissional com experiência em desenvolvimento e operações (DevOps), o foco tradicional do CI/CD recai sobre o código-fonte. No entanto, os sistemas de ML introduzem uma complexidade adicional que os diferencia do *software* convencional.

De acordo com Gift e Deza (2021), a complexidade inerente ao ML exige que o MLOps gerencie uma tríade singular em produção: o código, os dados e o modelo treinado. A gestão integrada desses três componentes interdependentes constitui o principal diferencial e o cerne conceitual do MLOps.

A emergência do MLOps é impulsionada por fatores operacionais e de mitigação de risco. Segundo Treveil *et al.* (2020), a adoção do MLOps é motivada pela necessidade de reduzir riscos e lidar com a natureza dinâmica e o volume crescente dos dados. Enquanto o código de um aplicativo tradicional permanece estático até a próxima atualização, o desempenho de um modelo de ML degrada-se com o tempo — fenômeno conhecido como *model drift*.

No contexto da crescente complexidade, Eken *et al.* (2025) destacam que o MLOps surgiu como uma solução fundamental para abordar desafios sociotécnicos na operacionalização de modelos de ML, como a integração, monitoramento contínuo, manutenção e retreinamento de modelos implantados.

Considerando sua evolução e adoção em larga escala, Treveil *et al.* (2020) afirmam que o MLOps representa um estágio de maturidade da Engenharia de *Machine Learning*, fornecendo o arcabouço necessário para a escalabilidade das

soluções de IA aplicada no ambiente corporativo. A adoção dessas práticas é fundamental para garantir rastreabilidade, reprodutibilidade e governança, transformando a experimentação acadêmica em um sistema operacional sustentável.

1.1 MLOPS: DEVOPS PARA MACHINE LEARNING EM PRODUÇÃO

Para o profissional DevOps, a base do trabalho é a construção de pipelines de CI/CD voltados ao código. O MLOps herda essa filosofia de automação e iteração, mas, conforme já discutido, precisa lidar simultaneamente com código, dados e modelos. De acordo com Moreschini (2023), a melhor maneira de alcançar a automação completa no MLOps é criar pipelines capazes de automatizar o processo de desenvolvimento, teste, implantação e monitoramento.

O desafio técnico de colocar ML em produção está em exigir uma plataforma de engenharia robusta, que transcende o simples controle de versão do código. Os sistemas tradicionais de controle de versão, de acordo com Diaz-de-Arcaya *et al.* (2023), não conseguem distinguir entre componentes específicos de ML, como modelos e *datasets*, o que ressalta a necessidade de ferramentas especializadas para gerenciar metainformações e governança de dados e modelos.

Conforme Osipov (2022), apenas cerca de 5% do sucesso de um sistema de ML advém do código de aprendizado de máquina propriamente dito, enquanto os 95% restantes dependem da plataforma de ML subjacente. Essa plataforma deve garantir ingestão e processamento de dados, gerenciamento de recursos, monitoramento e entrega contínua. O MLOps, portanto, configura-se como o modelo de desenvolvimento de *software* que opera sobre essa infraestrutura complexa.

A adaptabilidade e a gestão de mudanças são características centrais que o MLOps deve oferecer. De acordo com Raj (2021), o ML constitui essencialmente a união entre código e dados, em que a aplicação de um algoritmo (código) sobre um conjunto de dados resulta em um modelo que precisa ser mantido em produção. A capacidade de aprendizagem contínua é essencial, pois a inteligência do sistema está em sua habilidade de adaptar-se dinamicamente ao ambiente externo.

Os dados representam a principal fonte de volatilidade e complexidade dentro de um pipeline MLOps. A qualidade e a quantidade de dados influenciam, muitas vezes, o desempenho do modelo de maneira mais significativa do que a mera otimização algorítmica. Em um ambiente MLOps, a ingestão, limpeza e transformação

de dados devem ser processos automatizados e versionados com o mesmo rigor aplicado ao código.

Um elemento particularmente crítico da engenharia de dados em MLOps é o *Feature Engineering*. Essa etapa transforma dados brutos em atributos expressivos utilizados pelo modelo. O MLOps deve garantir a consistência da engenharia de features, pois qualquer divergência entre o cálculo dos atributos nos ambientes de treinamento e inferência resulta em degradação do desempenho do modelo. Conforme Moreschini (2023), a rastreabilidade de todas as alterações relacionadas aos dados de entrada é fundamental. O estabelecimento de *Feature Stores* centralizadas é uma prática recomendada para promover o reuso e a consistência das *features* entre a ciência de dados e as operações.

Apesar da ascensão do AutoML (*Automated Machine Learning*), que busca otimizar a seleção de modelos e a sintonia de hiperparâmetros, o MLOps defende uma abordagem híbrida. Segundo Xin *et al.* (2021), ferramentas de AutoML ainda carecem de suporte adequado para tarefas de pré e pós-processamento, reforçando a importância da intervenção humana em etapas críticas. O MLOps, inclusive, defende que a automação seja aplicada em tarefas tediosas, mas que *insights* humanos sejam incorporados onde necessário, especialmente para considerações éticas e de responsabilidades, conforme aponta Eken *et al.* (2025).

1.2 OS COMPONENTES ESSENCIAIS DO PIPELINE

No desenvolvimento de *software* tradicional, a *Continuous Integration* (CI) concentra-se na validação do código. Em MLOps, o pipeline de CI/CD deve estender essa validação também aos dados. De acordo com Raj (2021), o ML é, em essência, a união entre código e dados, o que torna a governança dos dados um pilar fundamental. Treveil *et al.* (2020) destacam que, antes mesmo da modelagem, é essencial priorizar a identificação de fontes de dados e a Análise Exploratória de Dados (EDA), seguida pela engenharia e seleção de atributos.

O MLOps gerencia tanto essa transformação quanto o consumo desses atributos. Gift e Deza (2021) apontam que o *Feature Store* — ou repositório de atributos — é um componente central do KaizenML, disciplina voltada à melhoria contínua da qualidade de dados e modelos. O *Feature Store* atua como um registro

de entradas de alta qualidade, promovendo o reuso e garantindo consistência entre os ambientes de treinamento e inferência.

Ao selecionar atributos, o engenheiro de MLOps deve seguir princípios rigorosos. Conforme Osipov (2022), um princípio orientador é assegurar que o valor de cada feature esteja disponível antes do tempo de inferência — ou seja, no momento em que o modelo é executado em produção. Essa prática previne um erro comum: utilizar atributos que existem no treinamento, mas não estão acessíveis durante a inferência.

Após a preparação e gestão das features, o MLOps automatiza a etapa de treinamento. Diferentemente do DevOps tradicional, o pipeline MLOps engloba também o *Continuous Training* (CT) e o monitoramento contínuo para mitigar o *model drift*. O *Continuous Training* requer a gestão sistemática de experimentos, que é facilitada por plataformas de rastreamento como o MLflow. Cada execução representa um experimento distinto, possibilitando reprodutibilidade e comparação:

- Automação e Otimização de Hiperparâmetros (HPO): o *pipeline* MLOps inclui a *Hyperparameter Optimization* (HPO), aplicável às etapas de preparação de dados, *feature engineering* e treinamento do modelo. Conforme Osipov (2022), ferramentas como Optuna podem ser integradas ao pipeline, permitindo a execução paralela de múltiplos modelos e a seleção do melhor com base em métricas de desempenho.

- Gestão de Experimentos (MLFlow): O rastreamento de experimentos é fundamental para a reprodutibilidade. Plataformas como o MLflow permitem armazenar e organizar diferentes instâncias de treinamento, promovendo o controle e o versionamento.

- Uso de Contêineres (Docker): Gift e Deza (2021) destacam que o empacotamento de modelos de ML em contêineres Docker é uma estratégia essencial para compartilhamento, distribuição e implantação consistente, independentemente do sistema operacional subjacente. Contêineres são frequentemente utilizados para entregar soluções de ML e são essenciais para orquestração e escalabilidade, como via Kubernetes. O Docker foi usado, por exemplo, em *frameworks* de *Edge* MLOps, conforme demonstra Moreschini (2023).

- Artefatos de ML: Conforme Treveil *et al.* (2020), um artefato de ML completo deve incluir o código, hiperparâmetros, dados de treinamento e validação, o modelo treinado em formato executável e o ambiente de *runtime*.

- Implantação: Os modelos empacotados são disponibilizados como APIs ou microserviços e podem ser implantados em diferentes ambientes, como instâncias de contêineres ou clusters orquestrados.

Esses componentes asseguram que o MLOps sustente um fluxo de trabalho integrado (CI/CD/CT) capaz de mover não apenas o código, mas também os dados e os modelos, do desenvolvimento à produção, com velocidade, rastreabilidade e segurança.

1.3 GOVERNANÇA DO DADO E DO MODELO

A governança em MLOps não constitui apenas uma diretriz administrativa, mas um pilar de engenharia que garante a sustentabilidade e a confiabilidade dos sistemas de ML em produção. Segundo Eken *et al.* (2025), a governança responsável (*Responsible MLOps*) é um esforço multidisciplinar que une pessoas, tecnologia e processos. O desempenho de qualquer modelo de IA — seja de classificação, regressão ou *deep learning* — depende criticamente da qualidade dos dados de entrada.

Conforme Raj (2021), a eficácia de um sistema de ML é determinada por sua performance robusta, e a governança é elemento indispensável para esse fim. Em contextos de *Big Data*, o MLOps precisa mitigar o risco de uso de dados inadequados em produção. Isso requer controle de qualidade rigoroso e versionamento dos conjuntos de dados, incluindo a verificação de legalidade, qualidade, segurança e privacidade de dados, sendo a validação de dados fundamental para a saúde do sistema, conforme aponta Eken *et al.* (2025).

A governança deve assegurar, por exemplo, a consistência da engenharia de *features* entre os ambientes de treinamento e inferência. Diferentemente do *software* convencional, o sistema de ML é dinâmico: ele aprende padrões a partir de dados históricos e pode perder acurácia quando o contexto de entrada se altera. Esse fenômeno, conhecido como *model drift*, constitui um dos principais riscos operacionais que o MLOps deve endereçar.

Raj (2021) observa que a eficácia de um sistema de ML depende de ações inteligentes baseadas em monitoramento e alertas. Assim, a governança requer controle rigoroso sobre as métricas de desempenho do modelo em produção. O monitoramento deve abranger tanto métricas de ML quanto indicadores de qualidade

de dados, auditoria e emissão de relatórios, garantindo a estabilidade operacional e a observabilidade, como aponta Diaz-de-Arcaya *et al.* (2023).

Quando a degradação do modelo é identificada, a governança deve acionar mecanismos automatizados de *Continuous Training* (CT) e reimplantação, preservando a performance e a confiabilidade do sistema. Moreschini *et al.* (2023) aponta que o monitoramento deve ser capaz de invocar o retreinamento do modelo, e o *Continuous Deployment* (CD) é responsável por implantar o sistema quando os resultados superam o treinamento anterior, facilitando a automação de CI/CD.

A crescente complexidade dos sistemas modernos de IA, como o ChatGPT, traz à tona preocupações éticas e técnicas relacionadas a vieses algorítmicos e inclusividade. Segundo Treveil *et al.* (2020), a governança em MLOps precisa incluir mecanismos para monitorar viés, garantir transparência e promover práticas responsáveis de IA em escala. Conforme Eken *et al.* (2025), o MLOps responsável engloba a conformidade com regulamentos, mitigação de viés e transparência, sendo consideradas questões *cross-silo* que devem ser integradas em todo o ciclo de vida.

Em síntese, a governança contínua de dados e modelos é indispensável para assegurar a integridade, a explicabilidade e a sustentabilidade das soluções baseadas em Machine Learning em ambientes produtivos.

REFERÊNCIAS

DIAZ-DE-ARCAYA, J.; TORRE-BATISDA, A. I.; ZÁRATE, G.; MIÑÓN, R.; ALMEIDA, A. **A Joint Study of the Challenges, Opportunities, and Roadmap of MLOps and AIOps: A Systematic Survey**. ACM Computing Surveys, 21 de outubro, 2023. DOI <https://doi.org/10.1145/3625289>

EKEN, B.; PALLEWATTA, S.; TRAN, N.; AYSE, T.; BABAR, M. A. **A Multivocal Review of MLOps Practices, Challenges and Open Issues**. ACM Computing Surveys, 8 de setembro, 2025. DOI <https://doi.org/10.1145/3747346>

GIFT, N.; DEZA, A. **Practical MLOps: Operationalizing Machine Learning Models**. Sebastopol, CA: O'Reilly Media, 2021

MORESCHINI, S.; HASTBACKA, D.; TAIBI, D. **MLOps Pipeline Development: The OSSARA Use Case**. RACS '23: Proceedings of the 2023 International Conference on Research in Adaptive and Convergent Systems, 29 de agosto, 2023. DOI <https://doi.org/10.1145/3599957.3606211>

OSIPOV, C. **MLOps Engineering at Scale**. Manning, 2022

RAJ, E. **Engineering MLOps: Rapidly build, test, and manage production-ready machine learning life cycles at scale**. Birmingham: Packt Publishing, 2021

TREVEIL, M.; OMONT, N.; STENAC, C.; LEFREVRE, K.; PHAN, D.; ZENTICI, J.; LAVOILLOTTE, A.; MIYAZAKI, M.; HEIDMANN, L. **Introducing MLOps: How to Scale Machine Learning in the Enterprise**. Sebastopol, CA: O'Reilly Media, 2020

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>

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

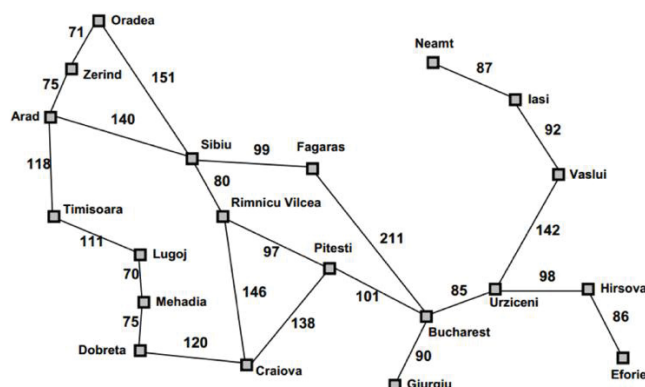
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

Se as uvas caem, então a raposa as come

Se a raposa as come, então estão maduras

As uvas estão verdes ou caem

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem

q: a raposa come as uvas

r: as uvas estão maduras

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

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

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

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

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

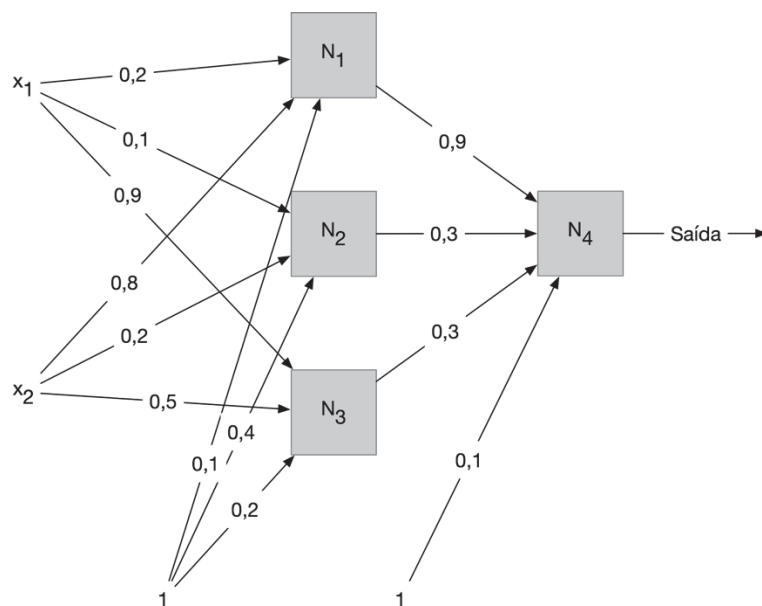
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

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

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

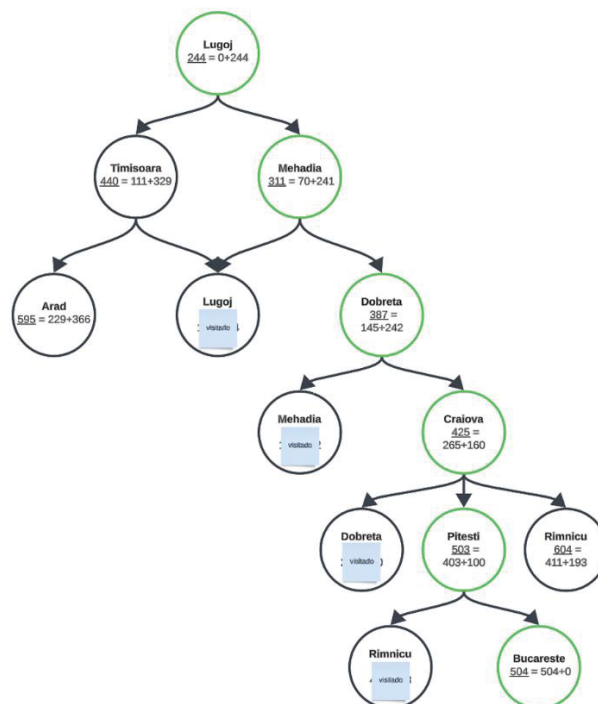
B – RESOLUÇÃO

1. ChatGPT

- a) Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e programas de computador capazes de realizar tarefas que normalmente requerem inteligência humana. Essas tarefas incluem reconhecimento de fala, visão computacional, tomada de decisão, aprendizado e resolução de problemas. A IA visa criar sistemas que possam simular a capacidade humana de raciocínio, aprendizado, adaptação e compreensão de contextos complexos. Isso é frequentemente alcançado por meio de algoritmos e modelos matemáticos que permitem aos computadores reconhecerem padrões em grandes conjuntos de dados, aprender com esses padrões e tomar decisões com base neles.
- b) Pensar como humanos: A IA busca simular a capacidade humana de raciocínio e compreensão de contextos complexos. Algoritmos e modelos matemáticos permitem aos computadores reconhecerem padrões em dados, similar ao processo de aprendizado humano. Agir como humanos: A IA é aplicada em tarefas como reconhecimento de fala, visão computacional e tomada de decisão, imitando habilidades humanas. A IA pode gerar textos e imagens que se assemelham à produção humana, criando a ilusão de inteligência. Pensar racionalmente: A IA utiliza algoritmos e modelos para analisar dados e tomar decisões, similar ao raciocínio lógico humano. A IA pode aprender com a experiência e ajustar suas ações com base em resultados anteriores, demonstrando capacidade de adaptação. Agir racionalmente: A IA é direcionada para alcançar objetivos específicos, como fornecer informações ou resolver problemas. A IA pode otimizar seus processos e recursos para alcançar a máxima eficiência na execução de tarefas.
- c) O ChatGPT é uma tecnologia inovadora que utiliza Machine Learning para gerar respostas a partir de perguntas e respostas anteriores. Ele pode gerar textos, traduzir idiomas, responder perguntas e manter diálogos, tudo de forma natural e informativa. O ChatGPT pode ser aplicado em diversas áreas, como atendimento ao cliente, marketing, educação e pesquisa. Ele pode automatizar tarefas repetitivas, criar conteúdo personalizado e analisar dados, liberando tempo para atividades mais complexas e de alto valor agregado. Os benefícios do ChatGPT incluem eficiência, precisão, personalização e escalabilidade. Ele O ChatGPT pode lidar com um grande volume de usuários ao mesmo tempo, sem sacrificar a qualidade das suas respostas. Mesmo com uma grande demanda, o ChatGPT mantém a qualidade das suas interações com os usuários. O ChatGPT é capaz de escalar para atender a muitos usuários sem que a qualidade das suas respostas seja afetada. No entanto, o ChatGPT também apresenta alguns desafios, como vieses, segurança e interpretação dos dados. É importante estar ciente desses desafios e tomar medidas para mitigá-los. O futuro do ChatGPT é promissor. Ele tem o potencial de revolucionar a maneira como interagimos com computadores e automatizar tarefas ainda mais complexas.

- d) Pensar como humanos: O ChatGPT utiliza Machine Learning para aprender padrões de linguagem e gerar respostas que imitam a maneira como os seres humanos escrevem e comunicam. Apesar de imitar a escrita humana, o ChatGPT não possui consciência ou real compreensão do que escreve, operando apenas com base em dados. Agir como humanos: O ChatGPT gera textos, traduz idiomas, responde perguntas e mantém diálogos de forma natural e informativa, imitando habilidades humanas. Em muitos casos, principalmente em conversas sobre tópicos conhecidos, pode ser difícil discernir se o usuário está interagindo com um humano ou com o ChatGPT. Pensar racionalmente: O ChatGPT busca a melhor resposta possível com base em seu treinamento, utilizando algoritmos que analisam dados e reconhecem padrões. O ChatGPT não "pensa" no sentido tradicional, mas aplica raciocínio lógico e matemático para tomar decisões. Agir racionalmente: O ChatGPT é direcionado a objetivos específicos, como fornecer informações relevantes e precisas ou automatizar tarefas repetitivas. O ChatGPT otimiza seus processos e recursos para alcançar a máxima eficiência na execução de tarefas. Porque: O ChatGPT imita a escrita e comunicação humana, mas sem consciência ou compreensão real. Ele gera textos, traduz idiomas, responde perguntas e mantém diálogos, dificultando a distinção entre humano e máquina em algumas conversas. O ChatGPT busca a melhor resposta com base em dados e algoritmos, utilizando raciocínio lógico e matemático. Ele se concentra em objetivos como fornecer informações ou automatizar tarefas, otimizando seus processos para alcançar eficiência. Resumindo: O ChatGPT parece humano, mas não é. Ele toma decisões racionais e eficientes. Ele pode ser usado para diversas tarefas.

2. Busca Heurística



a)

Lugoj > Mehadia > Dobreta > Craiova > Pitesti > Bucareste

3. Lógica

a) A base de conhecimento é:

1. R1: $p \rightarrow q$
2. R2: $q \rightarrow r$
3. R3: $\neg r \vee p$

Passo 1: Aplicar a equivalência da implicação em R1 e R2

R1 se torna $\neg p \vee q$

R2 se torna $\neg q \vee r$

Passo 2: Derivar $p \rightarrow q$ da base de conhecimento

Isso já é dado por R1

Passo 3: Derivar $q \rightarrow p$

Para fazer isso, precisamos considerar R3 e a implicação $\neg q \rightarrow \neg r$, que é a contrapositiva de R2.

Passo 4: Aplicar a regra do silogismo hipotético

$\neg r \rightarrow p$ (diretamente de R3)

$\neg q \rightarrow \neg r$ (contrapositiva de R2)

Combinando as duas, obtemos $\neg q \rightarrow p$

Passo 5: Simplificar para a forma bicondicional

Sabemos que $q \rightarrow p$ e $p \rightarrow q$ (de R1 e Passo 4).

A equivalência bicondicional nos diz que se temos $\alpha \rightarrow \beta$ e $\beta \rightarrow \alpha$, então podemos formar $\alpha \leftrightarrow \beta$.

Conclusão:

$Q \leftrightarrow p$ é provado pela combinação de R1 e a contrapositiva de R2 com a informação de R3.

Formato da Prova:

4. R1: $p \rightarrow q$ (Dado)

5. R2: $q \rightarrow r$ (Dado)

6. R3: $\neg r \vee p$ (Dado)

7. R2 Contrapositiva: $\neg r \rightarrow \neg q$

8. R4: $\neg q \rightarrow p$ (Combinando R3 e a contrapositiva de R2)

9. R5: $q \leftrightarrow p$ (Combinando R1 e R4 com a regra da bicondicional)

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

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

B - RESOLUÇÃO

1. Análise Exploratória dos dados

a)

```
dados = pd.read_csv('./precos_carros_brasil.csv')
```

b)

```
dados.head()
```

```
dados.shape
```

```
dados.isna().any()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
0	2021	January	004001-0	cfzictzlwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2002.0	9162.0
1	2021	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2001.0	8832.0
2	2021	January	004001-0	cb1f3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000.0	8388.0
3	2021	January	004001-0	cb9gcl6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000.0	8453.0
4	2021	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline	manual	1.6	2001.0	12525.0

(47607, 11)

	0
year_of_reference	False
month_of_reference	False
fipe_code	False
authentication	False
brand	True
model	True
fuel	True
gear	True
engine_size	True
year_model	True
avg_price_brl	True

dtype: bool

```
dados.isna().sum()
```

	0
year_of_reference	0
month_of_reference	0
fipe_code	0
authentication	0
brand	1
model	1
fuel	1
gear	1
engine_size	1
year_model	1
avg_price_brl	1

dtype: int64

```
print(round(dados.isnull().sum().sort_values(ascending=False)/dados.shape[0],2))
```

```
fuel      0.0
model     0.0
brand     0.0
year_model 0.0
engine_size 0.0
avg_price_brl 0.0
gear      0.0
fipecode  0.0
month_of_reference 0.0
year_of_reference 0.0
authentication 0.0
dtype: float64
```

```
dados = dados.dropna()
dados.isnull().sum()
```

```
0
year_of_reference 0
month_of_reference 0
fipecode 0
authentication 0
brand 0
model 0
fuel 0
gear 0
engine_size 0
year_model 0
avg_price_brl 0
dtype: int64
```

c)

```
dados.duplicated().sum()
```

```
np.int64(1)
```

```
dados.drop_duplicates(inplace=True)
dados.shape
```

```
(47605, 11)
```

d)

```
numérica_cols = [col for col in dados.columns if dados[col].dtype !=
'object']
categoricas_cols = [col for col in dados.columns if dados[col].dtype ==
'object']
dados[numérica_cols].describe()
```

	year_of_reference	year_model	avg_price_brl
count	47605.0	47605.000000	47605.000000
mean	2021.0	2011.041277	43809.420649
std	0.0	6.222069	42125.345951
min	2021.0	2000.000000	6647.000000
25%	2021.0	2006.000000	19758.000000
50%	2021.0	2011.000000	32537.000000
75%	2021.0	2016.000000	53284.000000
max	2021.0	2022.000000	860215.000000

```
dados[categoricas_cols].describe()
```

	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size
count	47605	47605	47605	47605	47605	47605	47605	47605
unique	6	2029	47605	6	2030	3	2	29
top	May	003281-6	5q41w3fw858q	Fiat	Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	Gasoline	manual	1,6
freq	7982	102	1	10645	102	39801	38511	11268

e)

```
dados['model'].value_counts()
```

model	
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	425
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	425
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	400
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	400
Doblo Adv/Adv TRYON/LOCKER 1.8 Flex	375
...	...
Polo Track 1.0 Flex 12V 5p	2
STEPWAY Zen Flex 1.0 12V Mec.	2
Saveiro Robust 1.6 Total Flex 16V CD	2
Saveiro Robust 1.6 Total Flex 16V	2
Gol Last Edition 1.0 Flex 12V 5p	2

2112 rows x 1 columns

```
dados['brand'].value_counts()
```

	count
brand	
Fiat	44962
VW - VolksWagen	44312
GM - Chevrolet	38590
Ford	33150
Renault	29191
Nissan	12090

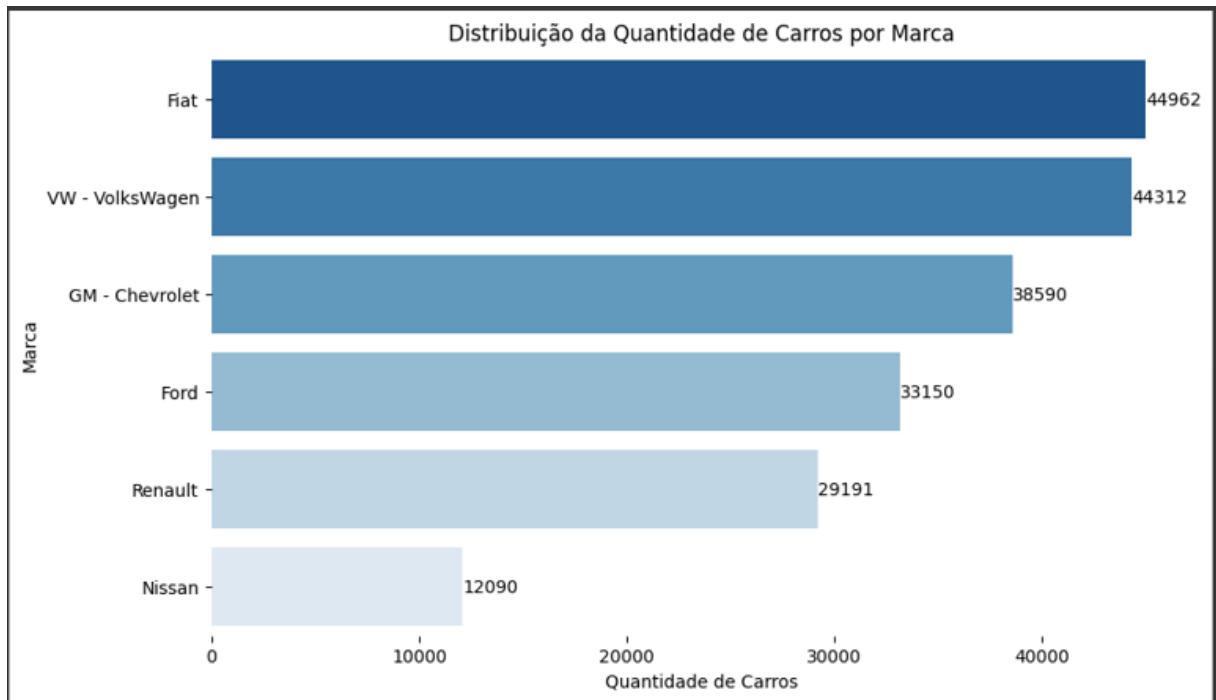
f) Removendo os dados inválidos e duplicados, conseguimos identificar que a marca mais vendida no Brasil é Fiat (44962), seguido por VW – VolksWagen (44312) e Chevrolet (385590) respectivamente. A marca Ford ficou em 4º lugar, mas tem seu modelo Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p em primeiro lugar com 425 vendas, mesma quantidade de vendas do Palio Week. Adv/Adv TRYON 1.8 mpi Flex da Fiat.

2. Visualização de Dados

a)

```
import matplotlib.pyplot as plt
import seaborn as sns

# Gráfico da distribuição da quantidade de carros por marca
fig, ax = plt.subplots(figsize=(10, 6))
ax = sns.countplot(y="brand", data=dados, order =
dados['brand'].value_counts().index, palette="Blues_r")
for i in ax.containers:
    ax.bar_label(i)
ax.set_title('Distribuição da Quantidade de Carros por Marca')
ax.set_xlabel('Quantidade de Carros')
ax.set_ylabel('Marca')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
plt.show()
```



b)

Gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro

```
plt.figure(figsize=(8, 7))
```

```
ax = sns.countplot(x="gear", data=dados, order =  
dados['gear'].value_counts().index, palette='Blues')
```

```
for i in ax.containers:
```

```
    ax.bar_label(i)
```

```
ax.set_title('Distribuição da Quantidade de Carros por Tipo de Engrenagem')
```

```
ax.set_xlabel('Tipo de Engrenagem')
```

```
ax.set_ylabel('Quantidade de Carros')
```

```
ax.spines['top'].set_visible(False)
```

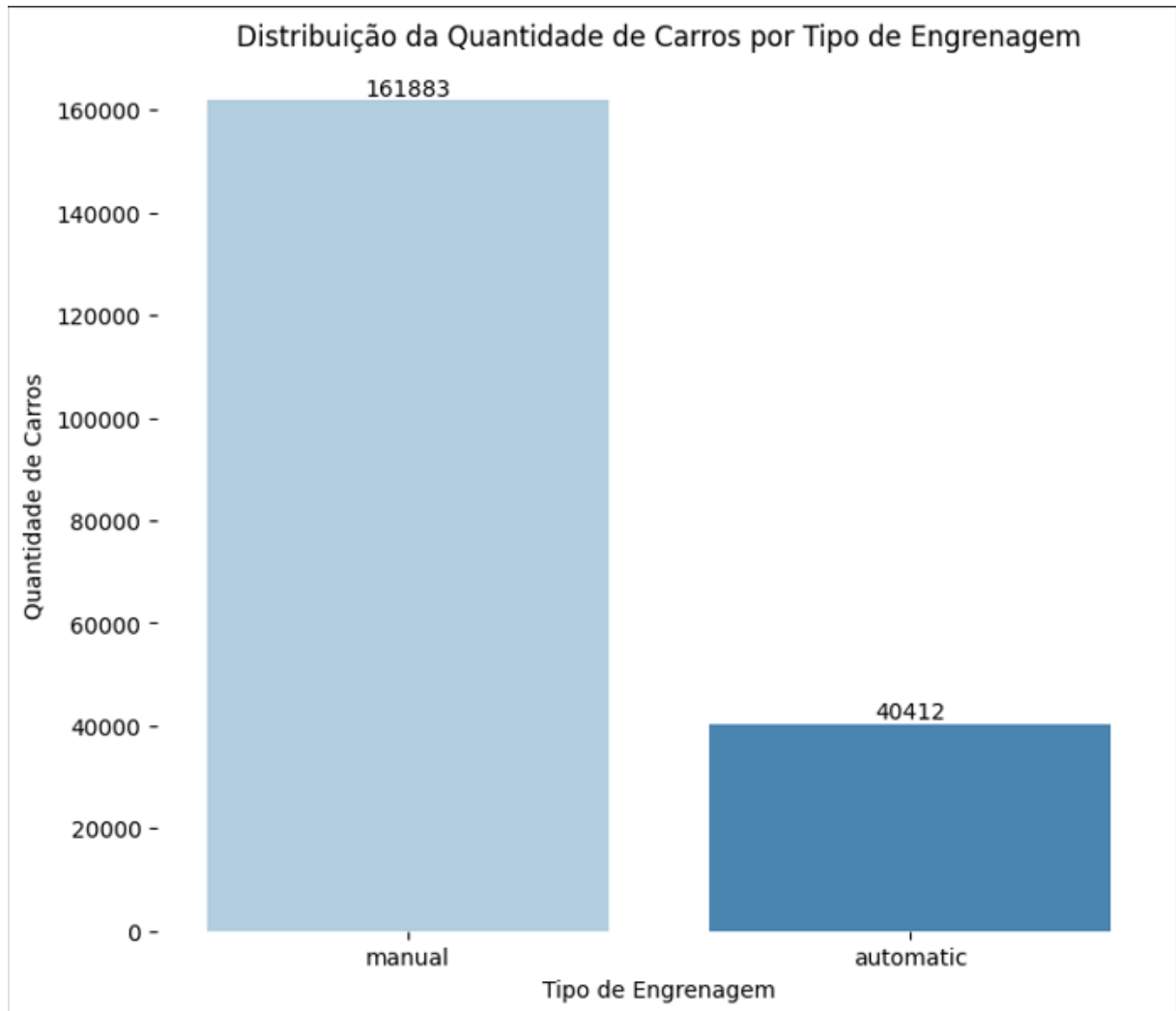
```
ax.spines['right'].set_visible(False)
```

```
ax.spines['bottom'].set_visible(False)
```

```
ax.spines['left'].set_visible(False)
```

```
plt.show()
```

```
plt.clf()
```

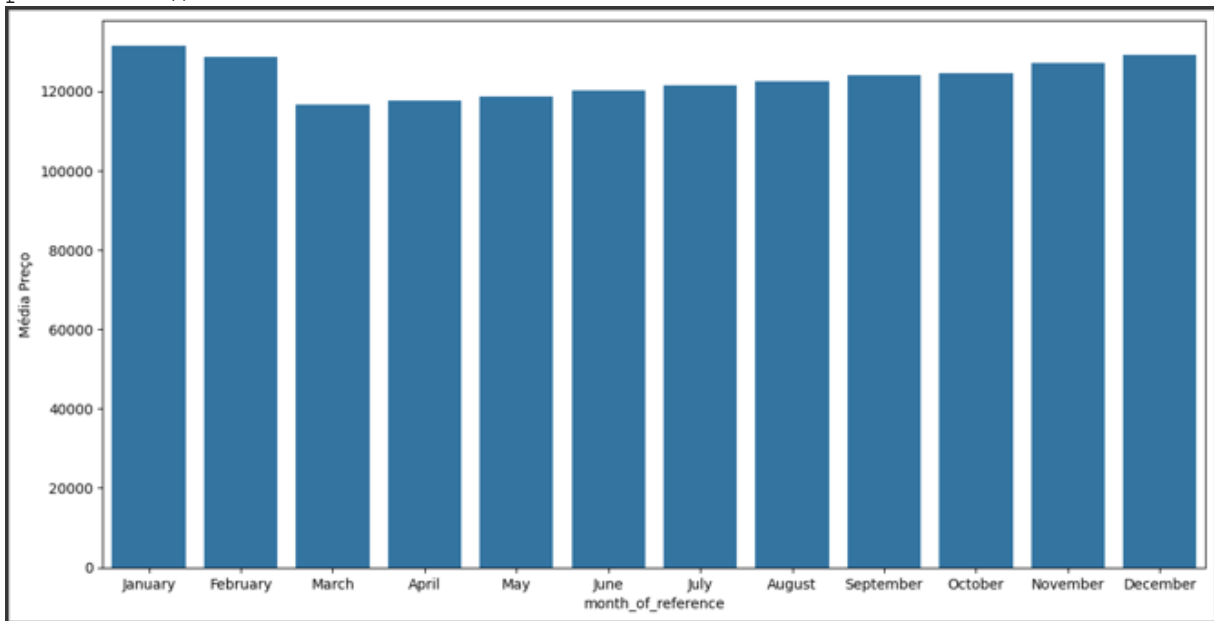


c)

```
media_preco = dados.groupby(['year_model',
                             'month_of_reference'])['avg_price_brl'].mean().round(0)
media_preco_2022 = media_preco[2022.0]
media_preco_2022 = media_preco_2022.reset_index(name='Média Preço')
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
          'August', 'September', 'October', 'November', 'December']
media_preco_2022['month_of_reference'] =
pd.Categorical(media_preco_2022['month_of_reference'], categories=months,
               ordered=True)
media_preco_2022.sort_values(by='month_of_reference', inplace=True)
media_preco_2022.head()
```

	month_of_reference	Média Preço
4	January	131339.0
3	February	128550.0
7	March	116449.0
0	April	117703.0
8	May	118526.0

```
plt.figure(figsize=(14,7))
sns.barplot(y='Média Preço', x='month_of_reference',
data=media_preco_2022)
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
plt.show()
```



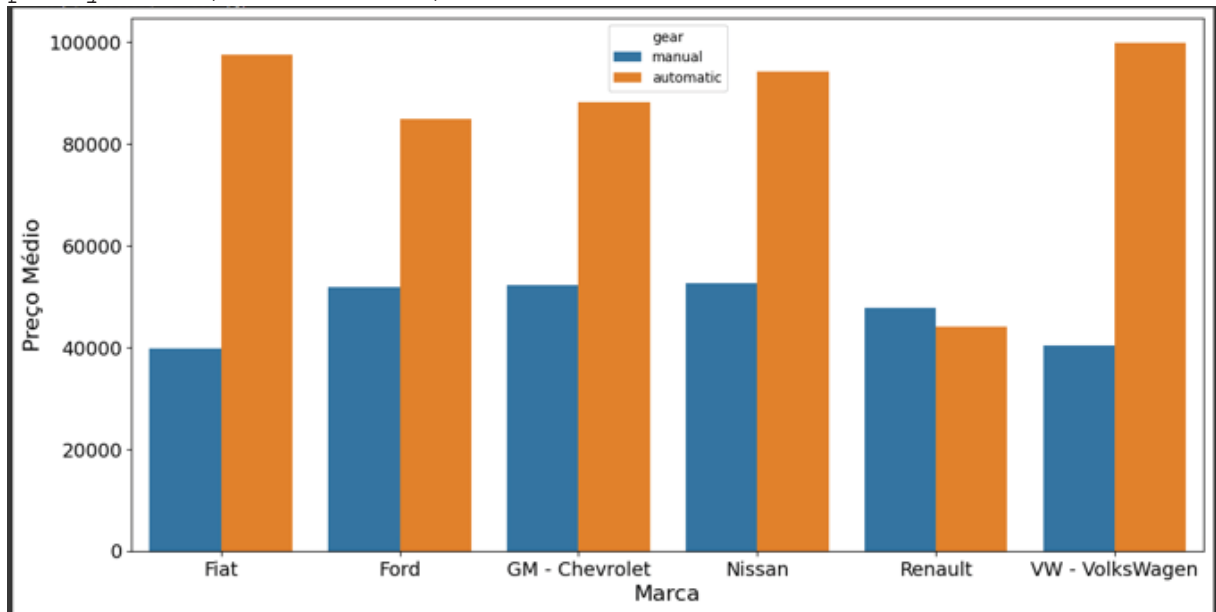
```
d)
media_preco_brand_gear = dados.groupby(['brand',
'gear'])['avg_price_brl'].mean().round(0)
media_preco_brand_gear = media_preco_brand_gear.reset_index(name='Média
Preço')
media_preco_brand_gear.head()
```

	brand	gear	Média Preço
0	Fiat	automatic	97397.0
1	Fiat	manual	39694.0
2	Ford	automatic	84769.0
3	Ford	manual	51784.0
4	GM - Chevrolet	automatic	88157.0

```
plt.figure(figsize=(14,7))
sns.barplot(x='brand', y='Média Preço', hue='gear',
data=media_preco_brand_gear, hue_order=['manual',
'automatic'])
```

```
plt.ylabel('Preço Médio', fontsize=16)
plt.xlabel('Marca', fontsize=16)
```

```
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
```



e) Os carros com engrenagem automática têm um valor médio maior que os carros com engrenagem manual, exceto da marca Renault. Os carros automáticos da VW, Fiat e Nissan são os mais caros. Os carros com engrenagem manual da Renault são mais caros que os automáticos.

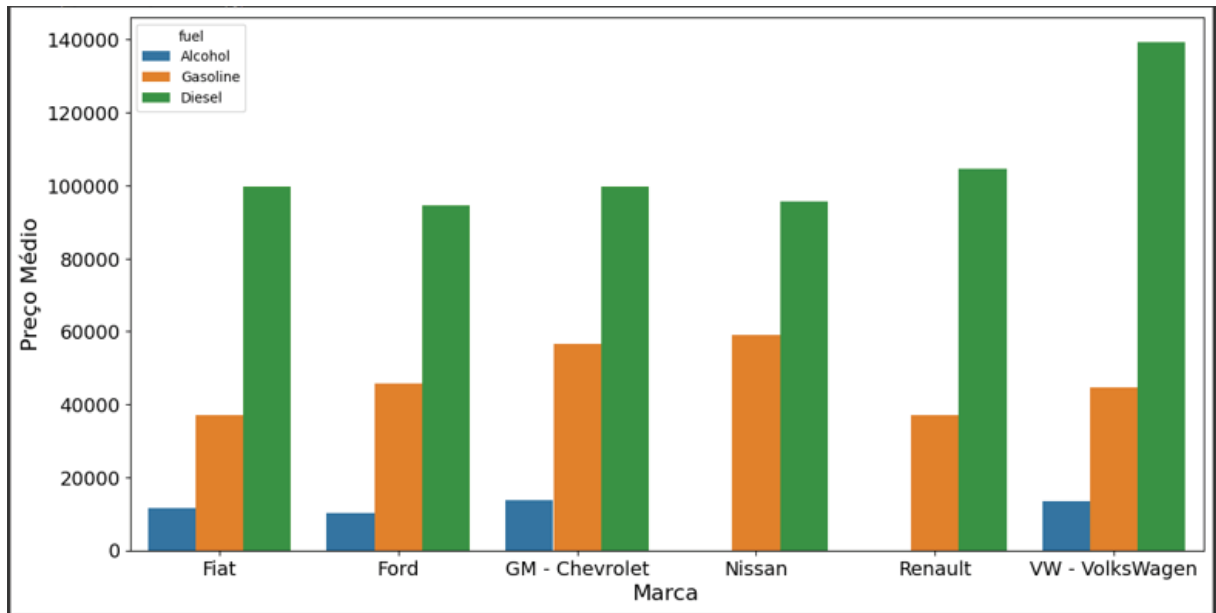
```
f)
media_preco_brand_fuel = dados.groupby(['brand',
'fuel'])['avg_price_brl'].mean().round(0)
media_preco_brand_fuel = media_preco_brand_fuel.reset_index(name='Média
Preço')
media_preco_brand_fuel.head()
```

	brand	fuel	Média Preço
0	Fiat	Alcohol	11510.0
1	Fiat	Diesel	99814.0
2	Fiat	Gasoline	37197.0
3	Ford	Alcohol	10149.0
4	Ford	Diesel	94526.0

```
plt.figure(figsize=(14,7))
sns.barplot(x='brand', y='Média Preço', hue='fuel',
            data=media_preco_brand_fuel, hue_order=['Alcohol', 'Gasoline',
'Diesel'])
```

```
plt.ylabel('Preço Médio', fontsize=16)
plt.xlabel('Marca', fontsize=16)
```

```
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
```

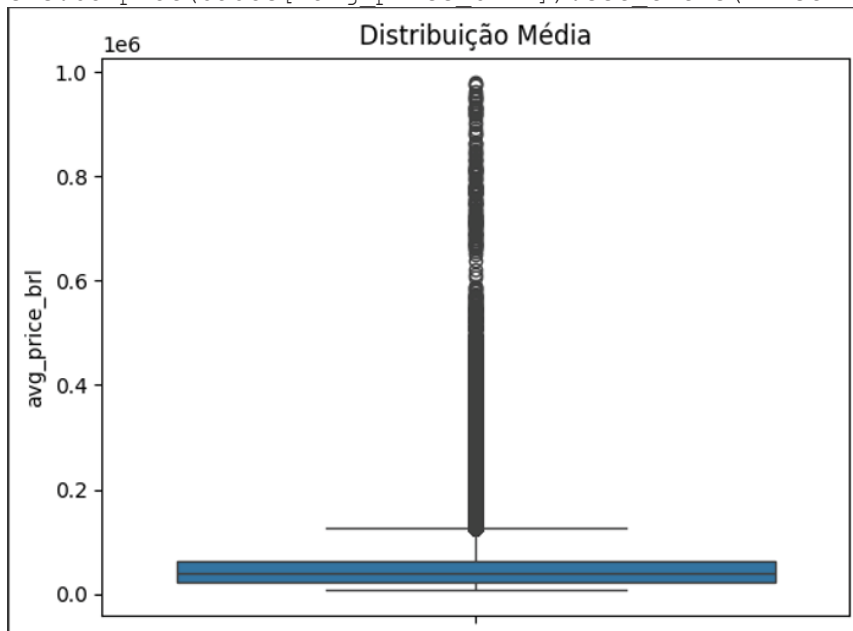



g) Os carros a diesel são os mais caros. Nissan e Renault não tem carro a álcool. Carro a álcool são os mais baratos comparados a outros combustíveis.

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

a)

```
# Analise da variavel target (avg_price_brl)
sns.boxplot(dados['avg_price_brl']).set_title("Distribuição Média")
```



```
# Transformando categoricas em numericas
dados['gear'].unique()
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])
dados['fuel'].unique()
dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])
dados['brand'].unique()
dados['brand'] = LabelEncoder().fit_transform(dados['brand'])
dados.head()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
0	2021.0	January	004001-0	ctzktzlwrcp	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2002.0	9162.0
1	2021.0	January	004001-0	cdqwwpw3y2p	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2001.0	8832.0
2	2021.0	January	004001-0	cb113xwwj1xp	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2000.0	8388.0
3	2021.0	January	004001-0	cb9gcl6j65r0	2	Corsa Wind 1.0 MPFI / EFI 2p	0	1	1	2000.0	8453.0
4	2021.0	January	004003-7	g15wg0gbz1fx	2	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	2	1	1.6	2001.0	12525.0

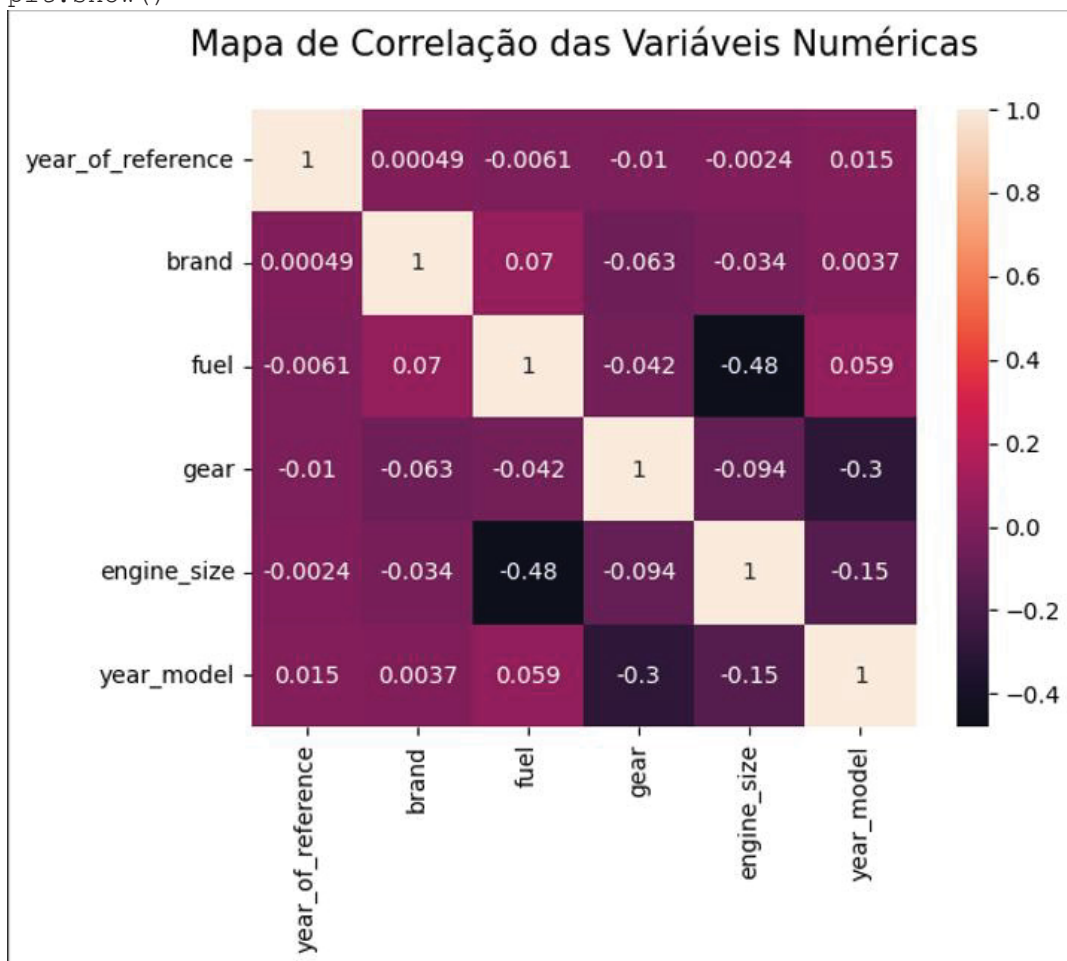
```
# Transformando 'engine_size' em numérico
dados['engine_size'] = dados['engine_size'].str.replace(',',
'.').astype(float)

# Definindo as variáveis independentes (X) e a variável dependente/target (y)
X = dados.drop(['fipe_code', 'authentication', 'model', 'avg_price_brl',
'month_of_reference'], axis=1)
y = dados['avg_price_brl']
```

```
X.head()
```

	year_of_reference	brand	fuel	gear	engine_size	year_model
0	2021.0	2	2	1	1.0	2002.0
1	2021.0	2	2	1	1.0	2001.0
2	2021.0	2	2	1	1.0	2000.0
3	2021.0	2	0	1	1.0	2000.0
4	2021.0	2	2	1	1.6	2001.0

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



b)

```
# Criando partições de dados para treino e teste (75% treino, 25% teste)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
```

c)

```
# Ajustando a criação de dummies
categorical_features = ['fuel', 'gear', 'brand', 'engine_size']

# Para treino
X_train = pd.get_dummies(X_train, columns=categorical_features,
drop_first=True)

# Para teste, garantindo que as mesmas colunas dummy sejam criadas
X_test = pd.get_dummies(X_test, columns=categorical_features,
drop_first=True)

X_test = X_test.reindex(columns=X_train.columns, fill_value=0)

# Verificando as dimensões
X_train.shape, X_test.shape, y_train.shape, y_test.shape
((151721, 38), (50574, 38), (151721,), (50574,))
```

d)

```
# Treinamento do modelo RandomForest
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)

# Predições com o modelo RandomForest
y_pred_rf = rf_model.predict(X_test)

# Treinamento do modelo XGBoost
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(X_train, y_train)

# Predições com o modelo XGBoost
y_pred_xgb = xgb_model.predict(X_test)

# Avaliando os modelos
# metrics_rf
print("Modelo RandomForest")
print("MSE:", mean_squared_error(y_test, y_pred_rf)),
print("MAE:", mean_absolute_error(y_test, y_pred_rf)),
print("R2:", r2_score(y_test, y_pred_rf))

# metrics_xgb
print("\nModelo XGBoost")
print("MSE:", mean_squared_error(y_test, y_pred_xgb)),
print("MAE:", mean_absolute_error(y_test, y_pred_xgb)),
print("R2:", r2_score(y_test, y_pred_xgb))
```

```
Modelo RandomForest
MSE: 106831121.7836354
MAE: 5597.930386761915
R2: 0.9603043823173815
```

```
Modelo XGBoost
MSE: 111065512.99152356
MAE: 5861.131681702261
R2: 0.9587309946031964
```

```
# Cross-validation
from sklearn.model_selection import cross_val_score

xgb_model_cross = XGBRegressor(random_state=42)

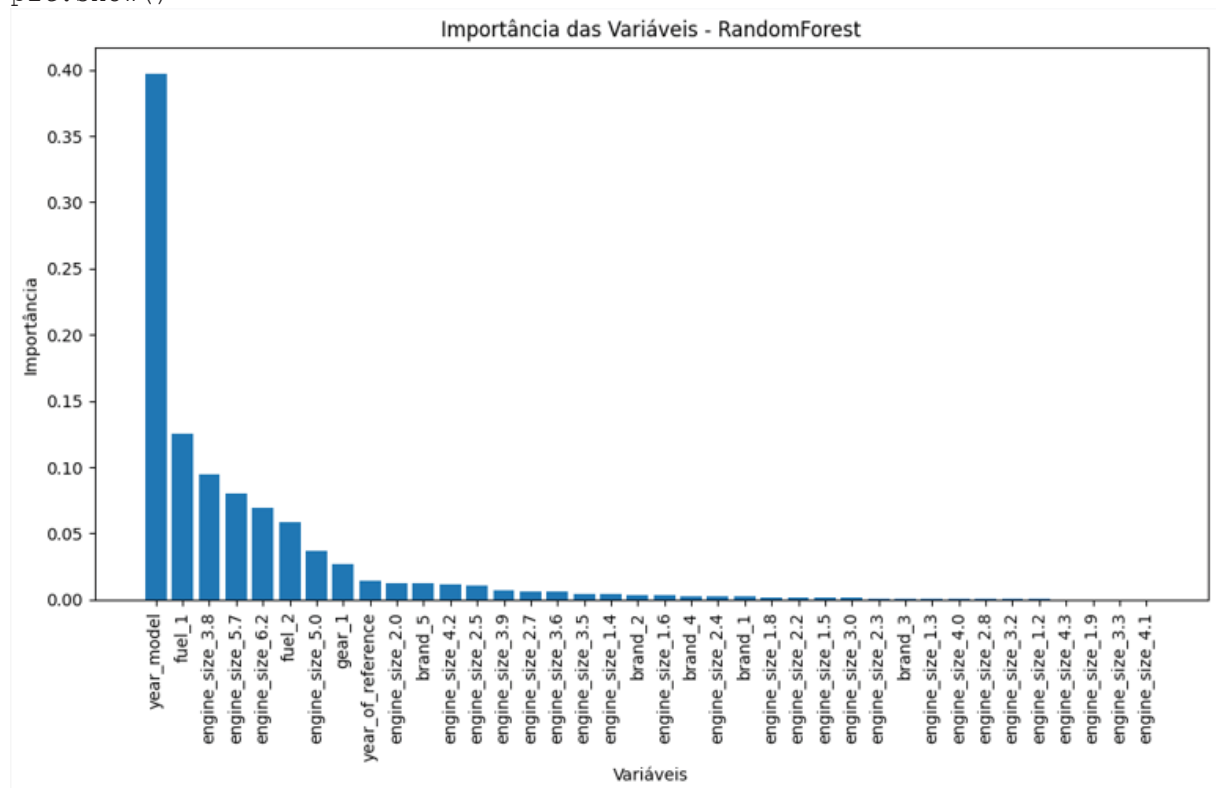
# Cross-validation
scores = cross_val_score(xgb_model_cross, X_train, y_train, cv=5,
scoring='neg_mean_squared_error')
print("MSE scores:", scores)
print("MSE mean:", scores.mean())
MSE scores: [-1.28387897e+08 -1.17135167e+08 -1.09676920e+08 -1.12798240e+08
-1.08947740e+08]
MSE mean: -115389192.90856056
```

e)

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

importances_rf = rf_model.feature_importances_
indices_rf = np.argsort(importances_rf)[::-1]

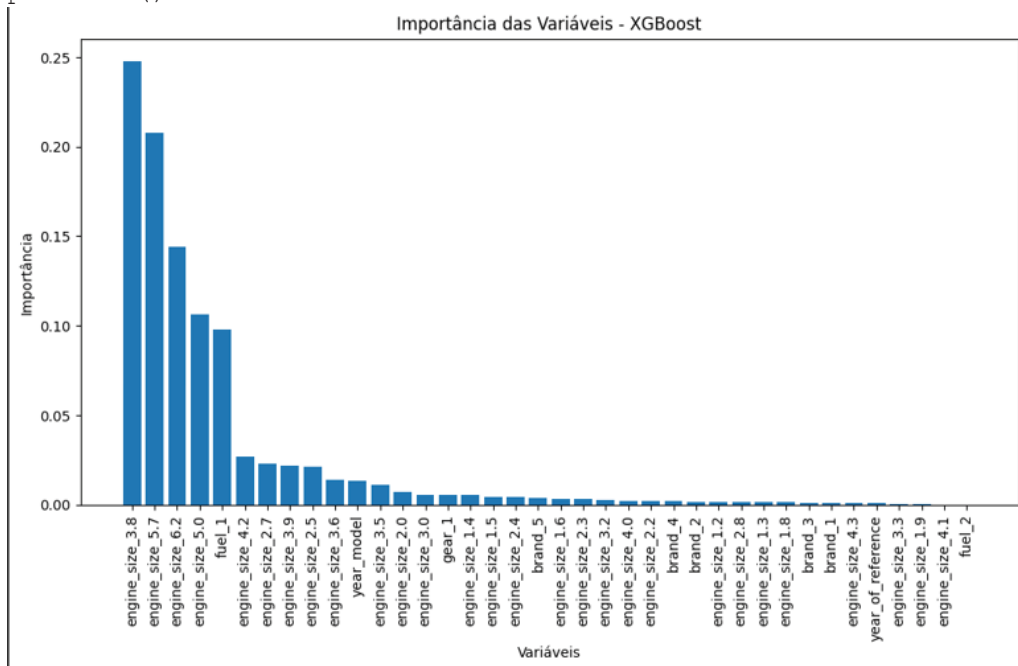
# Visualizando a importância das variáveis
plt.figure(figsize=(12,6))
plt.bar(range(X_train.shape[1]), importances_rf[indices_rf])
plt.xticks(range(X_train.shape[1]), X_train.columns[indices_rf],
rotation=90)
plt.xlabel("Variáveis")
plt.ylabel("Importância")
plt.title("Importância das Variáveis - RandomForest")
plt.show()
```



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

importances_rf = xgb_model.feature_importances_
indices_rf = np.argsort(importances_rf)[::-1]
```

```
# Visualizando a importância das variáveis
plt.figure(figsize=(12,6))
plt.bar(range(X_train.shape[1]), importances_rf[indices_rf])
plt.xticks(range(X_train.shape[1]), X_train.columns[indices_rf],
rotation=90)
plt.xlabel("Variáveis")
plt.ylabel("Importância")
plt.title("Importância das Variáveis - XGBoost")
plt.show()
```



f) Tamanho do motor, ano do modelo e tipo de combustível são os principais fatores que determinam o preço do carro. Outras variáveis também são importantes, mas em menor grau. Mês do ano tem menor impacto no preço.

g) Random Forest teve melhor desempenho que XGBoost em todas as métricas: Menor MSE (106.83M vs. 111.06M), Menor MAE (5.59K vs. 5.86K) e Maior R^2 (0.960 vs. 0.958). Conclusão, o modelo Random Forest é a melhor escolha para prever o preço do carro neste caso.

h) O modelo Random Forest apresentou o melhor resultado em termos de MSE, MAE e R^2 . Isso indica que o modelo é mais preciso na previsão do preço do carro do que o XGBoost neste caso.

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

- Coeficiente de determinação: R^2

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- $S_{yx}\%$

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1 Pesquisa com Dados de Satélite (Satellite)

```
# instalar o pacote mlbench
install.packages("mlbench")
# iniciar o pacote mlbench
library("mlbench")
# carregar a base Satellite (O banco de dados se encontra no pacote mlbench
e é completo (não possui dados faltantes))
data(Satellite)
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```


Random Forest:

Confusion Matrix and Statistics					
Prediction	Reference				
	red soil	cotton crop	grey soil	damp grey soil	grey soil
red soil	296	1	5		2
cotton crop	0	123	0		0
grey soil	3	0	238		29
damp grey soil	1	0	21		62
vegetation stubble	6	10	0		0
very damp grey soil	0	6	7		32

Prediction	Reference			
	vegetation stubble	very damp grey soil	grey soil	
red soil	8		0	
cotton crop	2		1	
grey soil	1		10	
damp grey soil	2		40	
vegetation stubble	120		8	
very damp grey soil	8		242	

Overall Statistics

Accuracy : 0.8419
 95% CI : (0.8208, 0.8614)
 No Information Rate : 0.2383
 P-Value [Acc > NIR] : < 2.2e-16
 ...
 Neg Pred Value : 0.9403
 Prevalence : 0.2344
 Detection Rate : 0.1885
 Detection Prevalence : 0.2298
 Balanced Accuracy : 0.8750

SVM:

Confusion Matrix and Statistics					
Prediction	Reference				
	red soil	cotton crop	grey soil	damp grey soil	grey soil
red soil	298	1	4		2
cotton crop	1	120	0		0
grey soil	4	0	260		29
damp grey soil	0	1	7		69
vegetation stubble	3	14	0		2
very damp grey soil	0	4	0		23

Prediction	Reference			
	vegetation stubble	very damp grey soil	grey soil	
red soil	7		0	
cotton crop	4		0	
grey soil	1		12	
damp grey soil	2		32	
vegetation stubble	117		3	
very damp grey soil	10		254	

Overall Statistics

Accuracy : 0.8707
 95% CI : (0.8511, 0.8886)
 No Information Rate : 0.2383
 P-Value [Acc > NIR] : < 2.2e-16
 ...
 Neg Pred Value : 0.9527
 Prevalence : 0.2344
 Detection Rate : 0.1978
 Detection Prevalence : 0.2266
 Balanced Accuracy : 0.9031

RNA:

Confusion Matrix and Statistics					
Prediction	Reference				
	red soil	cotton crop	grey soil	damp grey soil	grey soil
red soil	290	1	8		2
cotton crop	5	122	0		0
grey soil	3	0	262		79
damp grey soil	0	0	0		1
vegetation stubble	7	9	0		0
very damp grey soil	1	8	1		43

Prediction	Reference			
	vegetation stubble	very damp grey soil	grey soil	
red soil	10		0	
cotton crop	10		0	
grey soil	3		41	
damp grey soil	2		2	
vegetation stubble	100		2	
very damp grey soil	16		256	

Overall Statistics	
Accuracy	: 0.803
95% CI	: (0.7801, 0.8244)
No Information Rate	: 0.2383
P-Value [Acc > NIR]	: < 2.2e-16
...	
Neg Pred Value	0.9531
Prevalence	0.2344
Detection Rate	0.1994
Detection Prevalence	0.2531
Balanced Accuracy	0.8902

RF Accuracy : 0.8419

SVM Accuracy : 0.8707

RNA Accuracy : 0.803

O melhor modelo foi o SVM com 87% de acurácia.

2 Estimativa de Volumes de Árvores

```
# Carga da base de dados separado por ; e decimal por ,
df <- read.csv("http://www.razer.net.br/datasets/Volumes.csv", sep=";",
dec=",")
summary(df)
```

```

      NR      DAP      HT      HP
Min.   : 1.00   Min.  :27.50   Min.   :22.80   Min.   :1.000
1st Qu.: 25.75   1st Qu.:32.50   1st Qu.:25.50   1st Qu.:1.700
Median : 50.50   Median :35.25   Median :26.40   Median :2.225
Mean   : 50.50   Mean   :35.86   Mean   :26.24   Mean   :2.135
3rd Qu.: 75.25   3rd Qu.:39.05   3rd Qu.:27.15   3rd Qu.:2.562
Max.   :100.00   Max.   :49.00   Max.   :28.40   Max.   :3.400

      VOL
Min.   :0.6982
1st Qu.:1.0553
Median :1.3006
Mean   :1.3583
3rd Qu.:1.6191
Max.   :2.5245

```

```
df$NR <- NULL
```

```
head(df)
```

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

```

rf <- caret::train(VOL~., data=treino,method="rf")
svm <- caret::train(VOL~., data=treino, method="svmRadial")
rna <- caret::train(VOL~., data=treino, method="nnet")
alom <- nls(VOL~b0 + b1*DAP*DAP*HT, df, start=list(b0=0.5, b1=0.5))
predicoes.rf <- predict(rf, teste)
predicoes.svm <- predict(svm, teste)
predicoes.rna <- predict(rna, teste)
predicoes.alom <- predict(alom, teste)
y_real = c(df$VOL) # volume real
y_pred = c(predicoes.rf) # volume predito
n = length(y_real) # tamanho do vetor
n
length(y_real)
length(y_pred)
r2_rf <- ( 1 - (sum((y_real - y_pred) ^ 2)) / (sum((y_real - mean(y_real)) ^
2) ) )
syx_rf <- sqrt( sum( (y_real - y_pred) ^ 2 ) / ( n - 2 ) )
r2_svm <- ( 1 - (sum((y_real - y_pred) ^ 2)) / (sum((y_real - mean(y_real))
^ 2) ) )
syx_svm <- sqrt( sum( (y_real - y_pred) ^ 2 ) / ( n - 2 ) )

```

```

r2_rna <- ( 1 - (sum((y_real - y_pred) ^ 2)) / (sum((y_real - mean(y_real))
^ 2) ) )
syx_rna <- sqrt( sum( (y_real - y_pred) ^ 2 ) / ( n - 2 ) )
r2_spurr <- ( 1 - (sum((y_real - y_pred) ^ 2)) / (sum((y_real - mean(y_real))
^ 2) ) )
syx_spurr <- sqrt( sum( (y_real - y_pred) ^ 2 ) / ( n - 2 ) )

```

R²: Quanto mais perto de 1 melhor é o modelo!

Coeficiente de determinação R² para o Modelo RF: -0.6655005
 Coeficiente de determinação R² para o Modelo SVM: -0.5425426
 Coeficiente de determinação R² para o Modelo RNA: -0.77481
 Coeficiente de determinação R² para o Modelo SPURR: -0.7265465

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

Erro padrão da estimativa Syx para o Modelo RF: 0.530585
 Erro padrão da estimativa Syx para o Modelo SVM: 0.510624
 Erro padrão da estimativa Syx para o Modelo RNA: 0.5477199
 Erro padrão da estimativa Syx para o Modelo SPURR: 0.5402213

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

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

B – RESOLUÇÃO

1. Gráficos e Tabelas

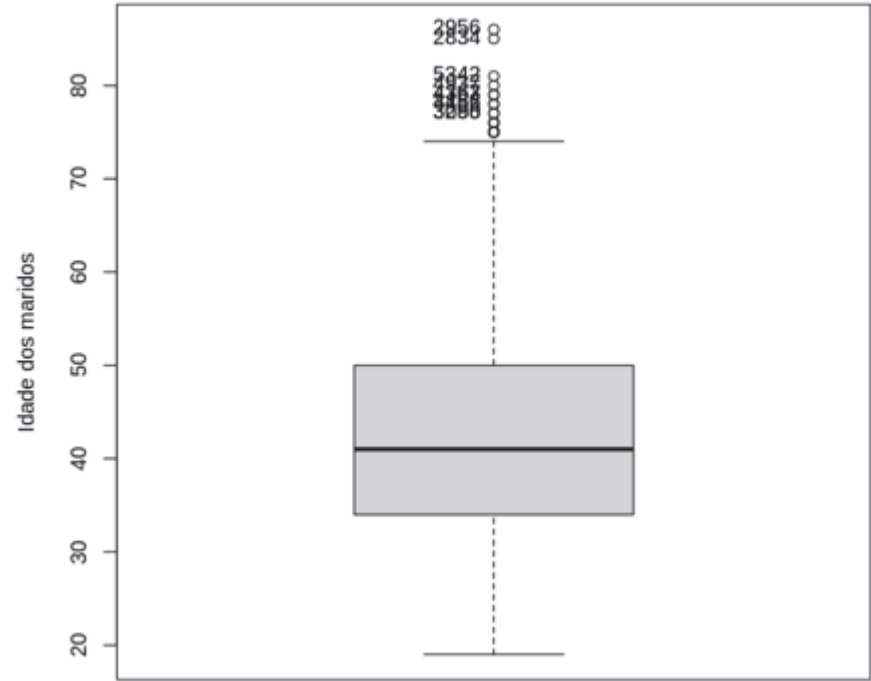
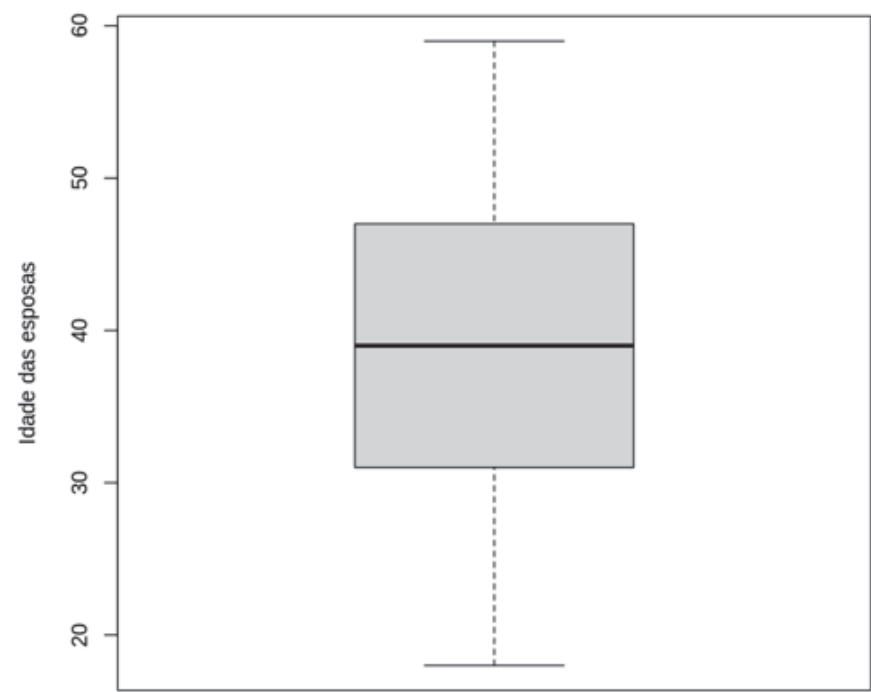
```
# Carregando a base de dados "salarios" em formato R
load("salarios.RData")
# head(salarios)
# visualizando algumas estatisticas e informacoes das variaveis
# summary(salarios)
# str(salarios)
# length(salarios)
dim(salarios)
# Boxplot - idade das esposas
Boxplot( ~ age, data=salarios, id=list(method="y"),
ylab="Idade das esposas")
# Boxplot - idade dos maridos
Boxplot( ~ husage, data=salarios, id=list(method="y"),
ylab="Idade dos maridos")
# verificar os id's outliers para husage (idade do marido)
outliers = c(2956,2834,5342,4937,4363,4367,3456,4182,3090,3288)
outliers = salarios[outliers, ]
# outliers para husage
sort(outliers$husage)
# comparar o boxplot de idade dos maridos e esposas
age_hw <- data.frame(salarios$husage, salarios$age) # data frame com idade
dos
maridos e esposas
Boxplot(
  age_hw,
  id=list(method="y"),
  main="Diferença de Idade dos Maridos e Esposas",
  ylab="Idade dos Maridos e Esposas"
)
# Histograma - idade das esposas
hist(
  x=salarios$age,
  main="Histograma de Idade das Esposas",
  xlab="Idade das Esposas",
  ylab = "Frequência"
)
# Histograma - idade dos maridos
```

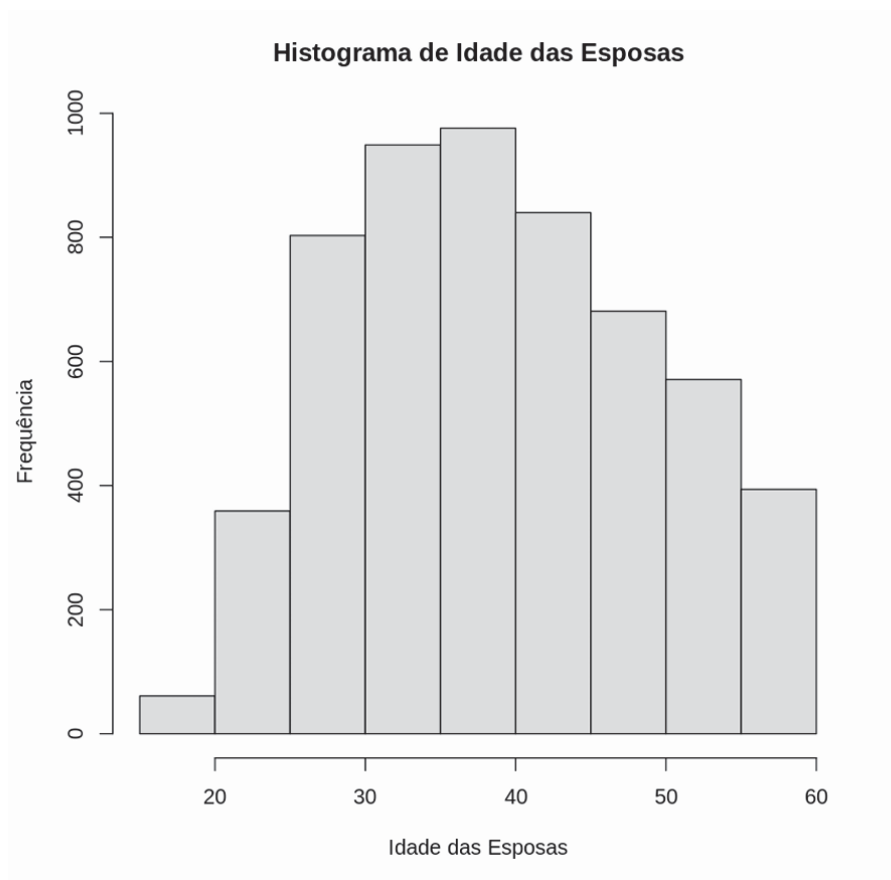
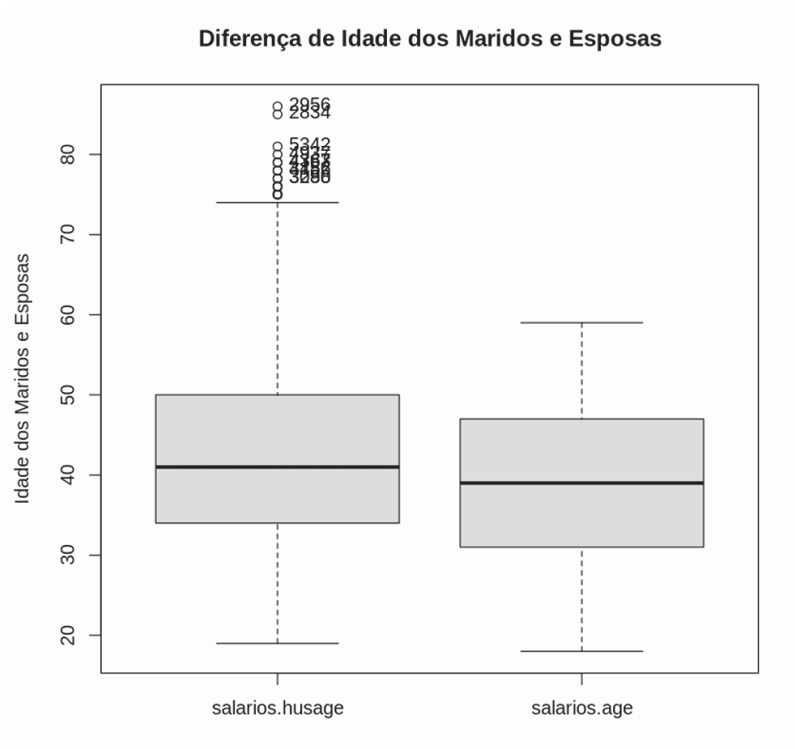
```

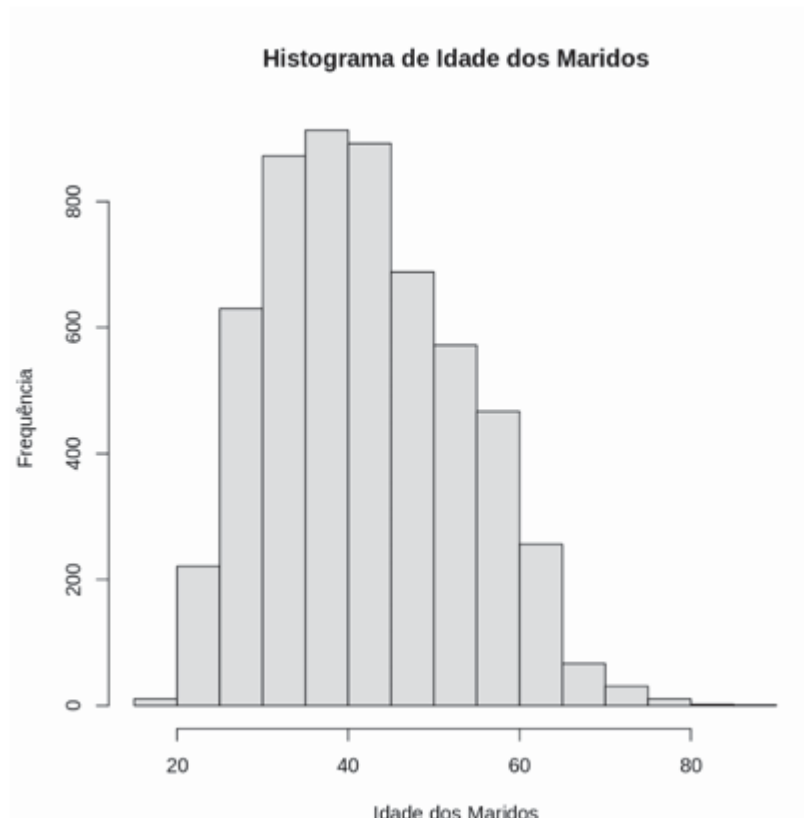
hist(
x=salarios$husage,
main="Histograma de Idade dos Maridos",
xlab="Idade dos Maridos",
ylab = "Frequência"
)
cat("RESULTADOS:")
cat("\nIdade das Esposas variam
entre",min(salarios$age),"e",max(salarios$age),"anos com média
de",mean(salarios$age),"anos e mediana de",median(salarios$age),"anos.")
cat("\nIdade dos Maridos variam
entre",min(salarios$husage),"e",max(salarios$husage),"anos com média
de",mean(salarios$husage),"anos e mediana
de",median(salarios$husage),"anos.")
cat("\nIdentificado",length(outliers$husage),"valores outliers para a idade
dos
Maridos variando
entre",min(outliers$husage),"e",max(outliers$husage),"anos.")
# Distribuição de frequência - idade das esposas
fdt_w <- fdt(salarios$age)
print(fdt_w)
# Com o pacote descriptr
ds_freq_table(salarios, age, 14)
# Distribuição de frequência - idade dos maridos
fdt_h <- fdt(salarios$husage)
print(fdt_h)
# Com o pacote descriptr
ds_freq_table(salarios, husage, 14)
cat("RESULTADOS:")
cat("\nIdentificado 14 classes (age e husage) que são utilizados pelo
histograma.")
cat("\nA idade mais frequente para as Esposas é entre 35.721 e 38.705 anos
com uma
frequência de 624 ou 11.08%.")
cat("\nA idade menos frequente para as Esposas é entre 17.82 e 20.804 anos
com uma
frequência de 61 ou 1.08%.")
cat("\nA idade mais frequente para os Maridos é entre 38.253 e 43.114 anos
com uma
frequência de 917 ou 16.28%.")

```

cat("\nA idade menos frequente para os Maridos é entre 81.999 e 86.86 anos
com uma
frequência de 2 ou 0.04%.")







Resultados: Idade das Esposas variam entre 18 e 59 anos com média de 39.42758 anos e mediana de 39 anos. Idade dos Maridos variam entre 19 e 86 anos com média de 42.45296 anos e mediana de 41 anos. Identificado 10 valores outliers para a idade dos Maridos variando entre 77 e 86 anos.

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

Variable: age

Bins			Frequency	Cum Frequency	Percent	Cum Percent
18	-	20.9	61	61	1.08	1.08
20.9	-	23.9	161	222	2.86	3.94
23.9	-	26.8	312	534	5.54	9.48
26.8	-	29.7	505	1039	8.96	18.44
29.7	-	32.6	562	1601	9.98	28.42
32.6	-	35.6	571	2172	10.13	38.55
35.6	-	38.5	624	2796	11.08	49.63
38.5	-	41.4	510	3306	9.05	58.68
41.4	-	44.4	542	3848	9.62	68.3
44.4	-	47.3	432	4280	7.67	75.97
47.3	-	50.2	389	4669	6.9	82.87
50.2	-	53.1	358	5027	6.35	89.23
53.1	-	56.1	304	5331	5.4	94.62
56.1	-	59	303	5634	5.38	100
Total			5634	-	100.00	-

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

Variable: husage

Bins			Frequency	Cum Frequency	Percent	Cum Percent
19	-	23.8	102	102	1.81	1.81
23.8	-	28.6	466	568	8.27	10.08
28.6	-	33.4	809	1377	14.36	24.44
33.4	-	38.1	895	2272	15.89	40.33
38.1	-	42.9	739	3011	13.12	53.44
42.9	-	47.7	807	3818	14.32	67.77
47.7	-	52.5	649	4467	11.52	79.29
52.5	-	57.3	541	5008	9.6	88.89
57.3	-	62.1	394	5402	6.99	95.88
62.1	-	66.9	138	5540	2.45	98.33
66.9	-	71.6	56	5596	0.99	99.33
71.6	-	76.4	28	5624	0.5	99.82
76.4	-	81.2	8	5632	0.14	99.96
81.2	-	86	2	5634	0.04	100
Total			5634	-	100.00	-

Resultados: Identificado 14 classes (age e husage) que são utilizados pelo histograma. A faixa etária predominante das esposas é de 35.721 a 38.705 anos, respondendo por 624 casos (11,08%). O intervalo de idade menos recorrente entre as esposas está compreendido entre 17,82 e 20,804 anos, com uma frequência registrada de 61 casos, equivalente a 1,08%. A faixa etária mais comum dos maridos é de 38,253 a 43,114 anos, com 917 casos (16,28%). A faixa etária mais rara entre os Maridos é de 81.999 a 86.86 anos, ocorrendo apenas 2 vezes (0,04%).

2. Medidas de posição e dispersão

```
# média de idade das esposas
mean_w <- mean(salarios$age)
# média de idade dos maridos
mean_h <- mean(salarios$husage)

# mediana de idade das esposas
median_w <- median(salarios$age)
# mediana de idade dos maridos
```

```

median_h <- median(salarios$husage)

# moda das idades das esposas
mode_w <- mfv(salarios$age)
subset(table(salarios$age), table(salarios$age) == max(table(salarios$age)))
# moda das idades dos maridos
mode_h <- mfv(salarios$husage)
subset(table(salarios$husage), table(salarios$husage) ==
max(table(salarios$husage)))

# diferença entre idade dos Maridos e Esposas
mean_diff <- (((mean_h/mean_w)-1)*100)
median_diff <- (((median_h/median_w)-1)*100)
mode_diff <- (((mode_h/mode_w)-1)*100)

cat("MÉDIA:\n")
cat("Média de Idade das Esposas:",mean_w)
cat("\nMédia de Idade dos Maridos:",mean_h)
cat("\nDiferença de",mean_diff,"% entre a média de idades dos Maridos e
Esposas.")
cat("\n\nMEDIANA:")
cat("\nMediana de Idade das Esposas:",median_w)
cat("\nMediana de Idade dos Maridos:",median_h)
cat("\nDiferença de",median_diff,"% entre a mediana de idades dos Maridos e
Esposas.")

cat("\n\nMODA:")
cat("\nModa das Idades das Esposas:",mode_w,"(217 repetições).")
cat("\nModa das Idades dos Maridos:",mode_h,"(201 repetições).")
# Calculando a variancia das idades das esposas
var_w <- var(salarios$age)
# Calculando a variancia das idades dos maridos
var_h <- var(salarios$husage)
# Diferença da variância das idades das esposas e maridos
var_diff <- (((var_h/var_w)-1)*100)

cat("VARIÂNCIA:\n")
cat("Variância das Idades das Esposas:",var_w)
cat("\nVariância das Idades dos Maridos:",var_h)
cat("\nDiferença de",var_diff,"% entre a média de idades dos Maridos e
Esposas.")

```

```

# Testar se as médias (se você escolher o teste paramétrico) - Unpaired Test

# notação científica
options(scipen = 999)

### Média ###

# média de idade das esposas
mean_w <- mean(salarios$age)
cat("média de idade das esposas:",mean_w)
# média de idade dos maridos
mean_h <- mean(salarios$husage)
cat("\nmédia de idade dos maridos:",mean_h)

### Desvio Padrão ###

# desvio padrão da idade das esposas
sd_w <- sd(salarios$age)
cat("\ndesvio padrão da idade das esposas",sd_w)
# desvio padrão da idade dos maridos
sd_h <- sd(salarios$husage)
cat("\ndesvio padrão da idade dos maridos",sd_h)

# - H0: os dados sao normalmente distribuidos
# - Ha: os dados nao sao normalmente distribuidos

# Teste de normalidade Kolmogorov-Smirnov
lillie.test(salarios$age) # p-value = 0.000000000000000022
lillie.test(salarios$husage) # p-value = 0.000000000000000022
# Teste de Jarque Berra
JarqueBeraTest(salarios$age, robust = TRUE)
JarqueBeraTest(salarios$husage, robust = TRUE)
# p-value < 0.05 então não é normalmente distribuida

cat("\nComo a distribuição não é normalmente distribuida, não utilizamos o
teste Z
e t")
# Testar se as medianas (se você escolher o teste não paramétrico) - Mann-
Whitney U
Test

```

```

options(scipen = 999)

w_age <- sort(c(salarios$age)) # idade das esposas
h_age <- sort(c(salarios$husage)) # idade dos maridos
size <- nrow(salarios) # tamanho em linhas

age <- data.frame(
  group = rep(c("Wife", "Husband"), each = size),
  age = c(w_age, h_age)
)

# Queremos saber se a idade mediano das mulheres difere da idade mediano dos
homens.

# Vamos primeiro calcular um sumario estatistico
group_by(age, group) %>%
  summarise(
    count = n(),
    median = median(age, na.rm = TRUE),
    IQR = IQR(age, na.rm = TRUE)
  )

# Vamos visualizar os dados usando box-plots
# Plotaremos o "age" por grupo
ggboxplot(age, x = "group", y = "age",
           color = "group", palette=c("red", "blue"), ylab = "Age", xlab =
"Groups")

# Hipoteses do teste:

# H0: A idade mediano das esposas é igual estatisticamente a idade mediano
dos
maridos;
# Ha: A idade mediano das esposas não é igual estatisticamente a idade mediano
dos
maridos;

# Wilcox Test
wilcox.test(age ~ group, data = age,
             exact = FALSE, conf.int=TRUE)

```

```
cat("\nO p-value do teste eh 0,000000000000000022, menor que o nivel de
significancia 0,05. Portanto rejeitamos H0.")
```

```
# Usamos a seguinte funcao:
wilcox.test(age ~ group, data = age,
            exact = FALSE, alternative = "less",
            conf.int=TRUE)
```

Média:

- Média de Idade das Esposas: 39.42758
- Média de Idade dos Maridos: 42.45296
- Diferença de 7.673262 % entre a média de idades dos Maridos e Esposas.

Mediana:

- Mediana de Idade das Esposas: 39
- Mediana de Idade dos Maridos: 41
- Diferença de 5.128205 % entre a mediana de idades dos Maridos e Esposas.

Moda:

- Moda das Idades das Esposas: 37 (217 repetições).
- Moda das Idades dos Maridos: 44 (201 repetições).

Variância:

- Variância das Idades das Esposas: 99.75234
- Variância das Idades dos Maridos: 126.0717
- Diferença de 26.38473 % entre a média de idades dos Maridos e Esposas.

3. Testes paramétricos ou não paramétricos

Média e Desvio Padrão das Idades:

- Média de idade das esposas: 39.42758
- Média de idade dos maridos: 42.45296
- Desvio padrão da idade das esposas: 9.98761
- Desvio padrão da idade dos maridos: 11.22817

Teste de Normalidade Lilliefors (Kolmogorov-Smirnov):

- Dados: salários\$age
 - D: 0.058909
 - p-valor: < 0.000000000000000022
- Dados: salários\$usage
 - D: 0.059662
 - p-valor: < 0.000000000000000022

Teste de Robustez Jarque Bera:

- Dados: salários\$age
 - X-quadrado: 158.49
 - df: 2
 - p-valor: < 0.00000000000000022
- Dados: salários\$husage
 - X-quadrado: 153.12
 - df: 2
 - p-valor: < 0.00000000000000022

Como a distribuição não é normal, não utilizamos os testes Z e t.

A tibble: 2 × 4

group	count	median	IQR
<chr>	<int>	<dbl>	<dbl>
Husband	5634	41	16
Wife	5634	39	16

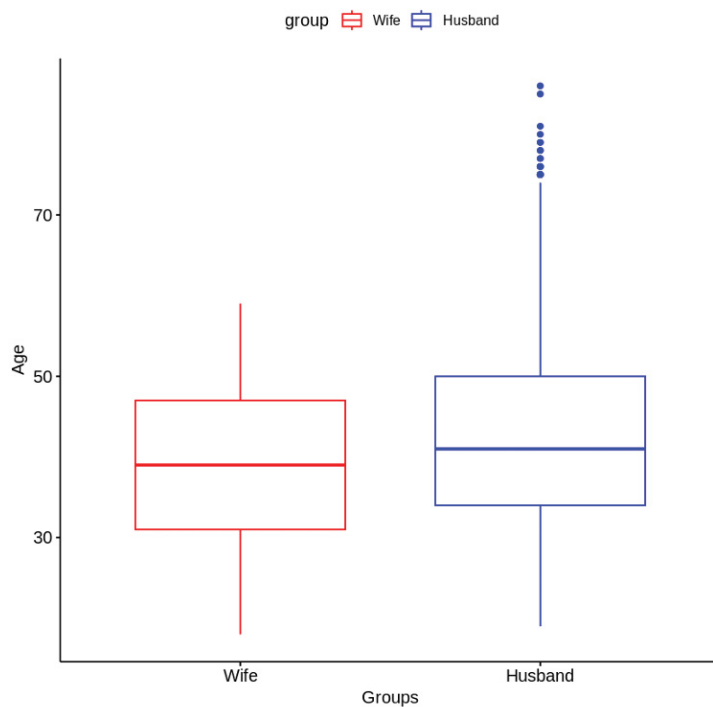
Teste de Wilcoxon com correção de continuidade:

- Dados: Idade por grupo
- Estatística W: 18122044
- p-valor: < 0.00000000000000022
- Hipótese alternativa: O deslocamento verdadeiro da localização não é igual a 0
- Intervalo de confiança de 95%: 2.000033 a 3.000024
- Estimativa da diferença na localização: 2.999966

O p-valor do teste é 0.00000000000000022, menor que o nível de significância de 0.05. Portanto, rejeitamos a hipótese nula (H_0).

Teste de Wilcoxon com correção de continuidade:

- Dados: Idade por grupo
- Estatística W: 18122044
- p-valor: 1
- Hipótese alternativa: O deslocamento verdadeiro da localização é menor que 0
- Intervalo de confiança de 95%: -Inf a 3.000034
- Estimativa da diferença na localização: 2.999966



Baseado nos resultados dos testes:

1. Violação da normalidade:

- Testes de normalidade:
 - Teste de Kolmogorov-Smirnov: p-valor = 0.00000000000000022 para ambas as variáveis (idade das esposas e idade dos maridos).
 - Teste de Jarque-Bera: p-valor < 0.05 para ambas as variáveis.
- Conclusão: A distribuição das idades das esposas e dos maridos não é normal.

2. Robustez:

- Testes paramétricos: Baseiam-se em suposições sobre a distribuição dos dados, como a normalidade.
- Testes não paramétricos: Não fazem suposições sobre a distribuição dos dados, sendo mais robustos a violações da normalidade.

Considerando os pontos acima:

- Teste Z e t: Inapropriados devido à violação da normalidade.
- Teste U de Mann-Whitney: Adequado para comparar as medianas das idades (não faz suposições sobre a normalidade).

Resultados do teste U de Mann-Whitney:

- p-valor: 0.00000000000000022 (menor que o nível de significância 0.05).

Conclusão: Rejeitamos a hipótese nula (H_0) de que as medianas das idades das esposas e dos maridos são iguais.

APÊNDICE E - ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

```
# glimpse
install.packages("dplyr")
library(dplyr)

# dummyVars
install.packages("caret")
library(caret)

# cv.glmnet
install.packages("glmnet")
library(glmnet)

# Carregando a base de dados "trabalhosalarios" em formato R
load("trabalhosalarios.RData")
# summary(trabalhosalarios)
# dim(trabalhosalarios)

# verificar dataset
glimpse(trabalhosalarios)
# guardar o dataset na variável dat com as variáveis do trabalho
dat <- select(trabalhosalarios, c(
  'husage',
  'husunion',
  'husearns',
  'huseduc',
  'husblack',
  'hushisp',
  'hushrs',
  'kidge6',
  'age',
  'black',
  'educ',
  'hispanic',
  'union',
  'exper',
  'kidlt6',
  'lwage'
```

```

))

dim(dat)
# semente para resultados iguais
set.seed(302)

# criar indice para particionar o dataset em 80% para treinamento

index = sample(1:nrow(dat),0.8*nrow(dat))

# criar base de treinamento 80%

train = dat[index,]
cat("treino: ",dim(train))

# criar base de teste 20%

test = dat[-index,]
cat("\nteste: ",dim(test))
cols = c(
  'husage',
  'husearns',
  'huseduc',
  'hushrs',
  'age',
  'educ',
  'exper',
  'lwage'
)

Cols
pre_proc_val <- preProcess(train[,cols],
                           method = c("center", "scale"))
train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])
cols_reg = c(
  'husage',
  'husunion',
  'husearns',
  'huseduc',
  'husblk',

```

```

'hushisp',
'hushrs',
'kidge6',
'age',
'black',
'educ',
'hispanic',
'union',
'exper',
'kidlt6',
'lwage'
)
dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+
                      husblk+hushisp+hushrs+kidge6+age+
                      black+educ+hispanic+union+exper+kidlt6,
                      data = dat[,cols_reg])
train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])
print(dim(train_dummies)); print(dim(test_dummies))

# RIDGE
# Vamos guardar a matriz de dados de treinamento das
# variaveis explicativas para o modelo em um objeto
# chamado "x"
x = as.matrix(train_dummies)

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

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

# Vamos guardar o vetor de dados de teste da variavel
# dependente para o modelo em um objeto chamado "y_test"
y_test = test$lwage

# Vamos testar os lambdas de 10^-3 ate 10^2, a cada 0.1

```

```

lambdas <- 10^seq(2, -3, by = -.1)
# Calculando o lambda:
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0,
                        lambda = lambdas)
# Vamos ver qual o lambda otimo
best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge

# RIDGE
# Para contar o tempo de processamento do modelo usamos:
start <- Sys.time()
# Estimando o modelo Ridge
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
                  family = 'gaussian',
                  lambda = best_lambda_ridge)
end <- Sys.time()
difftime(end, start, units="secs")

# Vamos ver o resultado (valores) da estimativa
# (coeficientes)
ridge_reg[["beta"]]

# RIDGE
# Vamos calcular o R^2 dos valores verdadeiros e
# preditos conforme a seguinte funcao:
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

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

# RIDGE
# Predicao e avaliacao nos dados de treinamento:
predictions_train <- predict(ridge_reg,

```

```

        s = best_lambda_ridge,
        newx = x)

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

# 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)

# RIDGE

# Como os valores do dataset sao padronizados, nos temos
# de padronizar tambem os dados que vamos fazer a predicao
# Note que as variaveis dummies nao devem ser padronizadas
# valores para predicao:
husage = (40
pre_proc_val[["mean"]][["husage"]])/pre_proc_val[["std"]][["husage"]]
husunion=0
husearns = (600
pre_proc_val[["mean"]][["husearns"]])/pre_proc_val[["std"]][["husearns"]]
huseduc = (13
pre_proc_val[["mean"]][["huseduc"]])/pre_proc_val[["std"]][["huseduc"]]
]
husblk=1
hushisp=0
hushrs = (40
pre_proc_val[["mean"]][["hushrs"]])/pre_proc_val[["std"]][["hushrs"]]
kidge6=1
age = (38
pre_proc_val[["mean"]][["age"]])/pre_proc_val[["std"]][["age"]]
black=0
educ = (13
pre_proc_val[["mean"]][["educ"]])/pre_proc_val[["std"]][["educ"]]
hispanic=1
union=0

```



```

exper= (18
pre_proc_val[["mean"]][["exper"]])/pre_proc_val[["std"]][["exper"]]
kidlt6=1

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

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

# ridge_reg
# best_lambda_ridge
# our_pred
# O resultado da predicao eh:
predict_our_ridge

# O resultado eh um valor padronizado, vamos converte-lo
# para o valor nominal, consistente com o dataset original
wage_pred_ridge=(predict_our_ridge*
                  pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]

# O resultado eh:

```

```

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

# Os valores sao:
CIlwr_ridge
CIupr_ridge
# Vamos criar o intervalo para o tuning do melhor lambda
lambdas <- 10^seq(2, -3, by = -.1)
# Vamos atribuir alpha = 1 para implementar a regressao
# lasso
lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
lambda = lambdas,
standardize = TRUE, nfolds = 5)
# Vamos guardar o lambda "otimo" em um objeto chamado
# best_lambda_lasso
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso
# Vamos estimar o modelo Lasso
lasso_model <- glmnet(x, y_train, alpha = 1,
lambda = best_lambda_lasso,
standardize = TRUE)
# Vamos visualizar os coeficientes estimados
lasso_model[["beta"]]
# Vamos fazer as predicoes na base de treinamento e
# avaliar a regressao Lasso
predictions_train <- predict(lasso_model,
s = best_lambda_lasso,
newx = x)
# Vamos calcular o R^2 dos valores verdadeiros e
# preditos conforme a seguinte funcao:
eval_results <- function(true, predicted, df) {
SSE <- sum((predicted - true)^2)
SST <- sum((true - mean(true))^2)
R_square <- 1 - SSE / SST

```

```

RMSE = sqrt(SSE/nrow(df))
# As metricas de performace do modelo:
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)
}
# As metricas da base de treinamento sao:
eval_results(y_train, predictions_train, train)
# Vamos fazer as predicoes na base de teste
predictions_test <- predict(lasso_model,
                             s = best_lambda_lasso,
                             newx = x_test)

# As metricas da base de teste sao:
eval_results(y_test, predictions_test, test)
# Perceba que os erros nao sao muito diferentes, assim
# como os valores de R2.
# LASSO

# Como os valores do dataset sao padronizados, nos temos
# de padronizar tambem os dados que vamos fazer a predicao
# Note que as variaveis dummies nao devem ser padronizadas
# valores para predicao:
husage = (40
pre_proc_val[["mean"]][["husage"]])/pre_proc_val[["std"]][["husage"]]
husunion=0
husearns = (600
pre_proc_val[["mean"]][["husearns"]])/pre_proc_val[["std"]][["husearns"]]
huseduc = (13
pre_proc_val[["mean"]][["huseduc"]])/pre_proc_val[["std"]][["huseduc"]]
]
husblck=1
hushisp=0
hushrs = (40
pre_proc_val[["mean"]][["hushrs"]])/pre_proc_val[["std"]][["hushrs"]]
kidge6=1
age = (38
pre_proc_val[["mean"]][["age"]])/pre_proc_val[["std"]][["age"]]
black=0

```

```

educ = (13
pre_proc_val[["mean"]][["educ"]])/pre_proc_val[["std"]][["educ"]]
hispanic=1
union=0
exper= (18
pre_proc_val[["mean"]][["exper"]])/pre_proc_val[["std"]][["exper"]]
kidlt6=1

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

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

predict_our_lasso

# Novamente, o resultado esta padronizado, nos temos de
# converte-lo para valor compativel com o dataset original
wage_pred_lasso=(predict_our_lasso*
                  pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]
wage_pred_lasso

# Vamos criar o intervalo de confianca para o nosso

```

```

# exemplo
n <- nrow(train)
m <- wage_pred_lasso
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
Cilwr_lasso <- m + (qnorm(0.025))*dam
CIupr_lasso <- m - (qnorm(0.025))*dam

# O intervalo de confianca eh:
Cilwr_lasso
CIupr_lasso

# Vamos configurar o treinamento do modelo por
# cross validation, com 10 folders, 5 repeticoes
# e busca aleatoria dos componentes das amostras
# de treinamento, o "verboseIter" eh soh para
# mostrar o processamento.
train_cont <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 5,
                           search = "random",
                           verboseIter = TRUE)

# O parametro "lambda" tambem eh escolhido por
# cross-validation

# Vamos treinar o modelo
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)

# O melhor parametro alpha escolhido eh:
elastic_reg$bestTune

# E os parametros sao:
elastic_reg[["finalModel"]][["beta"]]

# Vamos fazer as predicoes e avaliar a performance do
# modelo

# Vamos fazer as predicoes no modelo de treinamento:
predictions_train <- predict(elastic_reg, x)

# Vamos calcular o R^2 dos valores verdadeiros e
# preditos conforme a seguinte funcao:
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

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

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

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

# As metricas de performance na base de teste sao:

```

```

eval_results(y_test, predictions_test, test)

# Vamos agora fazer a predicao para o nosso exemplo,
# mesmos parametros utilizados para a regressao

husage = (40
pre_proc_val[["mean"]][["husage"]])/pre_proc_val[["std"]][["husage"]]
husunion=0
husearns = (600
pre_proc_val[["mean"]][["husearns"]])/pre_proc_val[["std"]][["husearn
s"]]
huseduc = (13
pre_proc_val[["mean"]][["huseduc"]])/pre_proc_val[["std"]][["huseduc"
]]
husblck=1
hushisp=0
hushrs = (40
pre_proc_val[["mean"]][["hushrs"]])/pre_proc_val[["std"]][["hushrs"]]
kidge6=1

age = (38
pre_proc_val[["mean"]][["age"]])/pre_proc_val[["std"]][["age"]]
black=0
educ = (13
pre_proc_val[["mean"]][["educ"]])/pre_proc_val[["std"]][["educ"]]
hispanic=1
union=0
exper= (18
pre_proc_val[["mean"]][["exper"]])/pre_proc_val[["std"]][["exper"]]
kidlt6=1

# Vamos construir uma matriz de dados para a predicao
our_pred = as.matrix(data.frame(
husage=husage,
husunion=husunion,
husearns=husearns,
huseduc=huseduc,
husblck=husblck,
hushisp=hushisp,
hushrs=hushrs,
kidge6=kidge6,

```

```

age=age,
black=black,
educ=educ,
hispanic=hispanic,
union=union,
exper=exper,
kidlt6=kidlt6
))

# Vamos fazer a predicao com base nos parametros que
# selecionamos
predict_our_elastic <- predict(elastic_reg,our_pred)
predict_our_elastic

# Novamente, o resultado eh padronizado, nos temos que
# reverte-lo para o nivel dos valores originais do
# dataset, vamos fazer isso:
wage_pred_elastic=(predict_our_elastic*
                    pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]
wage_pred_elastic

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

# Base      Number      Log      Antilog
# n          x      log(x, n)      x^n
# e          x      log(x)      exp(x)
# 10         x      log10(x)      10^x
cat("RESULTADOS\n\n")

cat("### REGRESSÃO RIDGE ###")
cat("\nSalário hora médio da Esposa é ",exp(wage_pred_ridge)," e pode
variar entre ",exp(CIlwr_ridge)," e ",exp(CIupr_ridge)," \n\n")

cat("### REGRESSÃO LASSO ###")
cat("\nSalário hora médio da Esposa é ",exp(wage_pred_lasso)," e pode

```



```

variar entre ",exp(CIlwr_lasso)," e ",exp(CIupr_lasso),"\\n\\n")

cat("### REGRESSÃO ELASTICNET ###")
cat("\\nSalário hora médio da Esposa é ",exp(wage_pred_elastic)," e
pode variar entre ",exp(CIlwr_elastic)," e ",exp(CIupr_elastic))
# antilog lwage
lw <- dat$lwage
exp(lw)

```

```

Rows: 2,574
Columns: 17
$ husage    <dbl> 56, 31, 33, 34, 42, 45, 33, 31, 31, 44, 45, 22, 66, 43, 26, 3...
$ husunion  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0...
$ husearns  <dbl> 1500, 800, 950, 1000, 730, 1154, 1350, 769, 340, 750, 1200, 2...
$ huseduc   <dbl> 14, 17, 13, 14, 14, 16, 16, 18, 12, 12, 12, 12, 16, 12, 14, 1...
$ husblck   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
$ hushisp   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
$ hushrs    <dbl> 40, 40, 60, 50, 40, 38, 40, 55, 40, 40, 50, 40, 40, 50, 36, 5...
$ kidge6    <dbl> 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0...
$ earns     <dbl> 100, 480, 455, 102, 300, 425, 770, 125, 245, 539, 300, 299, 5...
$ age       <dbl> 49, 29, 30, 31, 41, 45, 32, 27, 30, 42, 42, 23, 55, 31, 27, 3...
$ black     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
$ educ      <dbl> 12, 14, 12, 12, 12, 18, 12, 14, 15, 12, 12, 13, 12, 12, 14, 1...
$ hispanic  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
$ union     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0...
$ exper     <dbl> 31, 9, 12, 13, 23, 21, 14, 7, 9, 24, 24, 4, 37, 13, 7, 12, 31...
$ kidlt6    <dbl> 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1...
$ lwage     <dbl> 1.897120, 2.484907, 2.431418, 1.629241, 2.302585, 2.496741, 2...

```

```

Time difference of 0.005434752 secs
15 x 1 sparse Matrix of class "dgCMatrix"
      s0
husage  -0.004745469
husunion 0.001301385
husearns 0.230241265
huseduc  0.051370292
husblck  0.217056671
hushisp  0.086144980
hushrs   -0.069507967
kidge6   -0.181910965
age       0.069266121
black    -0.277110253
educ      0.327096706
hispanic -0.011406010
union     0.365669412
exper    -0.016268996
kidlt6   -0.042460770

```

A data.frame: 1 × 2

RMSE	Rsquare
<dbl>	<dbl>
0.8411446	0.292132

A data.frame: 1 × 2

RMSE	Rsquare
<dbl>	<dbl>
0.9893328	0.259084

A matrix: 1 × 1 of type dbl

s1
-0.06843352

A matrix: 1 × 1 of type dbl

s1
2.161212

A matrix: 1 × 1 of type dbl

s1
2.139354

A matrix: 1 × 1 of type dbl

s1
2.18307

RMSE Rsquare

<dbl> <dbl>

0.8423523 0.2900978

RMSE	Rsquare
<dbl>	<dbl>
0.9899383	0.2581767

A matrix: 1 × 1 of type dbl

s1

-0.2120462

A matrix: 1 × 1 of type dbl

s1

2.088536

A matrix: 1 × 1 of type dbl

s1

2.066677

A matrix: 1 × 1 of type dbl

s1

2.110394

```
+ Fold01.Rep1: alpha=0.37885, lambda=0.008378
- Fold01.Rep1: alpha=0.37885, lambda=0.008378
+ Fold01.Rep1: alpha=0.80441, lambda=1.497197
- Fold01.Rep1: alpha=0.80441, lambda=1.497197
+ Fold01.Rep1: alpha=0.51148, lambda=0.005422
- Fold01.Rep1: alpha=0.51148, lambda=0.005422
+ Fold01.Rep1: alpha=0.62836, lambda=3.578862
- Fold01.Rep1: alpha=0.62836, lambda=3.578862
+ Fold01.Rep1: alpha=0.74298, lambda=0.012335
- Fold01.Rep1: alpha=0.74298, lambda=0.012335
+ Fold01.Rep1: alpha=0.76859, lambda=0.264158
- Fold01.Rep1: alpha=0.76859, lambda=0.264158
+ Fold01.Rep1: alpha=0.04725, lambda=0.804106...
```

A data.frame: 1 × 2

	alpha	lambda
	<dbl>	<dbl>
7	0.7429763	0.01233549

```
[[ suppressing 77 column names 's0', 's1', 's2' ... ]]
15 x 77 sparse Matrix of class "dgCMatrix"

husage . . . . .
husunion . . . . .
husearns . . . 0.005265248 0.02487154
0.04296476
huseduc . . . . .
husblk . . . . .
hushisp . . . . .
hushrs . . . . .
kidge6 . . . . .
age . . . . .
black . . . . .
educ . 0.03507813 0.06778955 0.0982384 0.125145010 0.14617055
0.16558804
hispanic . . . . .
union . . . . .
exper . . . . .
kidlt6 . . . . .

husage . . . . .
husunion . . . . .
husearns 0.05963892 0.0750015 0.08913756 0.1021342 0.1140741 0.12505948
huseduc . . . . .
husblk . . . . .
hushisp . . . . .
hushrs . . . . .
kidge6 . . . . .
age . . . . .
black . . . . .
educ 0.18350187 0.2000085 0.21520547 0.2291845 0.2420330 0.25318972
hispanic . . . . .
union . . . . 0.01387615
exper . . . . .
kidlt6 . . . . .

husage . . . . .
husunion . . . . .
husearns 0.13515562 0.14442264 0.15291828 0.1607027 0.1678319
0.173833446
huseduc . . . . .
0.002054596
```

```

husblk . . . . .
hushisp . . . . .
hushrs . . . . .
kidge6 . . . . .
age . . . . .
black . . . . .
educ 0.26257786 0.27117395 0.27904309 0.2862438 0.2928303
0.297810945
hispanic . . . . .
union 0.04486632 0.07331676 0.09941872 0.1233520 0.1452849
0.165612373
exper . . . . .
kidlt6 . . . . .

husage . . . . .
husunion . . . . .
husearns 0.179156492 0.1840082886 0.188394608 0.192895267 0.19701187
huseduc 0.004588311 0.0070668547 0.009691862 0.012127439 0.01426328
husblk . . . . .
hushisp . . . . .
hushrs . . . -0.003737866 -0.009837365 -0.01541787
kidge6 . . -0.0070599317 -0.021563146 -0.034670629 -0.04664962
age . . 0.0009975204 0.006234541 0.010683673 0.01475270
black . . . . .
educ 0.302025726 0.3057289826 0.309131553 0.312206014 0.31506292
hispanic . . . . .
union 0.184302326 0.2014263463 0.216225946 0.229837789 0.24226937
exper . . . . .
kidlt6 . . . . .

husage . . . . .
husunion . . . . .
husearns 0.20076772 0.20420079 0.20733811 0.21020443 0.21282254
huseduc 0.01627001 0.01809984 0.01976777 0.02128807 0.02267374
husblk . . . . .
hushisp . . . . .
hushrs -0.02051792 -0.02518001 -0.02944058 -0.03333320 -0.03688880
kidge6 -0.05761321 -0.06763611 -0.07679628 -0.08516577 -0.09281097
age 0.01846904 0.02186304 0.02496201 0.02779107 0.03037329
black . . . . .
educ 0.31763607 0.31998310 0.32212393 0.32407648 0.32585714
hispanic . . . . .
union 0.25363908 0.26402769 0.27351750 0.28218434 0.29009797
exper . . . . .
kidlt6 . . . . .

```

husage	.	.	.	.	.
husunion	.	.	.	.	.
husearns	0.20076772	0.20420079	0.20733811	0.21020443	0.21282254
huseduc	0.01627001	0.01809984	0.01976777	0.02128807	0.02267374
husblk	.	.	.	.	.
hushisp	.	.	.	.	.
hushrs	-0.02051792	-0.02518001	-0.02944058	-0.03333320	-0.03688880
kidge6	-0.05761321	-0.06763611	-0.07679628	-0.08516577	-0.09281097
age	0.01846904	0.02186304	0.02496201	0.02779107	0.03037329
black	.	.	.	.	.
educ	0.31763607	0.31998310	0.32212393	0.32407648	0.32585714
hispanic	.	.	.	.	.
union	0.25363908	0.26402769	0.27351750	0.28218434	0.29009797
exper	.	.	.	.	.
kidlt6	.	.	.	.	.

husage	.	.	.	.	.
husunion	.	.	.	.	.
husearns	0.21521343	0.21739639	0.21938914	0.22120797	0.22286781
huseduc	0.02393668	0.02508771	0.02613673	0.02709276	0.02796402
husblk	.	.	.	.	.
hushisp	.	.	.	.	.
hushrs	-0.04013585	-0.04310055	-0.04580695	-0.04827715	-0.05053142
kidge6	-0.09979302	-0.10616816	-0.11198806	-0.11730018	-0.12214810
age	0.03272985	0.03488016	0.03684202	0.03863175	0.04026426
black	.	.	.	.	.
educ	0.32748093	0.32896155	0.33031154	0.33154234	0.33266443
hispanic	.	.	.	.	.
union	0.29732250	0.30391684	0.30993501	0.31542658	0.32043698
exper	.	.	.	.	.
kidlt6	.	.	.	.	.

husage	.	.	.	.	.
husunion	.	.	.	.	.
husearns	0.224255907	0.22548974	0.22661144	0.22763380	0.22856610
huseduc	0.028776704	0.02942079	0.03008110	0.03068577	0.03123698
husblk	.	.	.	.	.
hushisp	.	.	.	.	.
hushrs	-0.052597485	-0.05448836	-0.05621127	-0.05778263	-0.05921578
kidge6	-0.126558307	-0.13056251	-0.13422616	-0.13756844	-0.14061685
age	0.041736540	0.04307692	0.04429700	0.04540955	0.04642395
black	-0.005667735	-0.01249777	-0.01873372	-0.02442181	-0.02960940
educ	0.333662297	0.33462474	0.33545846	0.33621681	0.33690793
hispanic	.	.	.	.	.
union	0.325283734	0.32977157	0.33387612	0.33762020	0.34103483
exper	.	.	.	.	.
kidlt6	.	.	.	.	.

husage	.	.	.	.	.
husunion	.	.	.	.	.
husearns	0.229398367	0.230110780	0.230763435	0.23135795	0.23189986
huseduc	0.031865943	0.032632851	0.033397497	0.03409776	0.03473649
husblk	.	.	.	.	.
hushisp	0.002352205	0.009404264	0.015816930	0.02166397	0.02699579
hushrs	-0.060499328	-0.061583307	-0.062572081	-0.06347341	-0.06429510
kidge6	-0.143599570	-0.147492078	-0.151073549	-0.15434126	-0.15732181
age	0.047280877	0.047556884	0.047789802	0.04800120	0.04819327
black	-0.034179442	-0.037894696	-0.041286264	-0.04437836	-0.04719700
educ	0.337577891	0.338327905	0.338968382	0.33955035	0.34008061
hispanic	.	.	.	.	.
union	0.344072673	0.346733717	0.349174360	0.35140012	0.35342937
exper	.	.	.	.	.
kidlt6	-0.000576432	-0.004424469	-0.007963405	-0.01119284	-0.01413909

```

husage -0.01181193 -0.01220566 -0.012596360
husunion . . .
husearns 0.23757467 0.23762223 0.237664496
huseduc 0.04260468 0.04264427 0.042683515
husblck 0.30543159 0.30807286 0.310710515
hushisp 0.08515599 0.08617269 0.087127411
hushrs -0.07238293 -0.07243357 -0.072479390
kidge6 -0.18699443 -0.18715784 -0.187310237
age 0.06014661 0.06049525 0.060840486
black -0.36548501 -0.36815157 -0.370801517
educ 0.34400477 0.34401378 0.344020123
hispanic -0.00387763 -0.00475531 -0.005573872
union 0.37089203 0.37096443 0.371028317
exper . . .
kidlt6 -0.04409694 -0.04428998 -0.044471566

```

A data.frame: 1 × 2

RMSE	Rsquare
<dbl>	<dbl>
0.8419625	0.2907546

A data.frame: 1 × 2

RMSE	Rsquare
<dbl>	<dbl>
0.9893722	0.259025

Resultados:

- **Ridge:** salário hora médio da esposa é 8.681654 e pode variar entre 8.493945 e 8.87351
- **Lasso:** salário hora médio da esposa é 8.073087 e pode variar entre 7.898536 e 8.251494
- **Elastic Net:** salário hora médio da esposa é 8.003155 e pode variar entre 7.830116 e 8.180017

Com base nos resultados apresentados, podemos observar que os três modelos (Ridge, Lasso e Elastic Net) estimam valores próximos para o salário hora médio da esposa, com um intervalo de confiança semelhante.

Ridge:

- Salário hora médio estimado: R\$8,68
- Intervalo de confiança: R\$8,49 a R\$ 8,87

Lasso:

- Salário hora médio estimado: R\$8,07
- Intervalo de confiança: R\$7,90 a R\$8,25

Elastic Net:

- Salário hora médio estimado: R\$ 8,00
- Intervalo de confiança: R\$ 7,83 a R\$ 8,18

Análise Comparativa:

- **Ridge:**
 - Maior valor de salário hora médio estimado (R\$ 8,68).
 - Maior incerteza na estimativa (intervalo de confiança: R\$ 8,49 a R\$ 8,87).
 - Captura mais ruído nos dados, levando à maior incerteza na estimativa.
- **Lasso:**
 - Menor valor de salário hora médio estimado (R\$ 8,07).
 - Maior precisão na estimativa (intervalo de confiança: R\$ 7,90 a R\$ 8,25).
 - Realiza seleção eficiente de variáveis, resultando em estimativa mais precisa.
- **Elastic Net:**
 - Valor de salário hora médio estimado próximo ao Lasso (R\$ 8,00).
 - Bom nível de precisão e robustez (intervalo de confiança: R\$ 7,83 a R\$ 8,18).
 - Combina as vantagens do Ridge e do Lasso, oferecendo precisão, robustez e interpretabilidade.

Elastic Net é o modelo ideal devido à:

- Combinação de precisão do Lasso e robustez do Ridge.
- Maior capacidade de lidar com multicolinearidade e outliers nos dados.
- Boa interpretabilidade dos resultados, facilitando a identificação das variáveis mais relevantes.

APÊNDICE F - ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

1 Construção de Características: Identificador automático de idioma

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

espanhol = [
```

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

portugues = [
 "Estou indo para o trabalho agora.",
 "Adoro passar tempo com minha família.",
 "Preciso comprar leite e pão.",
 "Vamos ao cinema no sábado.",
 "Gosto de praticar esportes ao ar livre.",
 "O trânsito está terrível hoje.",
 "A comida estava deliciosa!",
 "Você já visitou o Rio de Janeiro?",
]

```

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

```

```
import random
```

```

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

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

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

```

```
import pandas as pd
```

```

dados = pd.DataFrame(pre_padroes)
dados
# a entrada é o vetor pre_padroes e a saída desse passo deverá ser
"padrões"

# https://scikit-learn.org/stable/modules/feature\_extraction.html

import re
from nltk import ngrams
from collections import Counter
import statistics

def tamanhoMedioFrases(texto):
    texto = texto.lower()
    palavras = re.split("\s", texto)
    # print(palavras)
    tamanhos = [len(s) for s in palavras if len(s)>0]
    # print(tamanhos)
    soma = 0
    for t in tamanhos:
        soma=soma+t
    return soma / len(tamanhos)

def quantidadePalavrasComCaracterEspecial(texto):
    texto = texto.lower()
    quantidade = 0
    palavras = re.split("\s", texto)
    for palavra in palavras:
        if re.search(r'[áéíóúñçãõâêîôû]', palavra):
            quantidade += 1 # conta palavras com caracter especial
        # print(texto, palavra, quantidade)
    return quantidade

# Ocorrências de dois símbolos sucessivos
def contaSimboloSucessivo(texto):
    texto = texto.lower()
    regex = r'(\w)\1' # expressão regular para encontrar dois símbolos
    sucessivos
    ocorrencias = re.findall(regex, texto) # encontrar símbolos
    sucessivos no texto
    # print(ocorrencias, texto)

```

```

    return len(ocorrencias) # retorna a quantidade de ocorrências

def contarSufixosPT(texto):
    sufixos_comuns_pt = {"ão", "aço", "alhão", "aréu", "arra",
    "arrão", "eirão", "uça", "inho", "zinho", "acho", "icho", 'icha',
    "eco", "ela", "ote", "isco"}
    palavras = texto.split()
    count = sum(1 for palavra in palavras for sufixo in
    sufixos_comuns_pt if palavra.lower().endswith(sufixo))
    return count

def contarSufixosES(texto):
    sufixos_comuns_es = {"ito", "ita", "illo", "illa", "ón", "ona",
    "azo", "aza", "ucho", "ucha", "aco", "aca"}
    palavras = texto.split()
    count = sum(1 for palavra in palavras for sufixo in
    sufixos_comuns_es if palavra.lower().endswith(sufixo))
    return count

def contarSufixosIN(texto):
    sufixos_comuns_in = {"able", "age", "al", "ant", "cy", "er",
    "ery", "ry", "est", "ful", "ing", "ism", "ity", "less", "y", "tion"}
    palavras = texto.split()
    count = sum(1 for palavra in palavras for sufixo in
    sufixos_comuns_in if palavra.lower().endswith(sufixo))
    return count

def extraiCaracteristicas(frase):
    # frase é um vetor [ 'texto', 'lingua' ]
    texto = frase[0]
    pattern_regex = re.compile('[^\w+]', re.UNICODE)
    texto = re.sub(pattern_regex, ' ', texto)
    # print(texto)
    caracteristica1=tamanhoMedioFrases(texto)
    caracteristica2=quantidadePalavrasComCaracterEspecial(texto)
    caracteristica3=contaSimboloSucessivo(texto)
    caracteristica4=contarSufixosPT(texto)
    caracteristica5=contarSufixosES(texto)
    caracteristica6=contarSufixosIN(texto)
    # acrescenta as suas funcoes no vetor padrao
    padrao = [caracteristica1, caracteristica2, caracteristica3,
    caracteristica4, caracteristica5, caracteristica6, frase[1] ]

```

```

    return padrao

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

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

#
# apenas para visualizacao
print(padroes)
dados = pd.DataFrame(padroes)
dados
from sklearn.model_selection import train_test_split
import numpy as np
#from sklearn.metrics import confusion_matrix
vet = np.array(padroes)
classes = vet[:, -1]
# classes = [p[-1] for p in padroes]
# print(classes)
padroes_sem_classe = vet[:, 0:-1]
# print(padroes_sem_classe)
X_train, X_test, y_train, y_test =
train_test_split(padroes_sem_classe, classes, test_size=0.25,
stratify=classes)

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

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

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

```

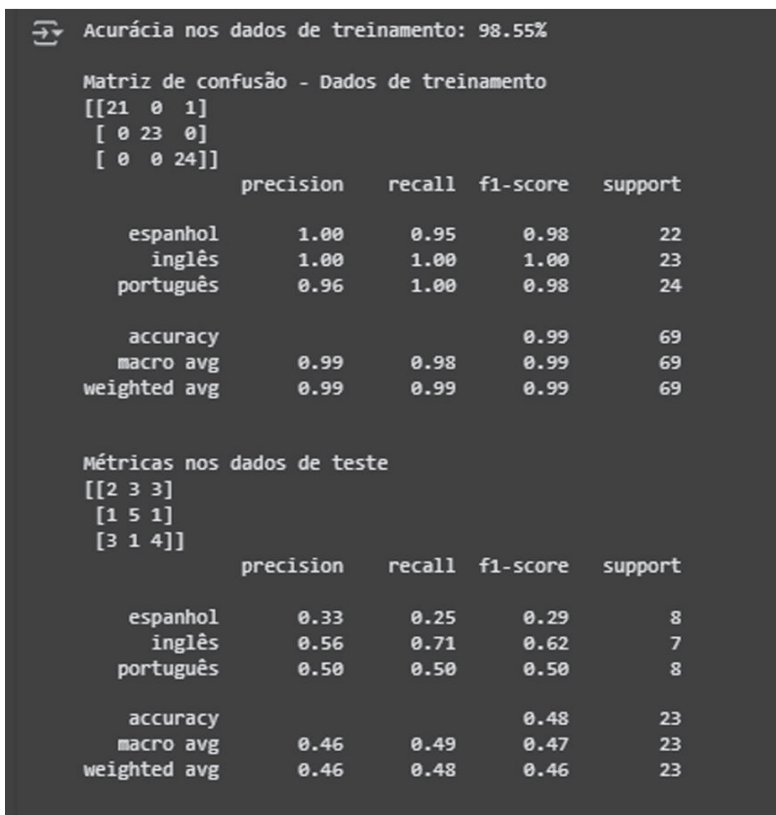
```

print("Acurácia nos dados de treinamento: {:.2f}%".format(acuracia *
100))

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

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

```



Acurácia nos dados de treinamento: 98.55%

Matriz de confusão - Dados de treinamento

```
[[21  0  1]
 [ 0 23  0]
 [ 0  0 24]]
```

	precision	recall	f1-score	support
espanhol	1.00	0.95	0.98	22
inglês	1.00	1.00	1.00	23
português	0.96	1.00	0.98	24
accuracy			0.99	69
macro avg	0.99	0.98	0.99	69
weighted avg	0.99	0.99	0.99	69

Métricas nos dados de teste

```
[[2 3 3]
 [1 5 1]
 [3 1 4]]
```

	precision	recall	f1-score	support
espanhol	0.33	0.25	0.29	8
inglês	0.56	0.71	0.62	7
português	0.50	0.50	0.50	8
accuracy			0.48	23
macro avg	0.46	0.49	0.47	23
weighted avg	0.46	0.48	0.46	23

2 Melhore uma base de dados ruim

Base utilizada câncer de mama e Algoritmo escolhido svm.

```
import requests, zipfile, io
```



```

from io import BytesIO
import numpy as np
import pandas as pd

# colunas do dataset

heart_colunas = ['age', 'anaemia', 'creatinine_phosphokinase',
                 'diabetes', 'ejection_fraction',
                 'high_blood_pressure', 'platelets',
                 'serum_creatinine', 'serum_sodium', 'sex',
                 'smoking', 'time', 'death_event']

# Baixar e extrair o dataset

r =
requests.get('https://archive.ics.uci.edu/static/public/519/heart+fa
  ilure+clinical+records.zip')
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()
# Carregando o conjunto de dados de insuficiência cardíaca em um
DataFrame do Pandas
heart_failure =
pd.read_csv('heart_failure_clinical_records_dataset.csv')
# Renomeando as colunas conforme definido anteriormente
heart_failure.columns = heart_colunas
# Verificando as primeiras linhas
heart_failure.head()
# Descrevendo estatísticas do DataFrame
heart_failure.describe()
# Visualizando informações do DataFrame
# Dataset sem nulos
heart_failure.info()

Classificar as features
# features = ['age', 'ejection_fraction', 'serum_creatinine',
'serum_sodium']
X = heart_failure.drop(columns='death_event')
# X = heart_failure[features]
print("Features:\n")
X.info()

```

```

print("\n#####\n")

# Classificar o target
Y = heart_failure['death_event']
print("\nTarget:\n")
Y.info()

# Y original
Y_orig = heart_failure['death_event']

from sklearn.preprocessing import scale
# from sklearn.preprocessing import minmax_scale, StandardScaler
from sklearn.preprocessing import MinMaxScaler, StandardScaler,
RobustScaler, MaxAbsScaler
import pandas as pd

# Min-Max Scaling (Normalização)
# https://scikit
learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScale
r.html
# This estimator scales and translates each feature individually
such that it is in the given range on the training set, e.g. between
zero and one.
scaler_minmax = MinMaxScaler()
X_minmax = pd.DataFrame(scaler_minmax.fit_transform(X),
columns=X.columns)
# Standardize features by removing the mean and scaling to unit
variance.
scaler_standard = StandardScaler()
X_standard = pd.DataFrame(scaler_standard.fit_transform(X),
columns=X.columns)

# Robust Scaling (Padronização)
# https://scikit
learn.org/stable/modules/generated/sklearn.preprocessing.RobustScale
r.html
# This Scaler removes the median and scales the data according to
the quantile range (defaults to IQR: Interquartile Range). The IQR
is the range between the 1st quartile (25th quantile) and the 3rd
quartile (75th quantile).
scaler_robust = RobustScaler()

```

```

X_robust = pd.DataFrame(scaler_robust.fit_transform(X),
                        columns=X.columns)

# Max Scaling (Normalização)
# https://scikit
learn.org/stable/modules/generated/sklearn.preprocessing.MaxAbsScale
r.html
# Scale each feature by its maximum absolute value.
scaler_max = MaxAbsScaler()
X_max = pd.DataFrame(scaler_max.fit_transform(X), columns=X.columns)

# Exibindo as primeiras linhas das features normalizadas com Min-Max
Scaling
print("Min-Max Scaling:\n", X_minmax.head())

# Exibindo as primeiras linhas das features padronizadas
print("Standard Scaling:\n", X_standard.head())

# Exibindo as primeiras linhas das features escalonadas de forma
robusta
print("Robust Scaling:\n", X_robust.head())

# Exibindo as primeiras linhas das features normalizadas L2
print("Max Scaling:\n", X_max.head())
from sklearn.model_selection import train_test_split
import numpy as np

# com os dados tratados
X_train, X_test, y_train, y_test = train_test_split(X, Y,
                                                    test_size=0.25, stratify=Y, random_state=10)

# MIN-MAX
X_MinMax_train, X_MinMax_test, y_MinMax_train, y_MinMax_test =
train_test_split(X_minmax, Y, test_size=0.25, stratify=Y,
random_state=10)

# STANDARD
X_standard_train, X_standard_test, y_standard_train, y_standard_test
= train_test_split(X_standard, Y, test_size=0.25, stratify=Y,
random_state=10)

```

```

# ROBUST
X_robust_train, X_robust_test, y_robust_train, y_robust_test =
train_test_split(X_robust, Y, test_size=0.25, stratify=Y,
random_state=10)

# MAX
X_max_train, X_max_test, y_max_train, y_max_test =
train_test_split(X_max, Y, test_size=0.25, stratify=Y,
random_state=10)
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

## ORIGINAL ###

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

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

# predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(X_train)
cm_orig_train = confusion_matrix(y_train, y_orig_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no
TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_train, y_orig_pred))

# score com os dados de treinamento
acuracia = modelo_orig.score(X_test, y_test)

# print("Acurácia nos dados de treinamento (Modelo Original):
{:.2f}%".format(acuracia * 100), "\n")

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para
TESTES')
y2_orig_pred = modelo_orig.predict(X_test)

```

```

cm_orig_test = confusion_matrix(y_test, y2_orig_pred)
print(cm_orig_test)
print(classification_report(y_test, y2_orig_pred))
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
### MIN-MAX ###
treinador = svm.SVC() #algoritmo escolhido
# Modelo com Min-Max
modelo_minmax = treinador.fit(X_MinMax_train, y_MinMax_train)
# score com os dados de treinamento
acuracia_minmax = modelo_minmax.score(X_MinMax_train,
y_MinMax_train)
print("Acurácia nos dados de treinamento (Min-Max):
{:.2f}%".format(acuracia_minmax * 100), "\n")
# predição com os mesmos dados usados para treinar
y_pred = modelo_minmax.predict(X_MinMax_train)
cm_train = confusion_matrix(y_MinMax_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no
TREINAMENTO')
print(cm_train)
print(classification_report(y_MinMax_train, y_pred))

# score com os dados de treinamento
# acuracia_minmax = modelo_minmax.score(X_MinMax_test,
y_MinMax_test)
# print("Acurácia nos dados de treinamento (Min-Max):
{:.2f}%".format(acuracia_minmax * 100), "\n")

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para
TESTES')
y2_pred = modelo_minmax.predict(X_MinMax_test)
cm_test = confusion_matrix(y_MinMax_test, y2_pred)
print(cm_test)
print(classification_report(y_MinMax_test, y2_pred))
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

```

```

#### STANDARD ####

treinador = svm.SVC() #algoritmo escolhido
# Modelo com Standard
modelo_standard = treinador.fit(X_standard_train, y_standard_train)

# score com os dados de treinamento
acuracia_standard = modelo_standard.score(X_standard_train,
y_standard_train)
print("Acurácia nos dados de treinamento (Standard):
{:.2f}%".format(acuracia_standard * 100), "\n")

# predição com os mesmos dados usados para treinar
y_pred = modelo_standard.predict(X_standard_train)
cm_train = confusion_matrix(y_standard_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no
TREINAMENTO')
print(cm_train)
print(classification_report(y_standard_train, y_pred))

# score com os dados de treinamento
# acuracia_standard = modelo_standard.score(X_standard_test,
y_standard_test)
# print("Acurácia nos dados de treinamento (Standard):
{:.2f}%".format(acuracia_standard * 100), "\n")

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para
TESTES')
y2_pred = modelo_standard.predict(X_standard_test)
cm_test = confusion_matrix(y_standard_test, y2_pred)
print(cm_test)
print(classification_report(y_standard_test, y2_pred))
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

#### ROBUST ####

treinador = svm.SVC() #algoritmo escolhido

```

```

# Modelo com Robust
modelo_robust = treinador.fit(X_robust_train, y_robust_train)

# score com os dados de treinamento
acuracia_robust = modelo_robust.score(X_robust_train,
y_robust_train)
print("Acurácia nos dados de treinamento (Robust):
{:.2f}%".format(acuracia_robust * 100), "\n")

# predição com os mesmos dados usados para treinar
y_pred = modelo_robust.predict(X_robust_train)
cm_train = confusion_matrix(y_robust_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no
TREINAMENTO')
print(cm_train)
print(classification_report(y_robust_train, y_pred))
# score com os dados de treinamento
# acuracia_robust = modelo_robust.score(X_robust_test,
y_robust_test)
# print("Acurácia nos dados de treinamento (Robust):
{:.2f}%".format(acuracia_robust * 100), "\n")

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para
TESTES')
y2_pred = modelo_robust.predict(X_robust_test)
cm_test = confusion_matrix(y_robust_test, y2_pred)
print(cm_test)
print(classification_report(y_robust_test, y2_pred))
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

### MAX ###

treinador = svm.SVC() #algoritmo escolhido

# Modelo com Max
modelo_max = treinador.fit(X_max_train, y_max_train)
acuracia_max = modelo_max.score(X_max_train, y_max_train)
print("Acurácia nos dados de treinamento (MAX):

```

```

{:.2f}%".format(acuracia_max * 100), "\n")
# predição com os mesmos dados usados para treinar
y_pred = modelo_max.predict(X_max_train)
cm_train = confusion_matrix(y_max_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no
TREINAMENTO')
print(cm_train)
print(classification_report(y_max_train, y_pred))
# score com os dados de treinamento
# acuracia_max = modelo_l2.score(X_l2_test, y_l2_test)
# print("Acurácia nos dados de treinamento (MAX):
{:.2f}%".format(acuracia_l2 * 100), "\n")
# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para
TESTES')
y2_pred = modelo_max.predict(X_max_test)
cm_test = confusion_matrix(y_max_test, y2_pred)
print(cm_test)
print(classification_report(y_max_test, y2_pred))
### Prediction ###

import warnings
warnings.simplefilter("ignore")

# print(heart_failure.head(10))

primeiras_linhas = heart_failure.head(10) # death = 1
ultimas_linhas = heart_failure.tail(10) # death = 0
predict = pd.concat([primeiras_linhas, ultimas_linhas]) # dataset
predict = predict.drop(columns=['death_event']) # sem target

# print("primeiras_linhas \n", primeiras_linhas)
# print("ultimas_linhas \n", ultimas_linhas)

# Min-Max Scaling
scaler_minmax = MinMaxScaler()
predict_minmax = pd.DataFrame(scaler_minmax.fit_transform(predict),
columns=predict.columns)
print(modelo_minmax.predict(predict_minmax))

# Standard Scaling

```



```

scaler_standard = StandardScaler()
predict_standard =
pd.DataFrame(scaler_standard.fit_transform(predict),
columns=predict.columns)
print(modelo_standard.predict(predict_standard))

# Robust Scaling
scaler_robust = RobustScaler()
predict_robust = pd.DataFrame(scaler_robust.fit_transform(predict),
columns=predict.columns)
print(modelo_robust.predict(predict_robust))

# MAX Scaling
normalizer_max = MaxAbsScaler()
predict_max = pd.DataFrame(normalizer_max.fit_transform(predict),
columns=predict.columns)
print(modelo_max.predict(predict_max))

print("Acurácia nos dados de treinamento (Modelo Original):
{:.2f}%".format(acuracia * 100), "\n")
print("Acurácia nos dados de treinamento (Min-Max):
{:.2f}%".format(acuracia_minmax * 100), "\n")
print("Acurácia nos dados de treinamento (Standard):
{:.2f}%".format(acuracia_standard * 100), "\n")
print("Acurácia nos dados de treinamento (Robust):
{:.2f}%".format(acuracia_robust * 100), "\n")
print("Acurácia nos dados de treinamento (MAX):
{:.2f}%".format(acuracia_max * 100), "\n")
import seaborn as sns
sns.pairplot(heart_failure, hue='death_event', height=2)

```

```

Acurácia nos dados de treinamento (Standard): 92.41%
Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO
[[148  4]
 [ 13 59]]
      precision    recall  f1-score   support

     0       0.92      0.97      0.95       152
     1       0.94      0.82      0.87        72

   accuracy          0.92       224
  macro avg       0.93      0.90      0.91       224
 weighted avg     0.92      0.92      0.92       224

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[46  5]
 [ 8 16]]
      precision    recall  f1-score   support

     0       0.85      0.90      0.88        51
     1       0.76      0.67      0.71        24

   accuracy          0.83       75
  macro avg       0.81      0.78      0.79       75
 weighted avg     0.82      0.83      0.82       75

```

APÊNDICE G - APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

```
### Veículos ###

### Leitura dos dados
dados <- read.csv("6 - Veiculos - Dados.csv")

### Retira o atributo ID
dados$a <- NULL

head(dados)

cat("\n \n ##### Usando Cross-Validation ##### \n \n")

#### Cross-validation SVM
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
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(202485)
svm <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl,
tuneGrid=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 - Dados - Novos Casos.csv")
View(dados_novos_casos)
predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
resultado$tipo <- NULL
View(resultado)

### RNA ###

### Diabetes ###

### Leitura dos dados
temp_dados <- read.csv("10 - Diabetes - Dados.csv")

# head(temp_dados)

### Retira o atributo ID e Missing Values
temp_dados$num <- NULL
imp <- mice(temp_dados)
dados <- complete(imp, 1)

head(dados)

### Criar bases de Treino e Teste
set.seed(202485)
indices <- createDataPartition(dados$diabetes, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

# Usando Cross-validation
cat("\n \n ##### Usando Cross-validation ##### \n \n")

### indica o método cv e numero de folders 10
ctrl <- trainControl(method = "cv", number = 10)

```

```

### executa a RNA com esse ctrl
set.seed(202485)
rna <- train(diabetes~., data=treino, method="nnet",trace=FALSE,
trControl=ctrl)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))

# Parametrização da RNA
### size, decay
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202485)
rna <- train(
form = diabetes~. ,
data = treino ,
method = "nnet" ,
tuneGrid = grid ,
trControl = ctrl ,
maxit = 2000,trace=FALSE)

### Verifica o resultado do Treinamento
rna

### Faz as predições e mostra matriz de confusão
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))

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

predict.rna <- predict(rna, dados_novos_casos)
dados_novos_casos$diabetes <- NULL
resultado <- cbind(dados_novos_casos, predict.rna)
View(resultado)

### RNA ###
### Admissão ###

```

```

### Leitura dos dados
dados <- read.csv("9 - Admissao - Dados.csv",header=T)
### Retira o atributo ID
dados$num <- NULL

head(dados)

### Cria arquivos de treino e teste
set.seed(202485)
ind <- createDataPartition(dados$ChanceOfAdmit, p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]

cat("\nRNA com CV\n")

### CV e parametrizacao da RNA
control <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(size = seq(from = 1, to = 10, by = 1), decay =
seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202485)
rna <- train(ChanceOfAdmit~., data=treino, method="nnet",
trainControl=control, tuneGrid=tuneGrid, linout=T,
MaxNWts=10000, maxit=2000, trace=F)
rna

### Predições e métricas
predict.rna <- predict(rna, teste)

cat("\n### RESULTADO DO R2 ###\n")
r2 <- function(predito, observado) {
return(1 - (sum((predito-observado)^2) / sum((observado
mean(observado))^2)))
}
r2(predict.rna, teste$ChanceOfAdmit)

cat("\n### RESULTADO DO SYX ###\n")
syx <- function(predito, observado, n) {
return( sqrt( sum((predito - observado)^2) / ( n - 2 ) ) )
}
syx(predict.rna, teste$ChanceOfAdmit, length(teste$ChanceOfAdmit))

```

```

cat("\n### RESULTADO DO PEARSON ###\n")
pearson <- function(predito, observado) {
  return (sum((predito - mean(predito)) * (observado - mean(observado))) /
    (sqrt(sum((predito - mean(predito))^2)) * sqrt(sum((observado -
    mean(observado))^2))))
}
pearson(predict.rna, teste$ChanceOfAdmit)

```

```

cat("\n### RESULTADO DO RMSE ###\n")
rmse <- function(predito, observado, n) {
  return( sqrt( 1/n * sum((observado - predito)^2 ) ) )
}
rmse(predict.rna, teste$ChanceOfAdmit, length(teste$ChanceOfAdmit))

```

```

cat("\n### RESULTADO DO MAE ###\n")
mae <- function(predito, observado, n) {
  return( ( 1/n * sum( (observado - predito) ) ) )
}
mae(predict.rna, teste$ChanceOfAdmit, length(teste$ChanceOfAdmit))

```

```

### PREDIÇÕES DE NOVOS CASOS
cat("\n### PREDIÇÕES DE NOVOS CASOS - RNA COM CV ###\n")
dados_novos_casos <- read.csv("9 - Admissao - Dados - Novos Casos.csv")
head(dados_novos_casos)

```

```

dados_novos_casos$ChanceOfAdmit <- NULL
predict.rna <- predict(rna, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.rna)
View(resultado)

```

```

### GRÁFICO DE RESÍDUOS
cat("\n### GRÁFICO DE RESÍDUOS - RNA COM CV ###\n")

```

```

residuo <- (resid(rna) * 100)
residuo

```

```

rna <- fitted(rna)

```

```

plot(rna, residuo,
      xlab = "Valores Ajustados",

```

```

    ylab = "Resíduos Percentuais",
    main = "Gráfico de Resíduos Percentuais",
    ylim = c(-100, 100),
    yaxt = "n") # Desabilitar os ticks padrão do eixo y

# Adicionar uma linha horizontal em y = 0
abline(h = 0, col = "red")

# Adicionar ticks personalizados no eixo y
axis(side = 2, at = seq(-100, 100, by = 10))

### RF ###
### Biomassa ###
### Leitura dos dados
dados <- read.csv("5 - Biomassa - Dados.csv",header=T)
### Retira o atributo ID
#dados$num <- NULL
head(dados)

### Cria arquivos de treino e teste
set.seed(202485)
ind <- createDataPartition(dados$biomassa, p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]

cat("\nRF com Hold-Out\n")

### Treino com Hold-Out
set.seed(202485)
rf <- train(biomassa~., data=treino, method="rf")
rf
### Aplicar modelos treinados na base de Teste
predict.rf <- predict(rf, teste)

cat("\n### RESULTADO DO R2 ###\n")
r2 <- function(predito, observado) {
  return(1 - (sum((predito-observado)^2) / sum((observado
mean(observado))^2)))
}
r2(predict.rf, teste$biomassa)

```



```

cat("\n### RESULTADO DO SYX ###\n")
syx <- function(predito, observado, n) {
  return( sqrt( sum((predito - observado)^2) / ( n - 2 ) ) )
}
syx(predict.rf, teste$biomassa, length(teste$biomassa))

cat("\n### RESULTADO DO PEARSON ###\n")
pearson <- function(predito, observado) {
  return (sum((predito - mean(predito)) * (observado - mean(observado))) /
    (sqrt(sum((predito - mean(predito))^2)) * sqrt(sum((observado -
    mean(observado))^2))))
}
pearson(predict.rf, teste$biomassa)

cat("\n### RESULTADO DO RMSE ###\n")
rmse <- function(predito, observado, n) {
  return( sqrt( 1/n * sum((observado - predito)^2 ) ) )
}
rmse(predict.rf, teste$biomassa, length(teste$biomassa))

cat("\n### RESULTADO DO MAE ###\n")
mae <- function(predito, observado, n) {
  return( ( 1/n * sum( (observado - predito) ) ) )
}
mae(predict.rf, teste$biomassa, length(teste$biomassa))

### PREDIÇÕES DE NOVOS CASOS
cat("\n### PREDIÇÕES DE NOVOS CASOS - RF COM HOLD-OUT ###\n")
dados_novos_casos <- read.csv("5 - Biomassa - Dados - Novos Casos.csv")
head(dados_novos_casos)

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

### GRÁFICO DE RESÍDUOS
cat("\n### GRÁFICO DE RESÍDUOS - RF COM HOLD-OUT ###\n")

residuo <- (resid(rf) * 100)
residuo

```

```

rf <- fitted(rf)

plot(rf, residuo,
     xlab = "Valores Ajustados",
     ylab = "Resíduos Percentuais",
     main = "Gráfico de Resíduos Percentuais",
     ylim = c(-100, 100),
     yaxt = "n") # Desabilitar os ticks padrão do eixo y

# Adicionar uma linha horizontal em y = 0
abline(h = 0, col = "red")
# Adicionar ticks personalizados no eixo y
axis(side = 2, at = seq(-100, 100, by = 10))

### AGRUPAMENTO ###
### Veículos ###
### Leitura dos dados
dados <- read.csv("6 - Veiculos - Dados.csv")
### Retira o atributo ID
dados$a <- NULL
head(dados)
### Executa o Kmeans
set.seed(202485)
veiculoCluster <- kmeans(dados[, 1:18], 10)
veiculoCluster
table(veiculoCluster$cluster, dados$tipo)
### Cria um arquivo com todos os registros e mais os clusters de cada um
resultado <- cbind(dados, veiculoCluster$cluster)
head(resultado, 10)

### AGRUPAMENTO ###
### Musculação ###
### Leitura dos dados
dados <- read.transactions(file="2 - Musculacao -
Dados.csv",format="basket",sep=";")
head(dados)
inspect(dados[1:3])
#summary(dados)
### Agora vamos obter as regras:
### Primeiramente definimos o Suporte=0,001 e Confiança=0,7
set.seed(202485)

```

```

rules <- apriori(dados, parameter = list(supp = 0.001, conf = 0.7,
minlen=2))
summary(rules)
### Vamos ver as 5 primeiras regras ordenadas pela confiança:
options(digits=2)
inspect(sort(rules[1:20], by="confidence"))

```

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM - CV	C = 100 Sigma = 0.015	0.8323	<pre> Prediction bus opel saab van bus 43 0 1 0 opel 0 30 14 1 saab 0 12 28 0 van 0 0 0 38 </pre>
RNA – CV	Size = 41 Decay = 0.7	0.8024	<pre> Prediction bus opel saab van bus 43 1 1 0 opel 0 27 15 1 saab 0 14 26 0 van 0 0 1 38 </pre>
RF – Hold-out	Mtry = 10	0.7725	<pre> Prediction bus opel saab van bus 42 0 4 0 opel 0 24 10 0 saab 1 17 25 1 van 0 1 4 38 </pre>
RF – CV	Mtry = 9	0.7545	<pre> Prediction bus opel saab van bus 40 0 3 0 opel 1 23 12 0 saab 1 18 25 1 van 1 1 3 38 </pre>
SVM – Hold-out	C = 1 Sigma = 0.06438	0.7425	<pre> Prediction bus opel saab van bus 41 2 1 0 opel 1 20 15 0 saab 0 19 24 0 van 1 1 3 39 </pre>
KNN	K = 1	0.6176	<pre> Prediction bus opel saab van bus 38 4 3 3 opel 3 15 11 2 saab 0 32 18 2 van 0 4 1 34 </pre>
<u>RNA – Hold-out</u>	<u>Size = 5</u> <u>Decay = 0.1</u>	<u>0.6048</u>	<pre> Prediction bus opel saab van bus 37 5 8 0 opel 0 1 1 0 saab 0 28 26 2 van 6 8 8 37 </pre>

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA - CV	Size = 11 Decay = 0.1	0.7582	<pre> Prediction neg pos neg 87 24 pos 13 29 </pre>

SVM – Hold-out	C = 0.25 Sigma = 0.1129486	0.7255	Prediction neg pos neg 88 30 pos 12 23
SVM – CV	C = 1 Sigma = 0.01	0.7255	Prediction neg pos neg 88 30 pos 12 23
RF – CV	Mtry = 2	0.7255	Prediction neg pos neg 86 28 pos 14 25
KNN	K = 9	0.7208	Prediction neg pos neg 77 27 pos 16 34
RF – Hold-out	Mtry = 2	0.7059	Prediction neg pos neg 85 30 pos 15 23
RNA – Hold-out	Size = 3 Decay = 0.1	0.6536	Prediction neg pos neg 89 42 pos 11 11

Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RNA – CV	Size = 10 Decay = 0.1	0.8330	0.0612	0.9163	0.0605	-0.0012
SVM – CV	C = 1 Sigma = 0.2074	0.8290	0.0619	0.9167	0.0613	-0.0104
RF – Hold-out	Mtry = 2	0.8266	0.0623	0.9120	0.0617	-0.0028
SVM – Hold-out	C = 0.5 Sigma = 0.2074	0.8261	0.0624	0.9190	0.0618	-0.0116
RF – CV	Mtry = 2	0.8257	0.0625	0.9113	0.0619	-0.0022
RNA – Hold-out	Size = 5 Decay = 0.1	0.8095	0.0653	0.9048	0.0647	-0.0012
KNN	K = 9	0.7536	0.0743	0.8696	0.0736	0.00169

Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	Mtry = 2	0.9840	157.7866	0.9922	155.1345	-23.3175

RNA – CV	Size = 5 Decay = 0.4	0.9821	166.7228	0.9920	164.9206	-0.7008
RF – CV	Mtry = 2	0.9768	190.1892	0.9895	186.9931	-11.9455
KNN	K = 3	0.8594	484.1457	0.9486	476.0082	31.7907
RNA – Hold-out	Size = 3 Decay = 0.1	0.7406	635.3614	0.9669	634.6823	-124.8898
SVM – CV	C = 1 Sigma = 1.0278	0.2861	1054.1573	0.5591	1036.4391	57.9005
SVM – Hold-out	C = 1 Sigma = 1.0278	0.2861	1054.1573	0.5591	1036.4391	57.9005

Regras de Associação

Musculação

Regras geradas com uma configuração de Suporte e Confiança.

```
data ntransactions support confidence
dados          26    0.001      0.7
```

```
call
apriori(data = dados, parameter = list(supp = 0.001, conf = 0.7, minlen = 2))
```

	lhs	rhs	support	confidence	coverage	lift	count
[1]	{Crucifixo}	=> {Afundo}	0.077	1.00	0.077	2.89	2
[2]	{Crucifixo}	=> {Gemeos}	0.077	1.00	0.077	1.53	2
[3]	{Crucifixo}	=> {LegPress}	0.077	1.00	0.077	1.24	2
[4]	{Adutor}	=> {Agachamento}	0.115	1.00	0.115	3.25	3
[5]	{Adutor}	=> {LegPress}	0.115	1.00	0.115	1.24	3
[6]	{Flexor}	=> {Esteira}	0.077	1.00	0.077	2.17	2
[7]	{Flexor}	=> {Extensor}	0.077	1.00	0.077	2.00	2
[8]	{Flexor}	=> {Bicicleta}	0.077	1.00	0.077	1.86	2
[9]	{Flexor}	=> {LegPress}	0.077	1.00	0.077	1.24	2
[10]	{Agachamento}	=> {LegPress}	0.308	1.00	0.308	1.24	8
[11]	{Afundo}	=> {Gemeos}	0.346	1.00	0.346	1.53	9
[12]	{Esteira}	=> {Extensor}	0.423	0.92	0.462	1.83	11
[13]	{AgachamentoSmith}	=> {Extensor}	0.346	0.90	0.385	1.80	9
[14]	{Extensor}	=> {Esteira}	0.423	0.85	0.500	1.83	11
[15]	{Esteira}	=> {Bicicleta}	0.385	0.83	0.462	1.55	10
[16]	{AgachamentoSmith}	=> {Esteira}	0.308	0.80	0.385	1.73	8
[17]	{AgachamentoSmith}	=> {Bicicleta}	0.308	0.80	0.385	1.49	8
[18]	{Afundo}	=> {LegPress}	0.269	0.78	0.346	0.96	7
[19]	{Bicicleta}	=> {Esteira}	0.385	0.71	0.538	1.55	10
[20]	{AgachamentoSmith}	=> {LegPress}	0.269	0.70	0.385	0.87	7

APÊNDICE H - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

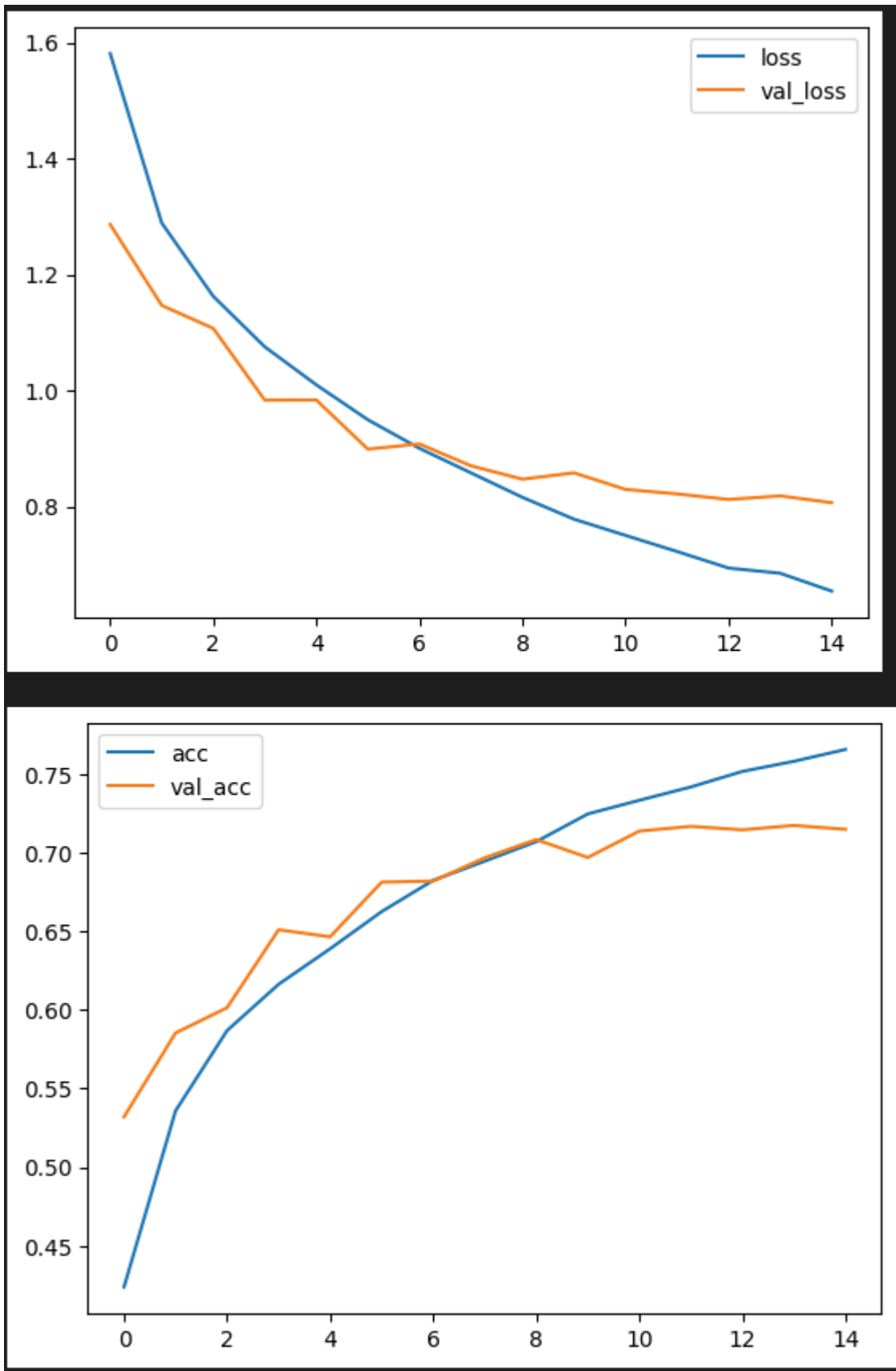
4 Tradutor de Textos (Transformer)

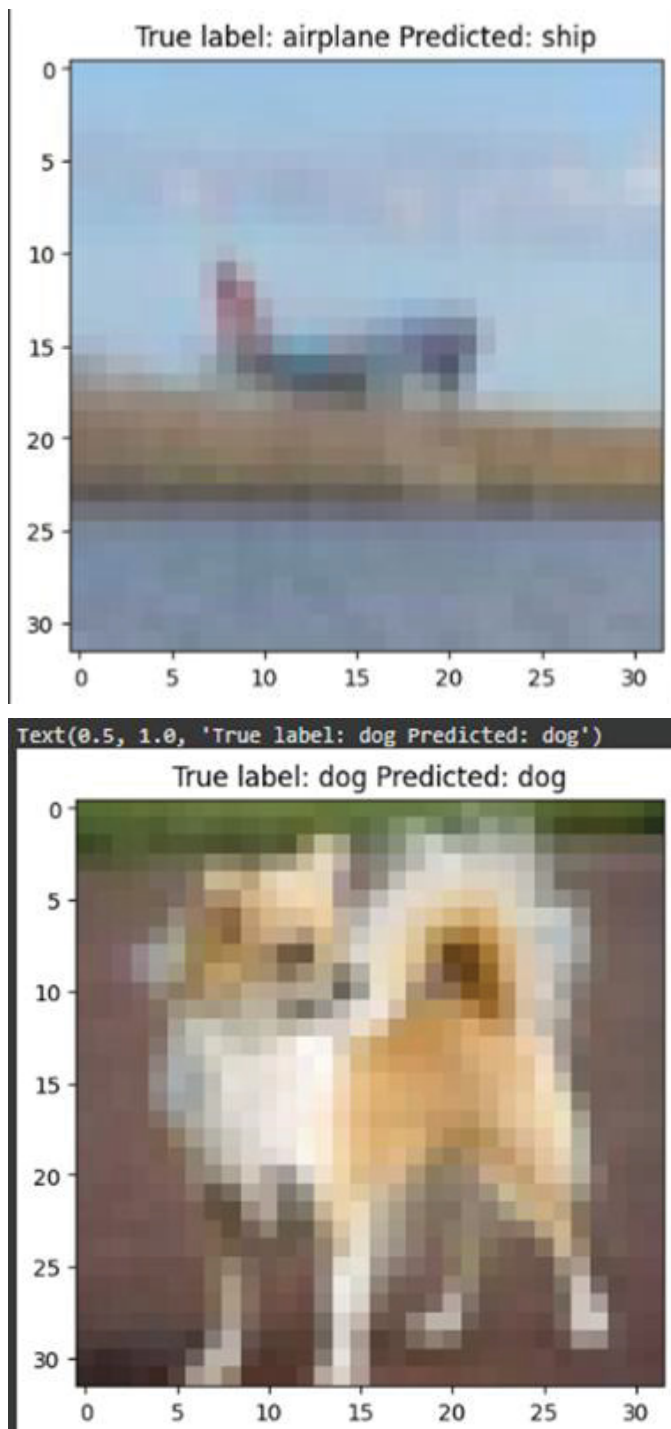
Implementar o tradutor de texto do português para o inglês, usando a base de dados e a arquitetura Transformer vista no curso.

B – RESOLUÇÃO

1 Classificação de Imagens (CNN)

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
# Carregar a base de dados (CIFAR10)
cifar10 = tf.keras.datasets.cifar10
# Separar base de treino e teste
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# Imagens em pixels de 0 - 255
# / 255.0 transforma em 0 - 1
x_train, x_test = x_train / 255.0, x_test / 255.0
# O dado y é a classe a qual faz parte
# O flatten torna os dados vetorizados
y_train, y_test = y_train.flatten(), y_test.flatten()
# Dimensão dos dados
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)
# 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()
```





2 Detector de SPAM (RNN)

```
!wget http://www.razer.net.br/datasets/spam.csv
# Realizar pré processamento
df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
```

```

df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values
x_train, x_test, y_train, y_test = train_test_split(df["data"],
y, test_size=0.33)
num_words = 20000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train) # sequencia de treino
tokenizer
sequences_test = tokenizer.texts_to_sequences(x_test) # sequencia de teste
tokenizer
word2index = tokenizer.word_index # quantidade de tokens
V = len(word2index)
print("%s tokens" % V)
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)
D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido (matrix)
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)

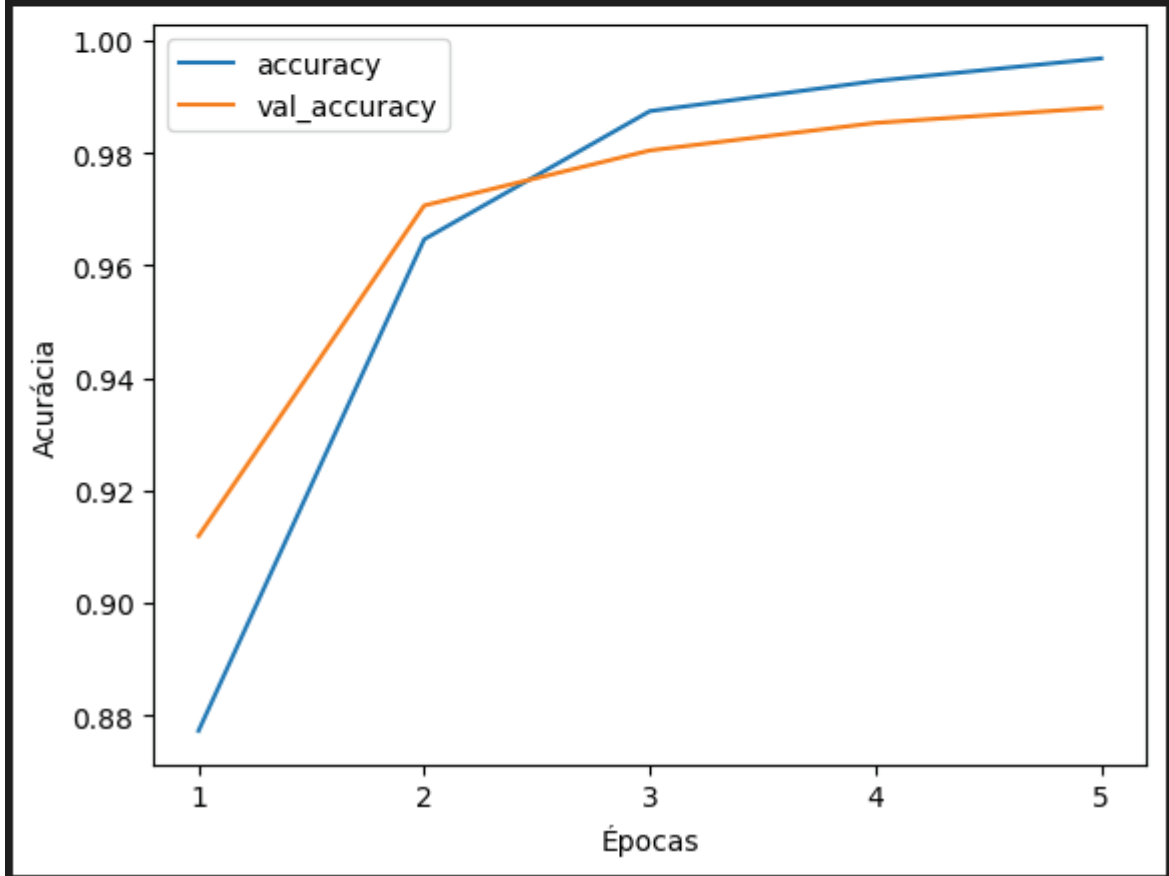
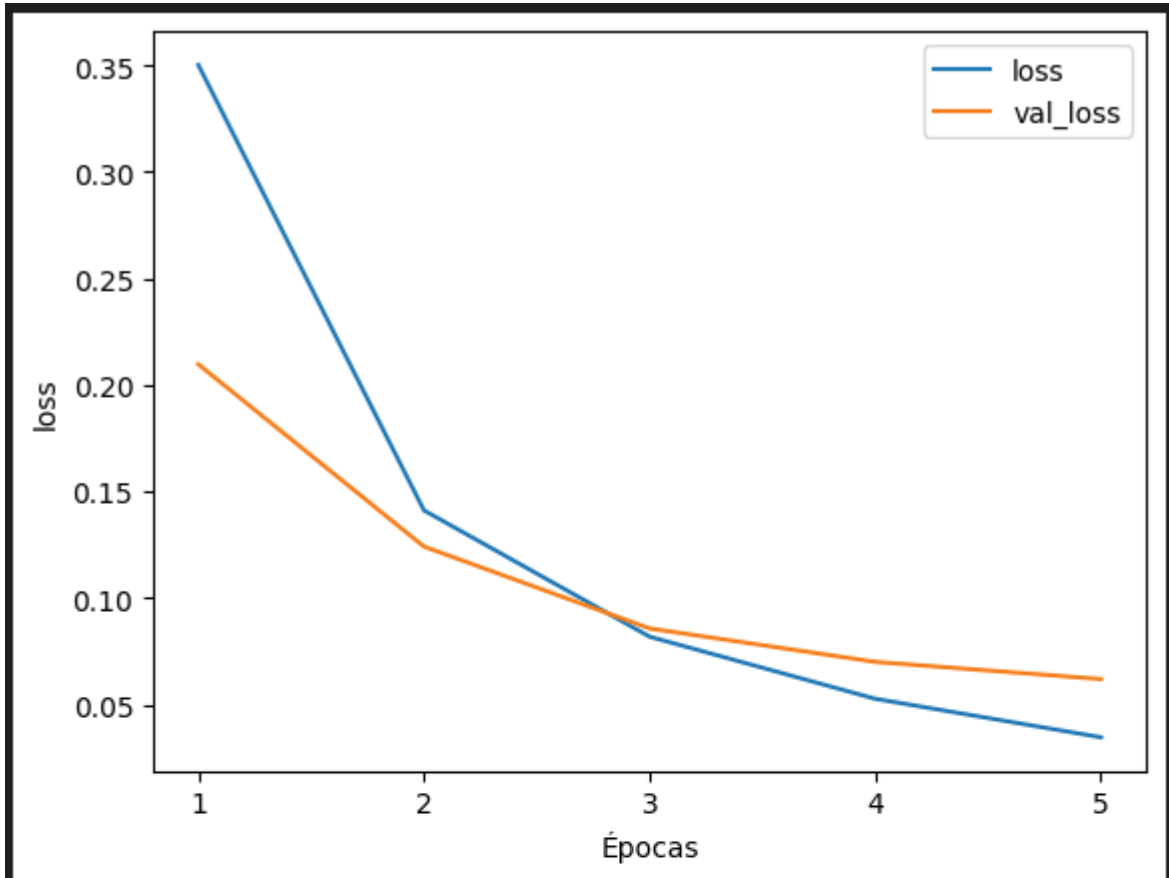
```

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 162)	0
embedding (Embedding)	(None, 162, 20)	144,620
lstm (LSTM)	(None, 5)	520
dense_2 (Dense)	(None, 1)	6

Total params: 145,146 (566.98 KB)

Trainable params: 145,146 (566.98 KB)

Non-trainable params: 0 (0.00 B)



3 Gerador de Dígitos Fake (GAN)

```

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

(train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()
# Normalização
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5 # Normaliza entre [-1, 1]
# Gera o banco em partes e randomiza
BUFFER_SIZE = 60000
BATCH_SIZE = 256

# Cria o dataset (from_tensor_slices)
# Randomiza (shuffle)
# Combina elementos consecutivos em lotes (batch)
train_dataset=tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFF
ER_SIZE).batch(BATCH_SIZE)
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

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

    # 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

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

plt.imshow(generated_image[0, :, :, 0], cmap='gray')
def make_discriminator_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

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

    return model

discriminator = make_discriminator_model()
decision = discriminator(generated_image)

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

# Perda do DISCRIMINADOR
def discriminator_loss(real_output, fake_output):

```

```

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

# Perda do GERADOR
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,
discriminator_optimizer=discriminator_optimizer, generator=generator,
discriminator=discriminator)
EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16

# 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])
def f(x, y):
    return 3*x**2 + 2*x*y

x, y = tf.Variable(5.), tf.Variable(3.)
with tf.GradientTape() as tape:
    z = f(x, y)
gradients = tape.gradient(z, [x, y])
print(gradients)
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

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

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

        gen_loss = generator_loss(fake_output)

```

```

        disc_loss = discriminator_loss(real_output, fake_output)
        gradients_of_generator=gen_tape.gradient(gen_loss,generator.trainable_variables)

        gradients_of_discriminator=disc_tape.gradient(disc_loss,
        discriminator.trainable_variables)
        generator_optimizer.apply_gradients(zip(gradients_of_generator,
        generator.trainable_variables))
        discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
        discriminator.trainable_variables))
def 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)
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()
train(train_dataset, EPOCHS)
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))
def display_image(epoch_no):
    return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))

display_image(EPOCHS)

anim_file = 'dcgan.gif'

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

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

```



4 Tradutor de Textos (Transformer)

```

import collections
import logging
import os
import pathlib
import re
import string
import sys
import time

import numpy as np
import matplotlib.pyplot as plt

import tensorflow_datasets as tfds
import tensorflow_text as text
import tensorflow as tf

print(tfds.__version__)
print(text.__version__)
print(tf.__version__)

logging.getLogger('tensorflow').setLevel(logging.ERROR) # suppress warnings
examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en',
with_info=True, as_supervised=True)

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

```

```

[item for item in dir(tokenizers.en) if not item.startswith('_')]
BUFFER_SIZE = 20000
BATCH_SIZE = 64

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

train_batches = make_batches(train_examples)
val_batches = make_batches(val_examples)
def get_angles(pos, i, d_model):
    angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model))
    return pos * angle_rates

def positional_encoding(position, d_model):
    angle_rads = get_angles(np.arange(position)[:, np.newaxis],
                             np.arange(d_model) [np.newaxis, :],
                             d_model)

    # apply sin to even indices in the array; 2i
    angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])

    # apply cos to odd indices in the array; 2i+1
    angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])

    pos_encoding = angle_rads[np.newaxis, ...]

    return tf.cast(pos_encoding, dtype=tf.float32)
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]

# Juggle the dimensions for the plot
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))

```

```
pos_encoding = tf.reshape(pos_encoding, (d, n))
```

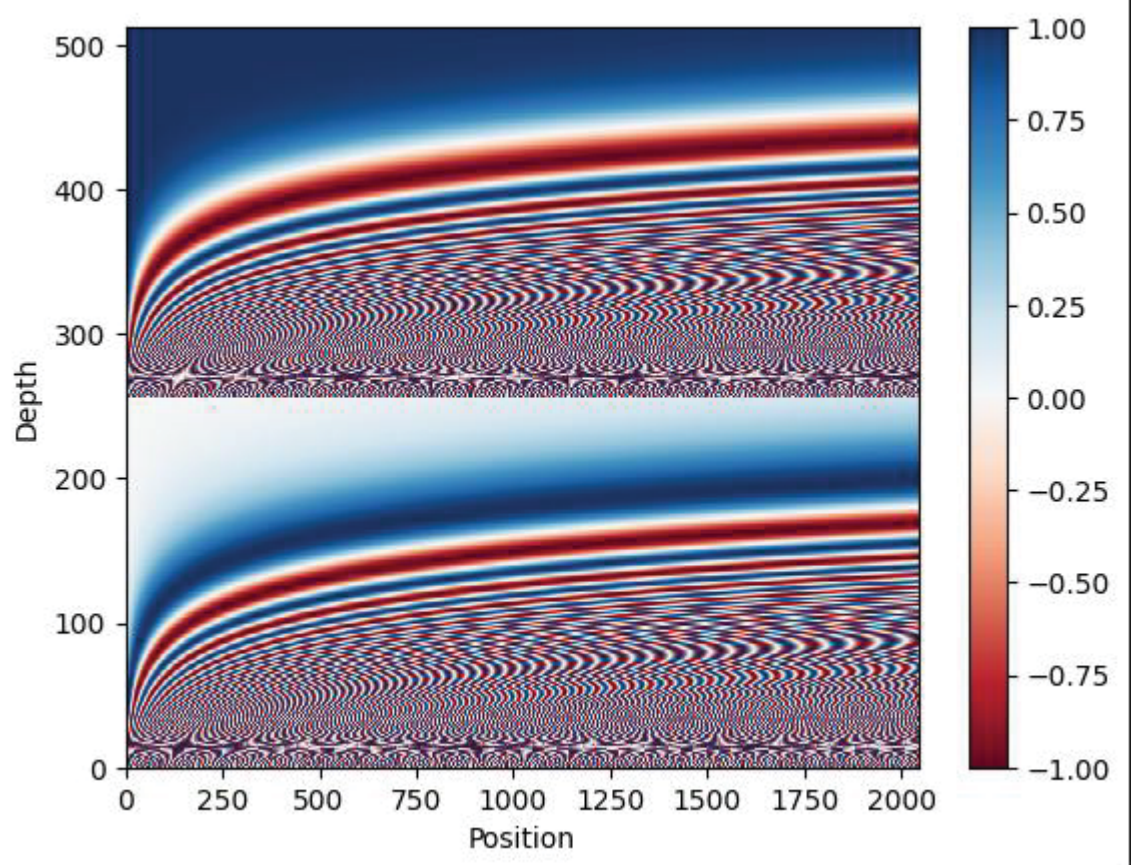
```
plt.pcolormesh(pos_encoding, cmap='RdBu')
```

```
plt.ylabel('Depth')
```

```
plt.xlabel('Position')
```

```
plt.colorbar()
```

```
plt.show()
```



```
def create_padding_mask(seq):
```

```
    seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
```

```
    # 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):
```

```
    mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
```

```
    return mask # (seq_len, seq_len)
```

```

    matmul_qk = tf.matmul(q, k, transpose_b=True) # (... , seq_len_q,
seq_len_k)

    # scale matmul_qk
    dk = tf.cast(tf.shape(k)[-1], tf.float32)
    scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)

    # add the mask to the scaled tensor.
    if mask is not None:
        scaled_attention_logits += (mask * -1e9)

    # softmax is normalized on the last axis (seq_len_k) so that the scores
    # add up to 1.
    attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1) #
(... , seq_len_q, seq_len_k)

    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):
        """Split the last dimension into (num_heads, depth).
        Transpose the result such that the shape is (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)

    # 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)

    scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3]) #
(batch_size, seq_len_q, num_heads, depth)

    concat_attention = tf.reshape(scaled_attention,
                                  (batch_size, -1, self.d_model)) #
(batch_size, seq_len_q, d_model)

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

    return output, attention_weights

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

```

```

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

        self.mha = MultiHeadAttention(d_model, num_heads)
        self.ffn = point_wise_feed_forward_network(d_model, dff)

        self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)

        self.dropout1 = tf.keras.layers.Dropout(rate)
        self.dropout2 = tf.keras.layers.Dropout(rate)

    def call(self, x, training, mask):

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

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

        return out2

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

        self.mha1 = MultiHeadAttention(d_model, num_heads)
        self.mha2 = MultiHeadAttention(d_model, num_heads)

        self.ffn = point_wise_feed_forward_network(d_model, dff)

        self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)

```



```

self.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)

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

        # 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)

```

```

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

    return final_output, attention_weights

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

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

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

    return enc_padding_mask, look_ahead_mask, dec_padding_mask

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

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

        self.d_model = d_model
        self.d_model = tf.cast(self.d_model, tf.float32)

        self.warmup_steps = warmup_steps

    def __call__(self, step):
        arg1 = tf.math.rsqrt(step)

```

```

    arg2 = step * (self.warmup_steps ** -1.5)

    return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

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

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

    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask

    return tf.reduce_sum(loss_) / tf.reduce_sum(mask)

def accuracy_function(real, pred):
    accuracies = tf.equal(real, tf.argmax(pred, axis=2))

    mask = tf.math.logical_not(tf.math.equal(real, 0))
    accuracies = tf.math.logical_and(mask, accuracies)

    accuracies = tf.cast(accuracies, dtype=tf.float32)
    mask = tf.cast(mask, dtype=tf.float32)
    return tf.reduce_sum(accuracies) / tf.reduce_sum(mask)

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

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

```

```

checkpoint_path = "./checkpoints/train"

ckpt = tf.train.Checkpoint(transformer=transformer,
                           optimizer=optimizer)

ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path,
max_to_keep=5)

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

EPOCHS = 20

# The @tf.function trace-compiler train_step into a TF graph for faster
# execution. The function specializes to the precise shape of the argument
# tensors. To avoid re-tracing due to the variable sequence lengths or
variable
# batch sizes (the last batch is smaller), use input_signature to specify
# more generic shapes.

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_states()
    train_accuracy.reset_states()

    # inp -> portuguese, tar -> english
    for (batch, (inp, tar)) in enumerate(train_batches):
        train_step(inp, tar)

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

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

            print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}')

            print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')

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

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

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

        encoder_input = sentence

```

```

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

# `tf.TensorArray` is required here (instead of a python list) so that
the
# dynamic-loop can be traced by `tf.function`.
output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
output_array = output_array.write(0, start)

for i in tf.range(max_length):
    output = tf.transpose(output_array.stack())
    predictions, _ = self.transformer([encoder_input, output],
training=False)

    # select the last token from the seq_len dimension
    predictions = predictions[:, -1:, :] # (batch_size, 1, vocab_size)

    predicted_id = tf.argmax(predictions, axis=-1)

    # concatenate the predicted_id to the output which is given to the
decoder
    # as its input.
    output_array = output_array.write(i+1, predicted_id[0])

    if predicted_id == end:
        break

output = tf.transpose(output_array.stack())
# output.shape (1, tokens)
text = tokenizers.en.detokenize(output)[0] # shape: ()

tokens = tokenizers.en.lookup(output)[0]

# `tf.function` prevents us from using the attention_weights that were
# calculated on the last iteration of the loop. So recalculate them
outside

```

```

        # the loop.
        _, attention_weights = self.transformer([encoder_input, output[:, :-1]],
        training=False)

        return text, tokens, attention_weights

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

```

Prediction : b'i read about triophons in the encyclopedia .'

APÊNDICE I - BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

Introdução

O projeto visa desenvolver um sistema de PLN que atenda às necessidades de um departamento de Recursos Humanos, fornecendo um canal de comunicação eficiente e personalizado para funcionários. O sistema será capaz de responder a perguntas frequentes, fornece orientações e esclarecer dúvidas, otimizando o trabalho dos profissionais de RH e melhorando a experiência dos colaboradores. Para alcançar esses objetivos, o projeto utilizará o MongoDB como banco de dados principal e Apache Hadoop com MapReduce, aproveitando suas capacidades para lidar com os 4Vs de Big Data: Volume, Velocidade, Variedade e Veracidade e bibliotecas de PLN.

1. MapReduce em PLN

O MapReduce é fundamental para o processamento de grandes volumes de dados no PLN. Os 4Vs de Big Data são abordados da seguinte forma:

- **Volume:** O MapReduce permite processar grandes conjuntos de dados de maneira distribuída, o que é essencial para manejar o volume de informações textuais e interações geradas.
- **Velocidade:** Facilita o processamento rápido ao distribuir tarefas em múltiplos nós, ajudando a manter a eficiência mesmo com dados de alta velocidade.
- **Variedade:** Pode lidar com diferentes tipos de dados textuais (como perguntas, respostas e documentos) e transformá-los em informações úteis.
- **Veracidade:** Assegura a precisão dos dados através de processos de verificação e análise distribuída.

2. Arquitetura do Sistema

Coleta e Preparação dos Dados

Criação de um Corpus: Reunir um conjunto abrangente de perguntas e respostas, políticas da empresa, documentos internos e informações relevantes sobre os processos de RH. O MongoDB será usado para armazenar estes dados, aproveitando sua flexibilidade para lidar com diferentes formatos e tipos de dados.

Limpeza e Pré-processamento:

- **Tokenização:** Dividir o texto em tokens (palavras ou subpalavras).
- **Remoção de Stop Words:** Remover palavras comuns que não agregam valor semântico.
- **Stemming ou Lematização:** Reduzir as palavras à sua raiz.
- **Normalização:** Converter para minúsculas, remover pontuação e caracteres especiais.
- **Criação de um Vocabulário:** Construir um vocabulário a partir do corpus, mapeando palavras frases para representações numéricas. Armazenar essas informações no MongoDB permite a recuperação e atualização eficiente dos dados.

3. Ambiente de Desenvolvimento

- **Máquinas Virtuais:** Utilizar máquinas virtuais para criar um ambiente de desenvolvimento isolado e escalável.
- **Sistema Operacional:** Distribuições Linux, como Ubuntu ou CentOS são opções populares para desenvolvimento em Python e bibliotecas PLN.
- **MongoDB:** Configurar o MongoDB para armazenar e gerenciar dados textuais e interações, abordando os desafios de Volume e Velocidade.

4. Ferramentas e Bibliotecas

- **NLTK:** Para pré-processamento de texto e tarefas básicas de PLN.
- **spaCy:** Para processamento de linguagem natural mais avançado, com modelos pré-treinados.
- **TensorFlow ou PyTorch:** Para construção e treinamento de modelos de deep learning.
- **Rasa ou Dialogflow:** Para criação de chatbots.

5. Pipeline de Processamento

Receber a pergunta do usuário;

Pré-processar a pergunta;

Vetorizar a pergunta;

Comparar a pergunta com o banco de dados de perguntas e respostas no MongoDB;

Gerar uma resposta utilizando o modelo de linguagem;

Apresentar a resposta ao usuário.

6. Conclusão

O Projeto de Processamento de Linguagem Natural (PLN) para Recursos Humanos visa modernizar e otimizar a comunicação interna por meio do uso de MongoDB e Apache Hadoop com MapReduce e PLN. O MongoDB facilita o armazenamento e a recuperação flexível de grandes volumes de dados textuais, enquanto o MapReduce garante processamento distribuído eficiente, abordando os 4Vs de Big Data: Volume, Velocidade, Variedade e Veracidade.

O sistema é projetado para processar e analisar uma variedade de dados textuais, proporcionando respostas precisas e rápidas por meio de chatbots e integração com sistemas de RH existentes. A abordagem contínua de avaliação e feedback assegura a adaptação e evolução do sistema conforme as necessidades dos usuários.

APÊNDICE J - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

- Utilize a base de dados de **Treino** já separadas em treino e validação do exercício anterior
- Treine modelos VGG16 e Resnet50 adaptadas com e sem *Data Augmentation*
- Aplique os modelos treinados nas imagens da base de **Teste**
- Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.
- Indique qual modelo dá o melhor o resultado e a métrica utilizada

B – RESOLUÇÃO

1 Extração de Características

```
# 1. Carregue a base de dados de Treino.
# extrair os dados de treino do arquivo Train_Warwick.zip
!rm -rfv /content/Train_4cls_amostra
!unzip Train_Warwick.zip
Show hidden output
# 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.
# 2. Crie partições contendo 80% para treino e 20% para validação (atenção
aos pacientes).
import os
import shutil
!rm -rf /content/Train_4cls_test /content/Train_4cls_train
# Defina o caminho para os diretórios
base_dir = 'Train_4cls_amostra/'
train_dir = 'Train_4cls_train/'
test_dir = 'Train_4cls_test/'
# Crie os diretórios de treino e teste
os.makedirs(train_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)
# Coleta de IDs únicos de pacientes
pacientes = set()
for classe in os.listdir(base_dir):
    if os.path.isdir(os.path.join(base_dir, classe)):
```

```

        for imagem in os.listdir(os.path.join(base_dir, classe)):
            paciente_id = imagem.split('_')[0]
            pacientes.add(paciente_id)

# Ordena os pacientes
pacientes = sorted(pacientes)

# Divide os pacientes em treino (80%) e teste (20%)
num_treino = int(0.8 * len(pacientes))
pacientes_treino = pacientes[:num_treino]
pacientes_teste = pacientes[num_treino:]

# Move as imagens para os diretórios de treino e teste
for classe in os.listdir(base_dir):
    if os.path.isdir(os.path.join(base_dir, classe)):
        # Crie o diretório da classe no conjunto de treino e teste
        os.makedirs(os.path.join(train_dir, classe), exist_ok=True)
        os.makedirs(os.path.join(test_dir, classe), exist_ok=True)
        for imagem in os.listdir(os.path.join(base_dir, classe)):
            paciente_id = imagem.split('_')[0]
            if paciente_id in pacientes_treino:
                shutil.copy(os.path.join(base_dir, classe, imagem),
os.path.join(train_dir, classe, imagem))
            elif paciente_id in pacientes_teste:
                shutil.copy(os.path.join(base_dir, classe, imagem),
os.path.join(test_dir, classe, imagem))

# Contagem total de imagens
total_train = sum(len(files) for _, _, files in os.walk(train_dir))
total_test = sum(len(files) for _, _, files in os.walk(test_dir))
print(f'Total de imagens: {total_train+total_test}')
print(f'Total de imagens no conjunto de treino: {total_train}')
print(f'Total de imagens no conjunto de teste: {total_test}')

# LBP #

#####
#####

import os
import cv2
import numpy as np
import pandas as pd
from skimage import feature

# Função para calcular LBP
def extract_lbp(image):
    # Converte a imagem para escala de cinza

```

```

    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Calcula LBP
    lbp = feature.local_binary_pattern(gray_image, P=8, R=1,
method='uniform')
    # Calcula o histograma do LBP
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, 11), range=(0,
10))
    # Normaliza o histograma
    hist = hist.astype("float")
    hist /= hist.sum()
    return hist
# Diretório das imagens
base_dir = 'Train_4cls_train/'
data = []
# Percorre as classes (subdiretórios)
for classe in os.listdir(base_dir):
    class_path = os.path.join(base_dir, classe)
    if os.path.isdir(class_path):
        # Percorre as imagens na classe
        for imagem in os.listdir(class_path):
            image_path = os.path.join(class_path, imagem)
            # Carrega a imagem
            image = cv2.imread(image_path)
            # Extraí LBP
            lbp_features = extract_lbp(image)
            # Adiciona os recursos, a classe (número do subdiretório) e o
nome da imagem
            data.append((*lbp_features, classe, imagem))
# Cria DataFrame
columns = [f'LBP_{i}' for i in range(len(data[0]) - 2)] + ['Class',
'Image_Name']
features_df = pd.DataFrame(data, columns=columns)
# Salva em CSV
features_df.to_csv('lbp_features_train.csv', index=False)
print("Características extraídas e salvas em 'lbp_features_train.csv'")

#####
#####

# VGG16 #

```

```
#####
#####

import os
import cv2
import numpy as np
import pandas as pd
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
# Função para carregar e pré-processar imagens
def load_and_preprocess_images(base_dir):
    images = []
    labels = []
    for classe in os.listdir(base_dir):
        class_path = os.path.join(base_dir, classe)
        if os.path.isdir(class_path):
            for imagem in os.listdir(class_path):
                image_path = os.path.join(class_path, imagem)
                image = cv2.imread(image_path)
                image = cv2.resize(image, (224, 224)) # VGG16 espera imagens
224x224
                images.append(image)
                labels.append(classe) # Adiciona a classe correspondente
    return np.array(images), np.array(labels)
# Carrega as imagens da base Train_4cls_train
base_dir = 'Train_4cls_train/'
images, labels = load_and_preprocess_images(base_dir)
# Carrega a VGG16 pré-treinada
model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224,
3))
# Extrai características
def extract_features(model, images):
    features = model.predict(preprocess_input(images))
    return features.reshape(features.shape[0], -1) # Flatten
features = extract_features(model, images)
# Cria um DataFrame com as características e as classes
features_df = pd.DataFrame(features)
features_df['Class'] = labels
# Salva em CSV
features_df.to_csv('vgg16_features_train.csv', index=False)
print("Características extraídas e salvas em 'vgg16_features_train.csv'")
```

```

# 4. Treine modelos Random Forest, SVM e RNA para predição dos dados
extraídos.

# LBP

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from tensorflow import keras
from tensorflow.keras import layers

# Carrega os dados
features_df = pd.read_csv('lbp_features_train.csv')
#print(features_df.head())

# Remove a coluna Image_Name
features_df = features_df.drop(columns=['Image_Name'])
#print(features_df.head())

# Separa as características e as classes
X = features_df.drop(columns=['Class']).values
y = features_df['Class'].values

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

print("#####")
print("#####")
print("TREINO LBP")

print("#####")
print("#####")

# 1. Treinando Random Forest
rf_model_lbp = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_lbp.fit(X_train, y_train)
rf_predictions_lbp = rf_model_lbp.predict(X_test)
print("LBP - Random Forest Classification Report:")
print(classification_report(y_test, rf_predictions_lbp))

# 2. Treinando SVM
svm_model_lbp = SVC(kernel='linear', random_state=42)
svm_model_lbp.fit(X_train, y_train)
svm_predictions_lbp = svm_model_lbp.predict(X_test)
print("LBP - SVM Classification Report:")
print(classification_report(y_test, svm_predictions_lbp))

```



```

# 3. Treinando Redes Neurais Artificiais (RNA)
rna_model_lbp = keras.Sequential([
    layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(64, activation='relu'),
    layers.Dense(len(set(y)), activation='softmax') # Saída para classes
])
rna_model_lbp.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
# Treina o modelo
rna_model_lbp.fit(X_train, y_train, epochs=10, batch_size=32,
validation_split=0.2)
# Avalia o modelo
rna_loss, rna_accuracy = rna_model_lbp.evaluate(X_test, y_test)
print(f"LBP - RNA Test Accuracy: {rna_accuracy:.2f}")

print("#####
#####")
print("TREINO VGG16")

print("#####
#####")
# VGG16
import pandas as pd
# Carregar os dados das características
data = pd.read_csv('vgg16_features_train.csv')
#print(data.head())
# Remove a coluna Image_Name
#data = data.drop(columns=['Image_Name'])
# Separa as características e as classes
X = data.drop(columns=['Class']).values
y = data['Class'].values
from sklearn.model_selection import train_test_split
# Divide os dados em 80% treino e 20% teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
stratify=y, random_state=42)
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
# Inicializa e treina o modelo Random Forest
rf_model_vgg16 = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_vgg16.fit(X_train, y_train)
# Faz previsões

```

```

rf_predictions_vgg16 = rf_model_vgg16.predict(X_test)
# Relatório de classificação
print("VGG16 - Random Forest Classification Report:")
print(classification_report(y_test, rf_predictions_vgg16))
from sklearn.svm import SVC
# Inicializa e treina o modelo SVM
svm_model_vgg16 = SVC(kernel='linear', random_state=42)
svm_model_vgg16.fit(X_train, y_train)
# Faz previsões
svm_predictions_vgg16 = svm_model_vgg16.predict(X_test)
# Relatório de classificação
print("VGG16 - SVM Classification Report:")
print(classification_report(y_test, svm_predictions_vgg16))
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical
# Codifica as classes em formato one-hot
num_classes = len(set(y))
y_train_encoded = to_categorical(y_train, num_classes)
y_test_encoded = to_categorical(y_test, num_classes)
# Inicializa o modelo da RNA
rna_model_vgg16 = Sequential()
rna_model_vgg16.add(Dense(256, activation='relu',
input_shape=(X_train.shape[1],)))
rna_model_vgg16.add(Dense(128, activation='relu'))
rna_model_vgg16.add(Dense(num_classes, activation='softmax'))
# Compila o modelo
rna_model_vgg16.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
# Treina o modelo
rna_model_vgg16.fit(X_train, y_train_encoded, epochs=50, batch_size=32,
validation_split=0.2)
# Faz previsões
rna_predictions_vgg16 = rna_model_vgg16.predict(X_test)
rna_predictions_classes_vgg16 = np.argmax(rna_predictions_vgg16, axis=1)
# Relatório de classificação
print("VGG16 - RNA Classification Report:")
print(classification_report(y_test, rna_predictions_classes_vgg16))

# extrair os dados de teste do arquivo Test_Warwick.zip
#!rm -rfv /content/Test_4cl_amostra

```

```

!unzip Test_Warwick.zip
# 5. Carregue a base de Teste e execute a tarefa 3 nesta base.
import os
import cv2
import numpy as np
import pandas as pd
from skimage import feature

#####
#####

# LBP #

#####
#####

# Função para calcular LBP
def extract_lbp(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    lbp = feature.local_binary_pattern(gray_image, P=8, R=1,
method='uniform')
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, 11), range=(0,
10))
    hist = hist.astype("float")
    hist /= hist.sum()
    return hist

# Diretório da base de teste
test_base_dir = '/content/Test_4cl_amostra'
test_data = []
# Percorre as classes (subdiretórios)
for classe in os.listdir(test_base_dir):
    class_path = os.path.join(test_base_dir, classe)
    if os.path.isdir(class_path):
        for imagem in os.listdir(class_path):
            image_path = os.path.join(class_path, imagem)
            image = cv2.imread(image_path)
            lbp_features = extract_lbp(image)
            test_data.append((*lbp_features, classe, imagem))

# Cria DataFrame
columns = [f'LBP_{i}' for i in range(len(test_data[0]) - 2)] + ['Class',
'Image_Name']
test_features_df = pd.DataFrame(test_data, columns=columns)
# Salva em CSV

```

```

test_features_df.to_csv('lbp_features_test.csv', index=False)
print("Características da base de teste extraídas e salvas em
'lbp_features_test.csv'")

#####
#####
# VGG16 #

#####
#####

import os
import cv2
import numpy as np
import pandas as pd
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
# Função para carregar e pré-processar imagens de teste
def load_and_preprocess_test_images(base_dir):
    images = []
    labels = []
    image_names = []
    for classe in os.listdir(base_dir):
        class_path = os.path.join(base_dir, classe)
        if os.path.isdir(class_path):
            for imagem in os.listdir(class_path):
                image_path = os.path.join(class_path, imagem)
                image = cv2.imread(image_path)
                image = cv2.resize(image, (224, 224)) # VGG16 espera imagens
224x224
                images.append(image)
                labels.append(classe) # Adiciona a classe correspondente
                image_names.append(imagem) # Armazena o nome da imagem
    return np.array(images), np.array(labels), image_names
# Carrega as imagens da base de teste
base_dir = 'Test_4cl_amostra/'
test_images, test_labels, test_image_names =
load_and_preprocess_test_images(base_dir)
# Carrega a VGG16 pré-treinada (se já não estiver carregada)
model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224,
3))
# Extraí características das imagens de teste

```

```

def extract_features(model, images):
    features = model.predict(preprocess_input(images))
    return features.reshape(features.shape[0], -1) # Flatten
test_features_vgg16 = extract_features(model, test_images)
# Cria um DataFrame com as características e os nomes das imagens
test_features_df = pd.DataFrame(test_features_vgg16)
test_features_df['Class'] = test_labels # Adiciona a classe correspondente
test_features_df['Image_Name'] = test_image_names
# Salva em CSV
test_features_df.to_csv('vgg16_features_test.csv', index=False)
print("Características do conjunto de teste extraídas e salvas em
'vgg16_features_test.csv'")
# 6. Aplique os modelos treinados nos dados de teste.
import pandas as pd
from sklearn.metrics import classification_report

print("#####
#####")
print("TREINO LBP")

print("#####
#####")
# Carregar os dados da base de teste
test_features_df = pd.read_csv('lbp_features_test.csv')
# Remove a coluna Image_Name
test_features_df = test_features_df.drop(columns=['Image_Name'])
# Separa as características e as classes
X_test = test_features_df.drop(columns=['Class']).values
y_test = test_features_df['Class'].values
print(test_features_df.head())
# 1. Previsões com Random Forest
rf_predictions_lbp = rf_model_lbp.predict(X_test)
print("LBP - Random Forest Classification Report:")
print(classification_report(y_test, rf_predictions_lbp))
# 2. Previsões com SVM
svm_predictions_lbp = svm_model_lbp.predict(X_test)
print("LBP - SVM Classification Report:")
print(classification_report(y_test, svm_predictions_lbp))
# 3. Previsões com RNA
rna_predictions_lbp = np.argmax(rna_model_lbp.predict(X_test), axis=1)
print("LBP - RNA Classification Report:")

```

```

print(classification_report(y_test, rna_predictions_lbp))

print("#####
#####")
print("TREINO VGG16")

print("#####
#####")
# Carregar os dados da base de teste
test_features_df = pd.read_csv('vgg16_features_test.csv')
# Remove a coluna Image_Name
test_features_df = test_features_df.drop(columns=['Image_Name'])
# Separa as características e as classes
X_test = test_features_df.drop(columns=['Class']).values
y_test = test_features_df['Class'].values
# 1. Previsões com Random Forest
rf_predictions_vgg16 = rf_model_vgg16.predict(X_test)
print("VGG16 - Random Forest Classification Report:")
print(classification_report(y_test, rf_predictions_vgg16))
# 2. Previsões com SVM
svm_predictions_vgg16 = svm_model_vgg16.predict(X_test)
print("VGG16 - SVM Classification Report:")
print(classification_report(y_test, svm_predictions_vgg16))
# 3. Previsões com RNA
rna_predictions_vgg16 = np.argmax(rna_model_vgg16.predict(X_test), axis=1)
print("VGG16 - RNA Classification Report:")
print(classification_report(y_test, rna_predictions_vgg16))
# 7. Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base
em suas matrizes de confusão.
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
# Função para plotar a matriz de confusão
def plot_confusion_matrix(y_true, y_pred, classes, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,
yticklabels=classes)
    plt.title(f'LBP - Matriz de Confusão - {model_name}')
    plt.xlabel('Classe Prevista')
    plt.ylabel('Classe Real')

```

```

plt.show()

# 1. Matriz de Confusão para Random Forest
plot_confusion_matrix(y_test, rf_predictions_lbp,
classes=np.unique(y_test), model_name='Random Forest')

# 2. Matriz de Confusão para SVM
plot_confusion_matrix(y_test, svm_predictions_lbp,
classes=np.unique(y_test), model_name='SVM')

# 3. Matriz de Confusão para RNA
plot_confusion_matrix(y_test, rna_predictions_lbp,
classes=np.unique(y_test), model_name='RNA')

# 7. Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base
em suas matrizes de confusão.
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Função para plotar a matriz de confusão
def plot_confusion_matrix(y_true, y_pred, classes, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,
yticklabels=classes)
    plt.title(f'VGG16 - Matriz de Confusão - {model_name}')
    plt.xlabel('Classe Prevista')
    plt.ylabel('Classe Real')
    plt.show()

# 1. Matriz de Confusão para Random Forest
plot_confusion_matrix(y_test, rf_predictions_vgg16,
classes=np.unique(y_test), model_name='Random Forest')

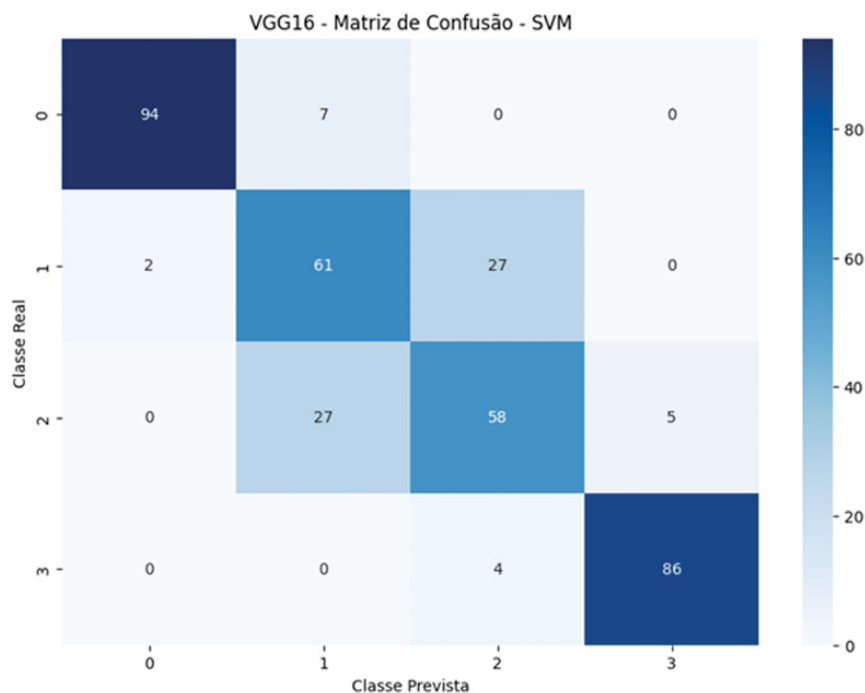
# 2. Matriz de Confusão para SVM
plot_confusion_matrix(y_test, svm_predictions_vgg16,
classes=np.unique(y_test), model_name='SVM')

# 3. Matriz de Confusão para RNA
plot_confusion_matrix(y_test, rna_predictions_vgg16,
classes=np.unique(y_test), model_name='RNA')

```

O modelo VGG16 com SVM teve a melhor acurácia.

VGG16 - SVM Classification Report:					
		precision	recall	f1-score	support
	0	0.98	0.93	0.95	101
	1	0.64	0.68	0.66	90
	2	0.65	0.64	0.65	90
	3	0.95	0.96	0.95	90
	accuracy			0.81	371
	macro avg	0.80	0.80	0.80	371
	weighted avg	0.81	0.81	0.81	371



2 Redes Neurais

```

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

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

BATCH_SIZE = 32

```



```

# Gerador com Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
valid_datagen = ImageDataGenerator(rescale=1./255)
# Gerador de treino com Data Augmentation
traingen = train_datagen.flow_from_directory(
    '/content/Train_4cls_train',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    seed=42
)
# Gerador de validação
validgen = valid_datagen.flow_from_directory(
    '/content/Train_4cls_train',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    seed=42
)
# Gerador sem Data Augmentation
train_datagen_no_aug = ImageDataGenerator(rescale=1./255)
traingen_no_aug = train_datagen_no_aug.flow_from_directory(
    '/content/Train_4cls_train',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    seed=42
)
# 2. Treine modelos VGG16 e Resnet50 adaptadas com e sem Data Augmentation

```

```

print('# Treinar VGG16 com Data Augmentation')
# Carregar o modelo VGG16 com pesos pré-treinados
vgg_model = Sequential()
vgg_model.add(VGG16(weights='imagenet',                      include_top=False,
input_shape=(224, 224, 3)))
vgg_model.add(Flatten())
vgg_model.add(Dense(256, activation='relu'))
vgg_model.add(Dense(4, activation='softmax')) # 4 classes
# Congelar as camadas do VGG16
for layer in vgg_model.layers[:-2]: # Congelar todas as camadas exceto as
últimas
    layer.trainable = False
# Compilar o modelo
vgg_model.compile(optimizer=Adam(),      loss='categorical_crossentropy',
metrics=['accuracy'])
# Treinar VGG16 com Data Augmentation
vgg_history_aug      =      vgg_model.fit(traingen,      validation_data=validgen,
epochs=10)
# Treinar VGG16 sem Data Augmentation
print('# Treinar VGG16 sem Data Augmentation')
vgg_model_no_aug = Sequential()
vgg_model_no_aug.add(VGG16(weights='imagenet',                      include_top=False,
input_shape=(224, 224, 3)))
vgg_model_no_aug.add(Flatten())
vgg_model_no_aug.add(Dense(256, activation='relu'))
vgg_model_no_aug.add(Dense(4, activation='softmax'))
# Congelar as camadas do VGG16
for layer in vgg_model_no_aug.layers[:-2]:
    layer.trainable = False
# Compilar o modelo
vgg_model_no_aug.compile(optimizer=Adam(),
loss='categorical_crossentropy', metrics=['accuracy'])
vgg_history_no_aug      =      vgg_model_no_aug.fit(traingen_no_aug,
validation_data=validgen, epochs=10)
# Carregar o modelo ResNet50 com pesos pré-treinados
print('# Treinar ResNet50 com Data Augmentation')
resnet_model = Sequential()
resnet_model.add(ResNet50(weights='imagenet',                      include_top=False,
input_shape=(224, 224, 3)))
resnet_model.add(Flatten())
resnet_model.add(Dense(256, activation='relu'))

```

```

resnet_model.add(Dense(4, activation='softmax')) # 4 classes
# Congelar as camadas do ResNet50
for layer in resnet_model.layers[:-2]:
    layer.trainable = False
# Compilar o modelo
resnet_model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
# Treinar ResNet50 com Data Augmentation
resnet_history_aug = resnet_model.fit(training_data, validation_data=validation_data,
epochs=10)
# Treinar ResNet50 sem Data Augmentation
print('# Treinar ResNet50 sem Data Augmentation')
resnet_model_no_aug = Sequential()
resnet_model_no_aug.add(ResNet50(weights='imagenet', include_top=False,
input_shape=(224, 224, 3)))
resnet_model_no_aug.add(Flatten())
resnet_model_no_aug.add(Dense(256, activation='relu'))
resnet_model_no_aug.add(Dense(4, activation='softmax'))
# Congelar as camadas do ResNet50
for layer in resnet_model_no_aug.layers[:-2]:
    layer.trainable = False
# Compilar o modelo
resnet_model_no_aug.compile(optimizer=Adam(),
loss='categorical_crossentropy', metrics=['accuracy'])
resnet_history_no_aug = resnet_model_no_aug.fit(training_data_no_aug,
validation_data=validation_data, epochs=10)
# 3. Aplique os modelos treinados nas imagens da base de Teste
# Gerador de Teste
testgen = ImageDataGenerator(rescale=1./255).flow_from_directory(
'/content/Train_4cls_test',
target_size=(224, 224),
batch_size=BATCH_SIZE,
class_mode=None, # Não temos os rótulos aqui
shuffle=False
)
# Previsões com VGG16
print('# Previsões com VGG16')
vgg_predictions_aug = vgg_model.predict(testgen)
vgg_pred_classes_aug = np.argmax(vgg_predictions_aug, axis=1)
# Previsões com VGG16 sem Data Augmentation
print('# Previsões com VGG16 sem Data Augmentation')

```

```

testgen_no_aug = ImageDataGenerator(rescale=1./255).flow_from_directory(
    '/content/Train_4cls_test',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode=None,
    shuffle=False
)
vgg_predictions_no_aug = vgg_model_no_aug.predict(testgen_no_aug)
vgg_pred_classes_no_aug = np.argmax(vgg_predictions_no_aug, axis=1)
# Previsões com ResNet50
print('# Previsões com ResNet50')
resnet_predictions_aug = resnet_model.predict(testgen)
resnet_pred_classes_aug = np.argmax(resnet_predictions_aug, axis=1)
# Previsões com ResNet50 sem Data Augmentation
print('# Previsões com ResNet50 sem Data Augmentation')
resnet_predictions_no_aug = resnet_model_no_aug.predict(testgen_no_aug)
resnet_pred_classes_no_aug = np.argmax(resnet_predictions_no_aug, axis=1)
# 4. Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base
em suas matrizes de confusão
# Supondo que você tenha os rótulos verdadeiros
true_labels = testgen.classes # Rótulos verdadeiros do gerador de teste
# Calcular matrizes de confusão
cm_vgg_aug = confusion_matrix(true_labels, vgg_pred_classes_aug)
cm_vgg_no_aug = confusion_matrix(true_labels, vgg_pred_classes_no_aug)
cm_resnet_aug = confusion_matrix(true_labels, resnet_pred_classes_aug)
cm_resnet_no_aug = confusion_matrix(true_labels,
resnet_pred_classes_no_aug)
# Função para calcular as métricas
def calculate_metrics(cm):
    TP = np.diag(cm)
    FP = cm.sum(axis=0) - TP
    FN = cm.sum(axis=1) - TP
    TN = cm.sum() - (FP + FN + TP)
    sensitivity = TP / (TP + FN) # Recall
    specificity = TN / (TN + FP)
    precision = TP / (TP + FP)
    f1_score = 2 * (precision * sensitivity) / (precision + sensitivity)
    return sensitivity, specificity, f1_score
# Calcular métricas para VGG16
sensitivity_vgg_aug, specificity_vgg_aug, f1_vgg_aug =
calculate_metrics(cm_vgg_aug)

```

```

    sensitivity_vgg_no_aug,    specificity_vgg_no_aug,    f1_vgg_no_aug    =
calculate_metrics(cm_vgg_no_aug)
    # Calcular métricas para ResNet50
    sensitivity_resnet_aug,    specificity_resnet_aug,    f1_resnet_aug    =
calculate_metrics(cm_resnet_aug)
    sensitivity_resnet_no_aug,    specificity_resnet_no_aug,    f1_resnet_no_aug    =
calculate_metrics(cm_resnet_no_aug)
    # Comparar resultados VGG16
    print("Resultados VGG16 com Data Augmentation:")
    print("Sensibilidade:", sensitivity_vgg_aug)
    print("Especificidade:", specificity_vgg_aug)
    print("F1-Score:", f1_vgg_aug)
    print("\nResultados VGG16 sem Data Augmentation:")
    print("Sensibilidade:", sensitivity_vgg_no_aug)
    print("Especificidade:", specificity_vgg_no_aug)
    print("F1-Score:", f1_vgg_no_aug)
    # Comparar resultados ResNet50
    print("\nResultados ResNet50 com Data Augmentation:")
    print("Sensibilidade:", sensitivity_resnet_aug)
    print("Especificidade:", specificity_resnet_aug)
    print("F1-Score:", f1_resnet_aug)
    print("\nResultados ResNet50 sem Data Augmentation:")
    print("Sensibilidade:", sensitivity_resnet_no_aug)
    print("Especificidade:", specificity_resnet_no_aug)
    print("F1-Score:", f1_resnet_no_aug)
    # Função para plotar a matriz de confusão
    def plot_confusion_matrix(cm, classes, title='Matriz de Confusão',
cmap=plt.cm.Blues):
    plt.figure(figsize=(10, 7))
    sns.heatmap(cm, annot=True, fmt='d', cmap=cmap,
xticklabels=classes, yticklabels=classes)
    plt.title(title)
    plt.xlabel('Predito')
    plt.ylabel('Verdadeiro')
    plt.show()
    # Classes (supondo que você tenha 4 classes)
    class_names = ['Classe 1', 'Classe 2', 'Classe 3', 'Classe 4']
    # Plotar matrizes de confusão para os 4 modelos
    plot_confusion_matrix(cm_vgg_aug, class_names, title='Matriz de Confusão
VGG16 com Data Augmentation')

```

```

plot_confusion_matrix(cm_vgg_no_aug, class_names, title='Matriz de Confusão
VGG16 sem Data Augmentation')
plot_confusion_matrix(cm_resnet_aug, class_names, title='Matriz de Confusão
ResNet50 com Data Augmentation')
plot_confusion_matrix(cm_resnet_no_aug, class_names, title='Matriz de
Confusão ResNet50 sem Data Augmentation')

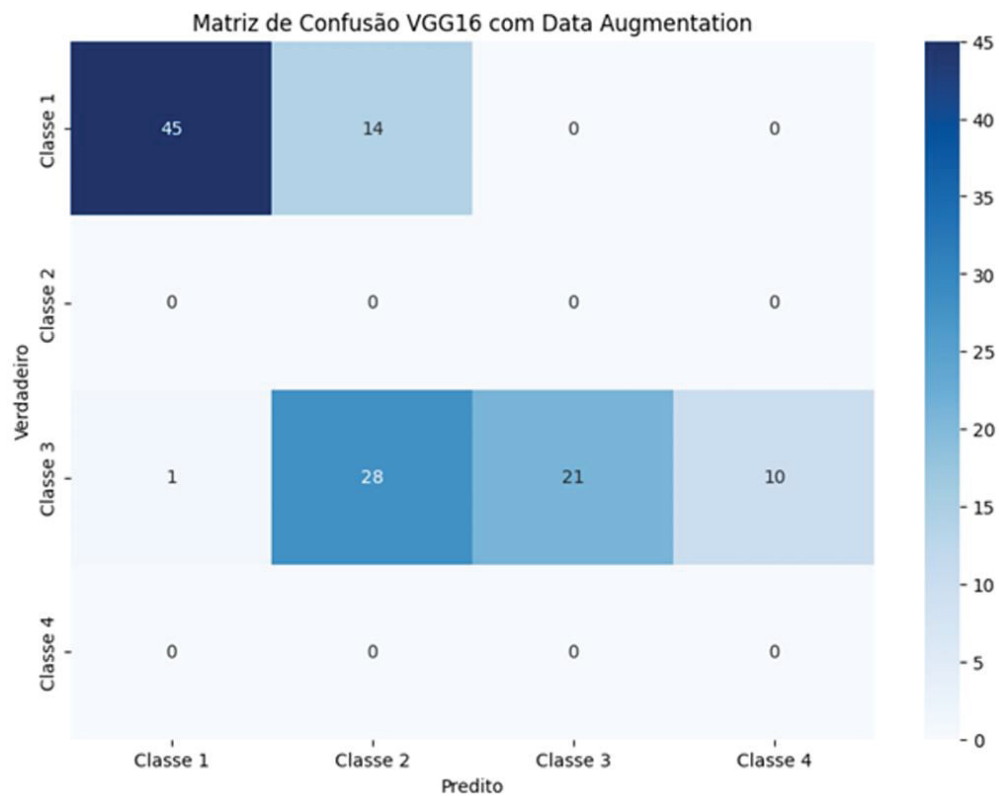
```

O modelo VGG16 com Data Augmentation teve a melhor acurácia.

```

Acurácia VGG16 com Data Augmentation: 0.5546218487394958
Acurácia VGG16 sem Data Augmentation: 0.48739495798319327
Acurácia ResNet50 com Data Augmentation: 0.3277310924369748
Acurácia ResNet50 sem Data Augmentation: 0.2689075630252101

```



APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

Parâmetros para elaboração do Trabalho:

1. Relevância Ética: O trabalho deve abordar questões éticas significativas relacionadas ao uso da inteligência artificial, especialmente no contexto do ChatGPT. Os alunos devem identificar dilemas éticos relevantes e explorar como esses dilemas afetam diferentes partes interessadas, como usuários, desenvolvedores e a sociedade em geral.

2. Análise Crítica: Os alunos devem realizar uma análise crítica das implicações éticas do uso do ChatGPT em estudos de caso específicos. Eles devem examinar como o algoritmo pode influenciar a disseminação de informações, a privacidade dos usuários e a tomada de decisões éticas. Além disso, devem considerar possíveis vieses algorítmicos, discriminação e questões de responsabilidade.

3. Soluções Responsáveis: Além de identificar os desafios éticos, os alunos devem propor soluções responsáveis e éticas para lidar com esses dilemas. Isso pode incluir sugestões para políticas, regulamentações ou práticas de design que promovam o uso responsável da inteligência artificial. Eles devem considerar como essas soluções podem equilibrar os interesses de diferentes partes interessadas e promover valores éticos fundamentais, como transparência, justiça e privacidade.

4. Colaboração e Discussão: O trabalho deve envolver discussões em grupo e colaboração entre os alunos. Eles devem compartilhar ideias, debater diferentes pontos de vista e chegar a conclusões informadas através do diálogo e da reflexão mútua. O estudo de caso do ChatGPT pode servir como um ponto de partida para essas discussões, incentivando os alunos a aplicar conceitos éticos e legais aprendidos ao analisar um caso concreto.

5. Limite de Palavras: O trabalho terá um limite de 6 a 10 páginas teria aproximadamente entre 1500 e 3000 palavras.

6. Estruturação Adequada: O trabalho siga uma estrutura adequada, incluindo introdução, desenvolvimento e conclusão. Cada seção deve ocupar uma parte proporcional do total de páginas, com a introdução e a conclusão ocupando menos espaço do que o desenvolvimento.

7. Controle de Informações: Evitar incluir informações desnecessárias que possam aumentar o comprimento do trabalho sem contribuir significativamente para o conteúdo. Concentre-se em informações relevantes, argumentos sólidos e evidências importantes para apoiar sua análise.

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

9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: <https://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx>

B – RESOLUÇÃO

RESUMO

Este trabalho investiga as implicações éticas do uso do ChatGPT em diferentes contextos, abordando questões críticas como a privacidade, a disseminação de desinformação, a dependência tecnológica e o impacto no emprego. Essas questões éticas afetam usuários, desenvolvedores e a sociedade em geral, exigindo análises profundas para compreender suas consequências. A análise crítica revela riscos significativos como violações da privacidade através da utilização de dados pessoais, disseminação de informações falsas, diminuição da capacidade de resolução de problemas e possível substituição de empregos por automação.

São propostas soluções responsáveis, incluindo transparência e consentimento informado, mecanismos de verificação de fatos, educação e programas de literacia digital, e a criação de diretrizes e regulamentos éticos específicos para a IA. A colaboração entre promotores, reguladores, organizações de direitos humanos e o público é essencial para resolver estes problemas, garantindo que diferentes perspectivas sejam consideradas e soluções equilibradas sejam desenvolvidas. Uma correta estruturação de políticas, com definição clara de responsabilidades, alocação de recursos e criação de canais de comunicação, é essencial para a implementação eficaz de práticas éticas. Além disso, o controle das informações deve equilibrar a privacidade e a segurança dos usuários com a transparência, garantindo revisões regulares das práticas para alinhar os princípios éticos.

Palavras-chave: implicações. IA. éticas. ChatGPT. privacidade.

ABSTRACT

This paper investigates the ethical implications of using ChatGPT across various contexts, addressing critical issues such as privacy, misinformation dissemination, technological dependency, and employment impact. These ethical concerns affect users, developers, and society at large, necessitating

in-depth analyses to understand their consequences. The critical analysis reveals significant risks, including privacy violations through the use of personal data, the spread of false information, a decrease in problem-solving abilities, and potential job displacement due to automation. Responsible solutions are proposed, including transparency and informed consent, fact-checking mechanisms, digital literacy education programs, and the creation of specific ethical guidelines and regulations for AI. Collaboration among developers, regulators, human rights organizations, and the public is essential to address these issues, ensuring that different perspectives are considered and balanced solutions are developed. Proper policy structuring, with clear definitions of responsibilities, resource allocation, and the creation of communication channels, is crucial for the effective implementation of ethical practices. Furthermore, information control must balance user privacy and security with transparency, ensuring regular reviews of practices align with ethical principles.

Keywords: implications, AI, ethics, ChatGPT, privacy

1 INTRODUÇÃO

O ChatGPT, um modelo de linguagem de inteligência artificial (IA) revolucionário, abre muitas possibilidades, algumas delas inéditas, para a interação entre humanos e máquinas. No entanto, existe vários contextos que são necessários serem analisados, à medida que essa tecnologia avança a passos largos, surgem preocupações urgentes sobre suas implicações éticas, jurídicas, profissionais e sociais em diversos contextos. Este trabalho, intitulado "Estudo de caso: Implicações éticas do uso do ChatGPT", oferece uma análise profunda das principais questões éticas relacionadas ao uso do ChatGPT e propõe soluções responsáveis para garantir um futuro promissor para a IA.

2 OS DIFERENTES CONTEXTOS

Relevância Ética

A relevância ética do ChatGPT reside em sua capacidade de influenciar significativamente a interação entre humanos e máquinas. Questões críticas emergem, incluindo a privacidade dos dados dos usuários, a disseminação de desinformação, a dependência tecnológica crescente e o impacto no emprego. Compreender essas implicações é vital para garantir que a tecnologia seja utilizada de maneira justa e responsável. Enfrenta desafios significativos e requer soluções tecnológicas adequadas para garantir um ambiente inclusivo e seguro. As limitações e vieses inerentes ao modelo exigem que pensadores através da ética promovam o pensamento crítico e diversidade de fontes para mitigar distorções. A prevenção do plágio e desonestidade acadêmica é essencial, com a implementação de ferramentas de detecção e reforço da integridade acadêmica. Evitar a dependência tecnológica excessiva é crucial para o desenvolvimento cognitivo de todos que querem conhecer a tecnologia de IA combinando o uso do ChatGPT com métodos que fomentem habilidades independentes. A acessibilidade universal às ferramentas de IA é fundamental para garantir oportunidades iguais de

aprendizado, enquanto medidas robustas de segurança e privacidade são necessárias para proteger os dados dos usuários. Soluções tecnológicas como ferramentas de avaliação ética, integração com plataformas educacionais, mecanismos de feedback, ferramentas de tradução e localização, e sistemas de segurança cibernética são essenciais para maximizar os benefícios do ChatGPT de maneira responsável e eficaz.

"A ética no uso do ChatGPT nas escolas é crucial para garantir uma educação baseada na integridade e na formação de cidadãos conscientes. Compreender as limitações e vieses do ChatGPT, promover a criatividade e o pensamento crítico dos alunos, prevenir o plágio e desonestidade acadêmica, e assegurar a acessibilidade para todos são aspectos essenciais para uma abordagem ética e responsável."

(FÁVERO JÚNIOR, 2024).

Análise Crítica

A inteligência artificial (IA) se apresenta como uma força transformadora inegável em nossa sociedade, impulsionando inovações em diversos setores e prometendo um futuro repleto de benefícios. No entanto, essa jornada promissora é acompanhada por preocupações crescentes sobre as implicações éticas da IA, especialmente em relação à privacidade, à disseminação de informações falsas, à dependência tecnológica e ao impacto no mercado de trabalho.

Violações de privacidade e coleta de dados: riscos para as liberdades individuais. A inteligência artificial representa riscos significativos para a privacidade devido à recolha e processamento massivo de dados pessoais. Algoritmos sofisticados podem criar perfis detalhados que revelam informações mais íntimas do que as pessoas estão dispostas a compartilhar.

A possibilidade de fuga ou utilização indevida destas informações sensíveis representa uma ameaça constante, pondo em perigo a liberdade pessoal e a autonomia na vida privada. Disseminação de desinformação e manipulação: uma ameaça à verdade e à democracia.

A utilização da inteligência artificial para criar e disseminar desinformação em grande escala representa uma ameaça significativa. Esta abordagem mina a confiança nas instituições, dificulta a tomada de decisões informadas e representa uma ameaça à democracia. A proliferação de notícias falsas e a formação de bolhas de informações filtradas podem levar à polarização social e à erosão dos valores democráticos. Redução da dependência da tecnologia e capacidade de resolução de problemas. O uso excessivo de sistemas de IA pode levar a uma diminuição na capacidade de um indivíduo para resolver problemas. A crescente dependência da tecnologia para tarefas que antes exigiam pensamento crítico e criativo pode enfraquecer estas importantes capacidades, tornando os indivíduos mais suscetíveis à manipulação e menos capazes de lidar com desafios complexos. Os impactos sociais e econômicos da automação A automação baseada em IA pode levar à perda de empregos em diferentes setores e criar desemprego e instabilidade social. A adaptação às novas exigências do mercado de trabalho exige o desenvolvimento de políticas públicas eficazes em termos de qualificação profissional e de formação contínua. Necessidade de governança e transparência para reduzir os riscos associados à IA e garantir

uma utilização responsável, devem ser criados mecanismos fortes de governança e transparência. As empresas que desenvolvem e utilizam inteligência artificial devem ser transparentes nos seus processos, garantir o consentimento informado dos utilizadores e proteger os seus dados. Uma regulamentação clara e eficaz é essencial para estabelecer limites éticos e garantir que a inteligência artificial seja utilizada em benefício da sociedade. Estas orientações constituem a base para soluções responsáveis na utilização da inteligência artificial, garantindo que os seus benefícios sejam maximizados e os seus riscos minimizados em benefício da sociedade na totalidade.

Soluções Responsáveis

A inteligência artificial (IA) está a tornar-se uma força transformadora inegável na nossa sociedade, impulsionando a inovação em todas as áreas e proporcionando um futuro repleto de benefícios. No entanto, esta jornada promissora é acompanhada por preocupações crescentes sobre as implicações éticas da inteligência artificial, apelando à implementação de soluções responsáveis para garantir que a sua utilização seja justa, transparente e benéfica para todos.

Para estabelecer uma boa governação da IA, devem ser definidos princípios éticos claros para orientar o seu desenvolvimento e utilização, incluindo o respeito pela privacidade, a não discriminação e a autonomia das pessoas. Além disso, são necessários mecanismos de controle eficazes para monitorizar e regular o impacto social da IA promovendo simultaneamente o diálogo entre os diferentes setores da sociedade.

A transparência desempenha um papel vital na construção da confiança e na garantia da responsabilização na utilização da inteligência artificial. Isso inclui a documentação clara dos processos de desenvolvimento, a explicação acessível sobre o funcionamento dos sistemas de IA e a criação de canais de feedback para os usuários.

Promover a diversidade e inclusão no desenvolvimento da IA é crucial para evitar vieses e discriminação. Isso implica formar equipes diversas, utilizar conjuntos de dados representativos e avaliar o impacto da IA em diferentes grupos sociais. A segurança e a proteção de dados são fundamentais para proteger a privacidade dos usuários e prevenir o uso indevido da IA. Medidas como segurança robusta, minimização de dados e transparência sobre o uso das informações pessoais são essenciais nesse contexto. É essencial estabelecer mecanismos claros de responsabilização e responsabilização para garantir que as pessoas envolvidas na criação e utilização da IA sejam responsabilizadas pelas suas ações. Isto envolve definir claramente responsabilidades, estabelecer mecanismos de responsabilização e realizar auditorias independentes dos sistemas de IA.

Estas orientações constituem a base para soluções responsáveis na utilização da inteligência artificial, garantindo que os seus benefícios sejam maximizados e os seus riscos minimizados em benefício da sociedade na totalidade.

Colaboração e Discussão

A colaboração entre desenvolvedores, reguladores, organizações de direitos humanos e o público é essencial para resolver esses problemas éticos. Mediante discussões e parcerias, é possível garantir que diferentes perspectivas sejam consideradas e soluções equilibradas sejam desenvolvidas.

A colaboração é crucial para integrar diversas perspectivas no debate ético sobre a IA desenvolver soluções equilibradas que considerem os benefícios e riscos da tecnologia, e construir confiança entre os setores da sociedade. Além disso, promove a inovação responsável ao abordar de forma proativa os desafios éticos da IA assegurando que seu uso contribua positivamente para a sociedade como um todo.

“UNESCO reconhece a importância da colaboração internacional em pesquisa, inovação e desenvolvimento de políticas públicas para garantir que a inteligência artificial (IA) seja utilizada de forma responsável e benéfica para todos. Essa colaboração deve ser realizada em um espírito de inclusão, participação e transparência, e deve visar o desenvolvimento de soluções que atendam às necessidades de todos os stakeholders.”

(UNESCO, 2021, p. 18).

CONSIDERAÇÕES FINAIS

A IA oferece um potencial imenso para o progresso e a inovação, mas seu uso responsável e ético é fundamental para garantir um futuro positivo para a sociedade. Abordar os desafios relacionados à privacidade, à desinformação, à dependência tecnológica e ao impacto no mercado de trabalho exige um esforço conjunto de governos, empresas, academia e sociedade civil. Através da colaboração e do diálogo, podemos garantir que a IA seja utilizada de forma ética, transparente e responsável, beneficiando a todos e construindo um futuro mais justo e próspero.

A implementação de soluções responsáveis para a IA é crucial para maximizar seus benefícios e minimizar seus riscos para a sociedade. A busca pela transparência na IA requer o engajamento contínuo de todos os stakeholders. Governos, empresas, academia e sociedade civil devem colaborar para desenvolver inovações que assegurem um uso ético e responsável da IA em benefício de todos.

A governança eficaz, a transparência na coleta e uso de dados, a promoção da diversidade e inclusão, a segurança cibernética robusta, a definição clara de responsabilidades e a educação contínua são fundamentais para construir um futuro em que a IA seja um motor de progresso e bem-estar social. A colaboração entre todos os setores é essencial para garantir que a IA contribua positivamente para uma sociedade mais justa e equitativa. Com diálogo aberto e participação ativa, podemos pavimentar um caminho para um futuro da IA que seja ético, transparente e responsável, beneficiando a todos e construindo um mundo melhor para as gerações futuras.

REFERÊNCIAS

FÁVERO JÚNIOR, Edson. Como será sua vida sem precisar trabalhar: O Livro de Edson Fávero Júnior. Editora Ultraconhecimento, 2023.

GRAÇA, Fabiano; BARBOSA, Inah Lúcia; FAGUNDES, Vladimir. Análise crítica sobre o ChatGPT. Disponível em: <https://www.serpro.gov.br/menu/noticias/noticias/2023/chatgpt-o-que-pode-dar-errado>. Acesso em: 15 jun. 2024.

Soares, Daniela; Cortez, Adriel. ChatGPT: o que é, como usar e dicas de comandos para o dia a dia. 2023. Disponível em: <https://www.alura.com.br/artigos/chatgpt>. Acesso em: 25 jun. 2024.

UNESCO. Recomendação sobre a Ética da Inteligência Artificial. Disponível em: <https://www.unesco.org/pt/fieldoffice/brasil/expertise/artificial-intelligence-brazil>. Acesso em: 28 jun. 2024.

APÊNDICE L - GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

- Quais os principais resultados obtidos pelo estudo?

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Artigo escolhido: *Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows.*

Qual o objetivo do estudo descrito pelo artigo?

O objetivo primário do artigo é entender em como ferramentas de Auto ML são usadas no dia a dia por profissionais de TI que variam de novatos até pesquisadores da área, o estudo é baseado em entrevista/pesquisa feita com 16 pessoas com variados níveis de experiência na área de TI em geral. O artigo examina os benefícios e as deficiências percebidas pelos usuários das atuais soluções de Auto ML. Faz uma análise do papel dos humanos no fluxo de desenvolvimento de machine learning e identifica as tarefas nas quais a intuição e expertise humana são indispensáveis no processo e o que efetivamente pode ser automatizado. De forma geral o artigo tem como objetivo propor que o desenvolvimento das ferramentas de Auto ML redirecione o objetivo de total automação dos processos de IA para um modelo colaborativo entre humanos e máquinas, de forma que a expertise humana seja integrada com processos automatizados.

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

O estudo teve como objetivo entender como ferramentas de Auto ML são usadas em cenários reais de machine learning. A falta de compreensão geral de quais estágios estas ferramentas são utilizadas no processo foi o principal motivador do estudo. Foram encontrados pouca evidência dos padrões de uso, sobre como essas ferramentas são integradas nas etapas do processo de machine learning. Também quais são os benefícios e as limitações do uso destas ferramentas. E sobre como essas automações impactam os papéis e as responsabilidades dos desenvolvedores no processo de machine learning. O estudo provê dados empíricos sobre o uso destas ferramentas que destacam os benefícios e as dificuldades experienciadas pelos usuários, para desta forma ajudar a guiar o futuro do

desenvolvimento de ferramentas de Auto ML para melhor prover o que é realmente necessário para os usuários.

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

Os autores usaram uma metodologia de pesquisa qualitativa para obter e analisar as informações necessárias para o estudo. A metodologia consistiu em; recrutamento dos participantes, estes deveriam ter pelo menos alguma experiência de uso com ferramentas de Auto ML. Após a seleção de 16 participantes, os pesquisadores classificaram estes em diferentes níveis de; expertise, tipo de indústria e papel exercido e quais casos de uso os participantes utilizavam soluções de IA. Os autores então entrevistaram todos os participantes, tanto ao vivo quanto remotamente, e buscaram entender os casos de uso, a experiência que estes tiveram no desenvolvimento de soluções de IA tanto com ferramentas de Auto ML quanto sem. Eles então aplicaram codificação indutiva de forma a extrair conceitos e padrões relevantes das entrevistas, com isso eles puderam categorizar os dados em temas mais amplos, das quais incluíam; vantagens e desvantagens de Auto ML, percepção geral de ferramentas de Auto ML, estratégias de processos com e sem ferramentas de Auto ML, desafios em geral no processo de machine learning e fatores que influenciaram a adoção de ferramentas de Auto ML.

Quais os principais resultados obtidos pelo estudo?

O estudo mostrou que embora as ferramentas de Auto ML tenham aumentado a acessibilidade elas não resolvem todos os desafios enfrentados pelos profissionais. As ferramentas servem para automatizar etapas específicas, como ajuste de hiper parâmetros e seleção de modelos. Porém elas têm deficiências especialmente na falta de suporte para tarefas de pré-processamento e pós-processamento de dados, como limpeza de dados, engenharia de características e interpretação de modelos. A falta de personalização e transparência também gerou problemas de confiança com os profissionais com mais experiência. Desta forma o estudo reafirma o papel indispensável dos humanos no fluxo de trabalho de aprendizado de máquina. Onde a expertise humana é essencial para tarefas que exigem conhecimento de domínio, compreensão contextual e considerações éticas. Com isto ao invés de substituir o envolvimento humano, as ferramentas de Auto ML devem complementar os profissionais, automatizando tarefas repetitivas para que estes possam se concentrar em decisões de alto nível. Com base nestes resultados, o estudo defende a mudança de paradigma no desenvolvimento destas ferramentas, para promover uma parceria colaborativa entre humanos e máquinas.

APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

- Gráficos de perda e de acurácia;
 - Imagem gerada na seção “**Mostrar algumas classificações erradas**”, apresentada na aula prática.
- Informações:
- **Base de dados:** Fashion MNIST Dataset
 - **Descrição:** Um dataset de imagens de roupas, onde o objetivo é classificar o tipo de vestuário. É semelhante ao famoso dataset MNIST, mas com peças de vestuário em vez de dígitos.
 - **Tamanho:** 70.000 amostras, 784 features (28x28 pixels).
 - **Importação do dataset:** Copiar código abaixo.

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

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

B – RESOLUÇÃO

1 Classificação (RNA)

```
# Bibliotecas

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
# Importar os dados do dataset fashion_mnist

data = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = data.load_data()
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", y_test.shape)
print("y_test.shape: ", y_test.shape)
display(x_train)
x_train, x_test = x_train/255.0, x_test/255.0
i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)

model = tf.keras.models.Model(i, x)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
r = model.fit(x_train,
              y_train,
              validation_data=(x_test, y_test),
              epochs=10)

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

# Plotar a acurácia
```

```

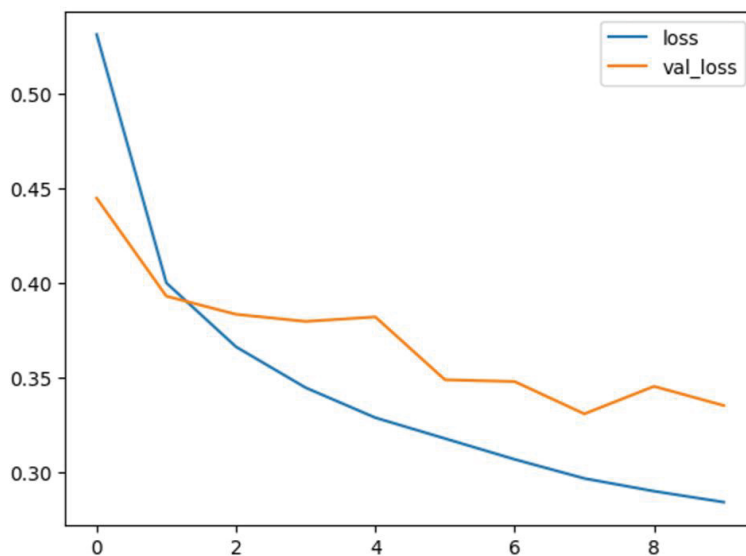
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
# Avaliar o modelo com a base de teste
print( model.evaluate(x_test, y_test) )
y_pred = model.predict(x_test).argmax(axis=1)
print(y_pred)
# Matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
                      show_normed=True)
misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))

```

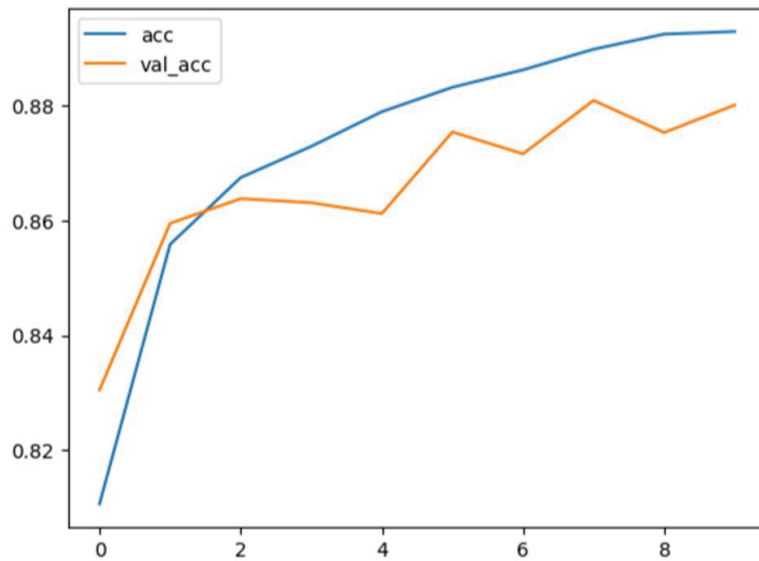
Gráficos de perda e de acurácia.

Gráfico de perda:



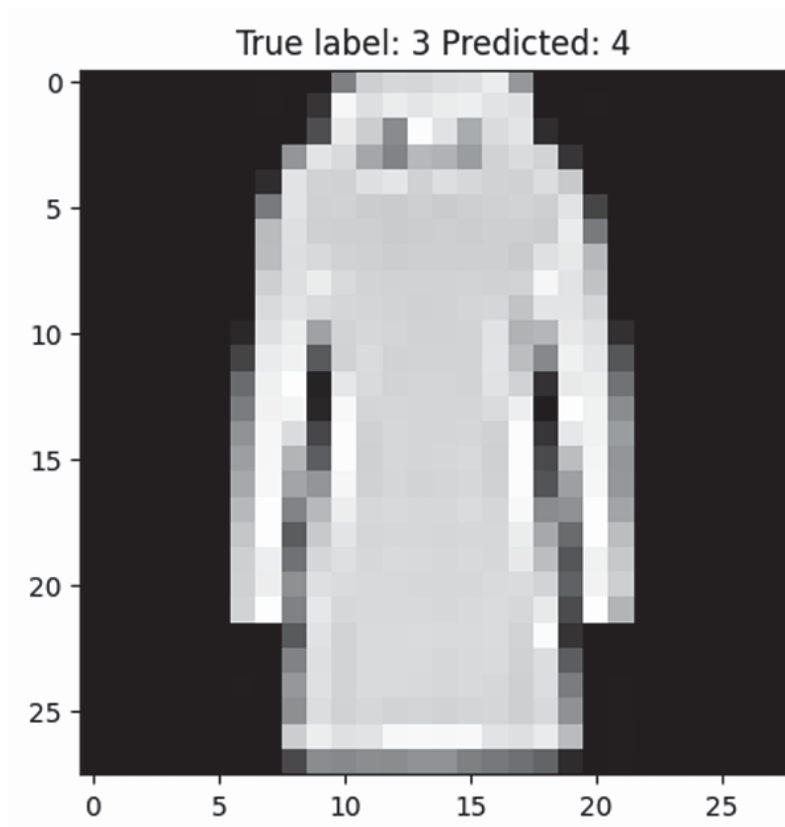
O modelo está aprendendo com o passar das épocas e melhorando sua capacidade de prever corretamente. O valor loss está relacionado à perda durante o treinamento e o valor val_loss está relacionado à perda durante a validação (dados que não conhecia durante o treinamento).

Gráfico de acurácia:



A acurácia do modelo está melhorando com o passar das épocas, ou seja, o percentual de previsões corretas de treinamento `acc` e validação `val_acc` estão aumentando conforme o modelo vai aprendendo.

Imagem gerada na seção “Mostrar algumas classificações erradas”, apresentada na aula prática:



Nesse exemplo é possível verificar que o RNA classificou como label 4 (casaco) onde o correto seria label 3 (vestido), talvez pela similaridade e falta de imagens e/ou épocas para treino e validação.

2 Regressão (RNA)

```

import tensorflow as tf
import pandas as pd
from math import sqrt

from sklearn.model_selection import train_test_split
from tensorflow.python.keras import backend
from sklearn.metrics import r2_score, mean_squared_error

# Verificar versão do TensorFlow
tf.__version__

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
]

Data
X = data.iloc[:, :-1].astype(float)
Y = data.iloc[:, -1].astype(float)

print(f"X:\n\n", X)
print(f"\n\nY:\n\n", Y)
# Número de linhas e colunas
data.shape
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)

```

```

# 3 camadas
i = tf.keras.layers.Input(shape=(11,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)

model = tf.keras.models.Model(i, x)
# Criação de funções para as métricas R2 e RMSE serem inseridas no modelo
def rmse(y_true, y_pred):
    return backend.sqrt(backend.mean( backend.square(y_pred - y_true), axis=-
1) )

def r2(y_true, y_pred):
    media = backend.mean(y_true)
    num    = backend.sum (backend.square(y_true - y_pred))
    den    = backend.sum (backend.square(y_true - media))
    return (1.0 - num/den)
# Compilação
optimizer=tf.keras.optimizers.Adam(learning_rate=0.05)
# optimizer=tf.keras.optimizers.SGD(learning_rate=0.2, momentum=0.5)
# optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
              loss="mse",
              metrics=[rmse, r2])
# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
                    monitor='val_loss',
                    patience=20,
                    restore_best_weights=True)
r = model.fit(x_train, y_train,
              epochs=1500,
              validation_data=(x_test, y_test),
              callbacks=[early_stop])
plt.plot( r.history["loss"], label="loss" )
plt.plot( r.history["val_loss"], label="val_loss" )
plt.legend()
plt.plot( r.history["rmse"], label="rmse" )
plt.plot( r.history["val_rmse"], label="val_rmse" )
plt.legend()
plt.plot( r.history["r2"], label="r2" )
plt.plot( r.history["val_r2"], label="val_r2" )

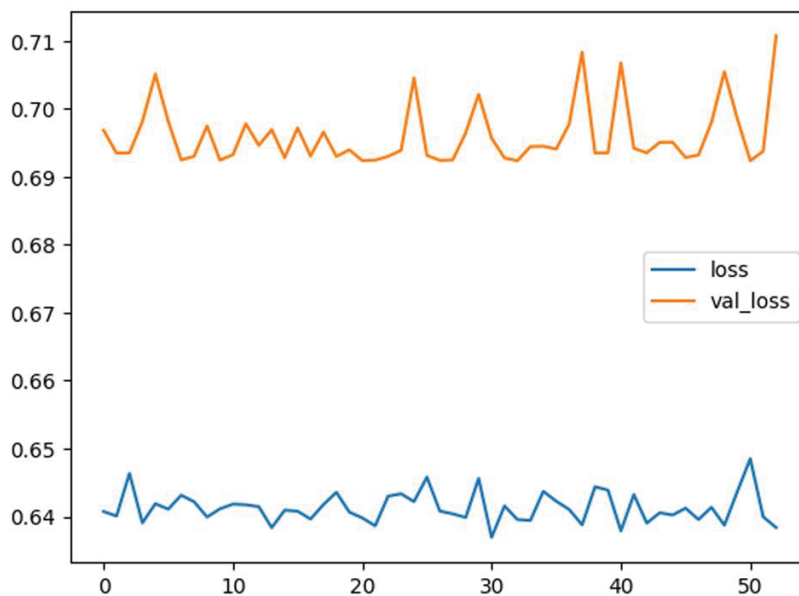
```



```
plt.legend()
y_pred = model.predict(x_test).flatten()
# Cálculo das métricas de acurácia: mse, r2 e rmse
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred)
# Resultados das métricas de acurácia
print("mse      = ", mse)
print("rmse     = ", rmse)
print("r2       = ", r2)
```

Gráficos de avaliação do modelo.

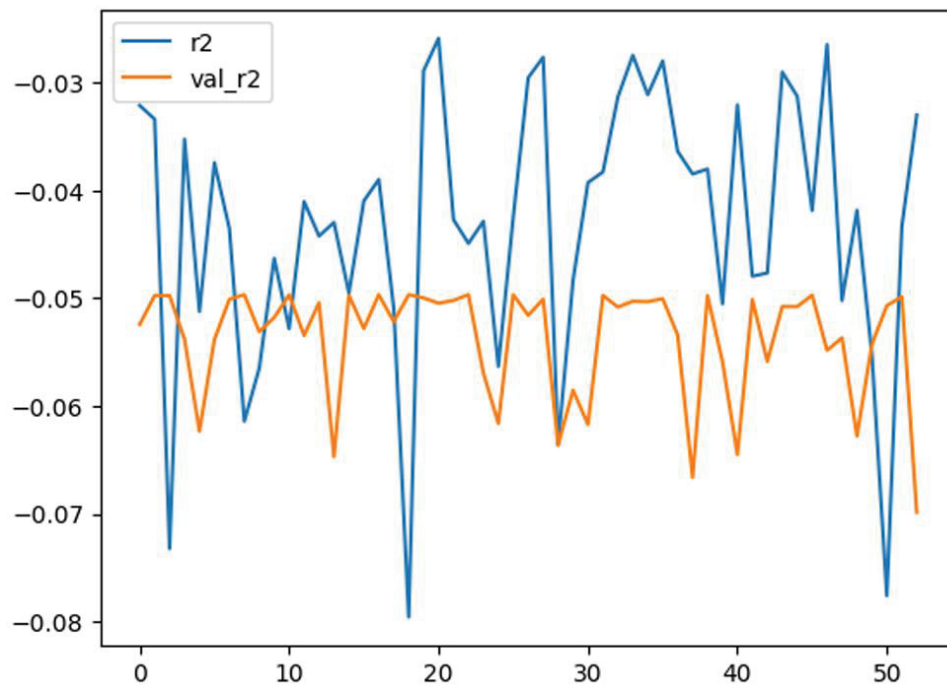
Gráfico de perda:



O *loss* (treinamento) e o *val_loss* (validação) apresentam uma variação de valores ao longo das épocas, mas não mostram uma tendência clara de melhoria significativa. Em algumas épocas, o *loss* diminui indicando que o modelo está se ajustando melhor aos dados de treinamento, porém, o *val_loss* permanece alto ou aumentando em algumas épocas, o que pode indicar *overfitting*.

Métricas de avaliação do modelo.

Gráfico de R2:



O R^2 é uma métrica que avalia o quanto o modelo consegue explicar a variação dos dados e varia entre 0 e 1. R^2 igual a zero significa que o modelo não consegue explicar nada da variância nos dados, igual a um significa que o modelo consegue explicar 100% da variância nos dados e negativo significa que o modelo está fazendo previsões muito ruins.

No caso apresentado, tanto o r^2 como o val_r^2 demonstram oscilações negativas, isso indica que o modelo não está conseguindo aprender e generalizar de forma estável durante as épocas.

3 Sistema de Recomendação

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

```

from tensorflow.keras.layers import Input, Embedding, Flatten, Concatenate,
Dense
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD
from sklearn.utils import shuffle
df = pd.read_csv("/content/Base_livros.csv")
df.head()
# Embeddings > devem ser categóricos

df.ID_usuario = pd.Categorical(df.ID_usuario)
df['new_user_id'] = df.ID_usuario.cat.codes

df.ISBN = pd.Categorical(df.ISBN)
df['new_isbn_id'] = df.ISBN.cat.codes

df.head()
# Dimensões
N = len(set(df.new_user_id))
M = len(set(df.new_isbn_id))

# dimensão do embedding (tentar outros)
K = 10

print(f"N = {N}")
print(f"M = {M}")
# Usuário
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

# ISBN - Livro
m = Input(shape=(1,))
m_emb = Embedding(M, K)(m) # saída : num_samples, 1, K
m_emb = Flatten()(m_emb)   # saída : num_samples, K

x = Concatenate()([u_emb, m_emb])

x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)

```

```

print(f"Modelo: \n\n{model.summary()}")
# Compilar o modelo

model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)
# Separar os dados e pré-processamento

user_ids, isbn_ids, ratings = shuffle(df.new_user_id, df.new_isbn_id,
df.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

train_user = user_ids[:Ntrain]
train_isbn = isbn_ids[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_isbn = isbn_ids[Ntrain:]
test_ratings = ratings[Ntrain:]

# centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating
# Treinar o modelo

epochs = 25
r = model.fit(
    x=[train_user, train_isbn],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso
    validation_data=(test_user, test_isbn, test_ratings)
)

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

```

```

# Gerar o array com o usuário único
# repete a quantidade de livros
input_usuario = np.repeat(a=29888, repeats=M)
book = np.array(list(set(isbn_ids)))

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

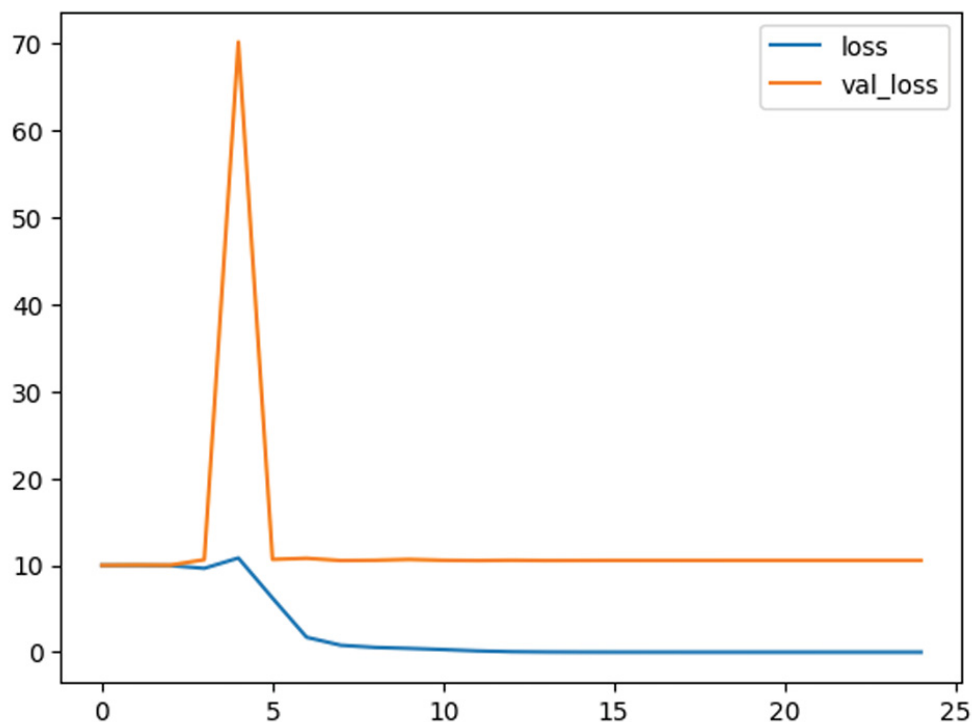
# descentraliza as predições
rat = preds.flatten() + avg_rating

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

print("Recomendação: Livro - ", book[idx], " / ", rat[idx] , "**")
df.loc[df.new_isbn_id == book[idx]]
df[df['ID_usuario'] == 29888]

```

Gráficos de avaliação do modelo:



Conforme o treinamento avança, o *loss* começa a diminuir constantemente, indicando que o modelo está aprendendo a realizar previsões mais precisas nos dados de treinamento e o *val_loss* não diminui consistentemente.

Exemplo de recomendação de livro para determinado usuário:

Recomendação do livro 121418 para o usuário 29888:

4028/4028 ————— 6s 2ms/step
 Recomendação: Livro - 121418 / 10.18502 *

Livro 121418:

	ISBN	Titulo	Autor	Ano	Editora	ID_usuario	Notas	new_user_id	new_isbn_id
24782	877931267	Not without parables: Stories of yesterday, to ...	Catherine de Hueck Doherty	1977	Ave Maria Press	5903	10	2117	121418

Avaliações do usuário 29888:

	ISBN	Titulo	Autor	Ano	Editora	ID_usuario	Notas	new_user_id	new_isbn_id
128887	1590580109	Black Gold	Charles O'Brien	2002	Poisoned Pen Press	29888	3	11136	21293
128888	159058015X	The Reunion	Curt Autry	2002	Poisoned Pen Press	29888	1	11136	21295
128889	553578979	Iron Fist (Star Wars: X-Wing Series, Book 6)	Aaron Allston	1998	Bantam	29888	6	11136	81128
128890	743436865	Funny Money	James Swain	2002	Atria	29888	10	11136	102931
128891	451410394	Sun Valley	Gena Hale	2002	Signet Book	29888	10	11136	71826
128892	345391101	The Adventures of Lando Calrissian: Lando Calr...	L. Neil Smith	1994	Del Rey Books	29888	6	11136	41183
128893	1591050480	Beloved Stranger	Patricia Crossley	2002	Novelbooks	29888	9	11136	21376
128894	051732170X	Who's Who in the Bible	RONALD BROWNRIGG	1993	Testament	29888	4	11136	4367

O modelo recomendou um livro classificado como ficção religiosa, talvez pelo usuário dar nota alta em três livros de ficção.

4 Deepdream

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

url = https://commons.wikimedia.org/wiki/Special:FilePath/Felis\_catus-  
cat\_on\_snow.jpg

# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
    name = url.split('/')[-1]
    image_path = tf.keras.utils.get_file(name, origin=url)
    img = PIL.Image.open(image_path)
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)

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

# Mostra a imagem
def show(img):
    display.display(PIL.Image.fromarray(np.array(img)))
```

```

# Redução do tamanho da imagem para facilitar o trabalho da RNN
original_img = download(url, max_dim=500)
show(original_img)
display.display(display.HTML('Image                                cc-by:                                <a
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-
cat_on_snow.jpg">Von.grzanka</a>'))
base_model      =      tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')
# Maximizando as ativações das camadas
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]

# Criação do modelo
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)
def calc_loss(img, model):
    # Passe a imagem pelo modelo para recuperar as ativações.
    # Converte a imagem em um batch de tamanho 1.
    img_batch = tf.expand_dims(img, axis=0)
    layer_activations = model(img_batch)
    if len(layer_activations) == 1:
        layer_activations = [layer_activations]

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

    return tf.reduce_sum(losses)
class DeepDream(tf.Module):
    def __init__(self, model):
        self.model = model

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

```

```

print("Tracing")
loss = tf.constant(0.0)

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

    # Calculo do gradiente da perda em relação aos pixels da imagem de
    entrada.
    gradients = tape.gradient(loss, img)

    # Normalizacao dos gradintes
    gradients /= tf.math.reduce_std(gradients) + 1e-8

    # Na subida gradiente, a "perda" é maximizada.
    # Você pode atualizar a imagem adicionando diretamente os gradientes
    (porque eles têm o mesmo formato!)
    img = img + gradients*step_size
    img = tf.clip_by_value(img, -1, 1)

    return loss, img

deepdream = DeepDream(dream_model)
def run_deep_dream_simple(img, steps=100, step_size=0.01):

    img = tf.keras.applications.inception_v3.preprocess_input(img)
    img = tf.convert_to_tensor(img)
    step_size = tf.convert_to_tensor(step_size)
    steps_remaining = steps
    step = 0
    while steps_remaining:
        if steps_remaining>100:
            run_steps = tf.constant(100)
        else:
            run_steps = tf.constant(steps_remaining)
        steps_remaining -= run_steps
        step += run_steps

    loss, img = deepdream(img, run_steps, tf.constant(step_size))

```



```

display.clear_output(wait=True)
show(deprocess(img))
print ("Step {}, loss {}".format(step, loss))

result = deprocess(img)
display.clear_output(wait=True)
show(result)

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

import time
start = time.time()

OCTAVE_SCALE = 1.30

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[: -1]
float_base_shape = tf.cast(base_shape, tf.float32)

for n in range(-2, 3):
    new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

    img = tf.image.resize(img, new_shape).numpy()

    img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

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

end = time.time()
end-start

```



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:



A imagem onírica da esquerda foi obtida pelo modelo *main loop*, possui baixa resolução e padrões seguem a mesma granularidade.

Foi levando o modelo até uma oitava, que gerou a imagem onírica da direita, onde foi corrigido a saída ruidosa, baixa resolução e padrões da mesma granularidade deixando a imagem com uma qualidade superior e mais próximo da onírica.

APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

B – RESOLUÇÃO

1 CONJUNTO DE DADOS E CONTEXTO

Este trabalho utiliza dados reais do IBGE referentes as pessoas de 14 anos ou mais ocupadas no mercado de trabalho brasileiro, segmentadas por nível de instrução entre os anos de 2013 e 2023. Os dados utilizados incluem o total da população ocupada e suas distribuições regionais pelos estados brasileiros. Os dados foram retirados do sistema SIDRA através do PNAD (<https://sidra.ibge.gov.br/tabela/7107>), foram selecionados os anos de 2013 a 2023 com todos os níveis de instrução e a quantidade de dados foi limitada a até 200.000 valores.

1.1 PÚBLICO-ALVO

Estudantes universitários em disciplinas relacionadas à economia, sociologia, políticas públicas e áreas correlacionadas. Professores e pesquisadores interessados na evolução de parâmetros educacionais e econômicos brasileiros. E formuladores de políticas públicas educacionais e sociais.

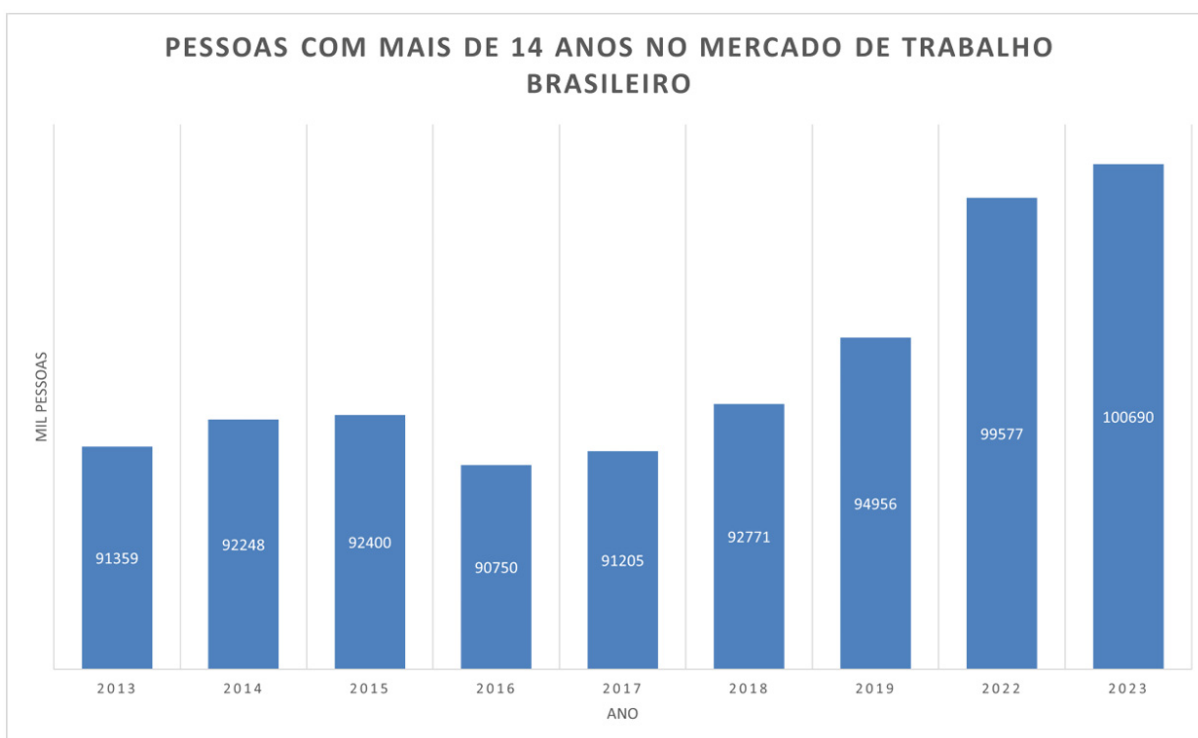
O objetivo é mostrar a importância da relação entre educação e trabalho, apontando as mudanças ocorridas ao longo de uma década de forma a estimular reflexões e ações estratégicas para melhorias na educação e empregabilidade.

1.2 FERRAMENTAS E VISUALIZAÇÕES

Para a execução deste trabalho foi utilizada a ferramenta Microsoft Excel para gerar gráficos de barras empilhadas, por permitir uma visão clara da composição total separada por nível educacional e mapas coropléticos, por permitir uma visualização clara das diferenças regionais.

2 STORYTELLING

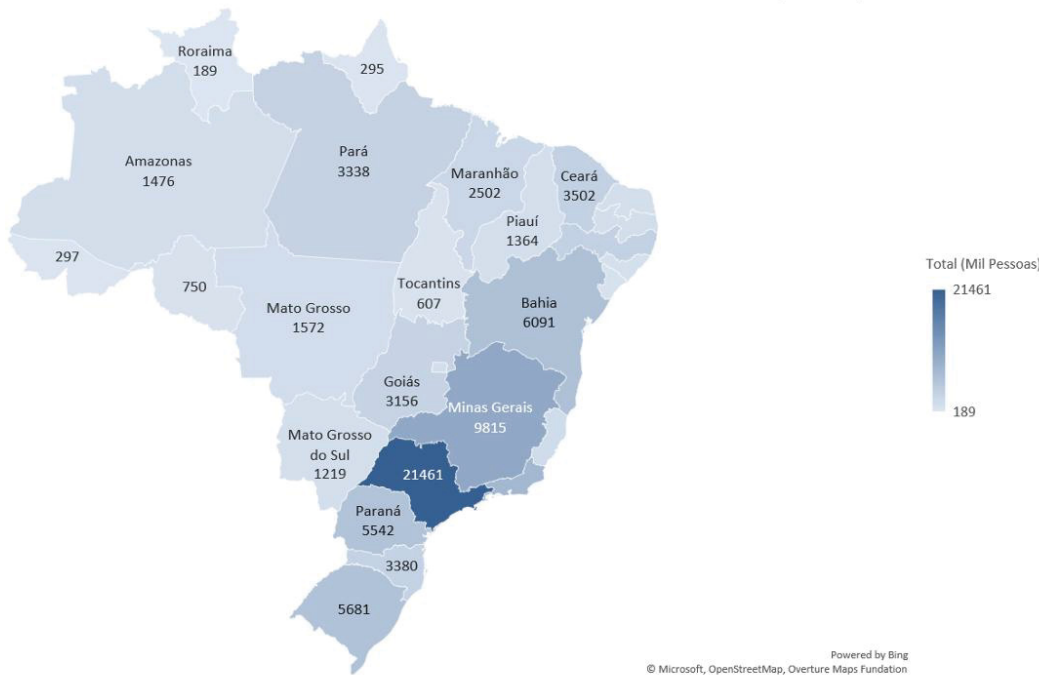
Entre 2013 e 2023 o mercado de trabalho brasileiro passou por profundas transformações que foram influenciadas de forma significativa pelo avanço na educação. Os dados do IBGE revelam que a população inserida no mercado de trabalho não apenas aumentou, mas também mudou seu perfil educacional.



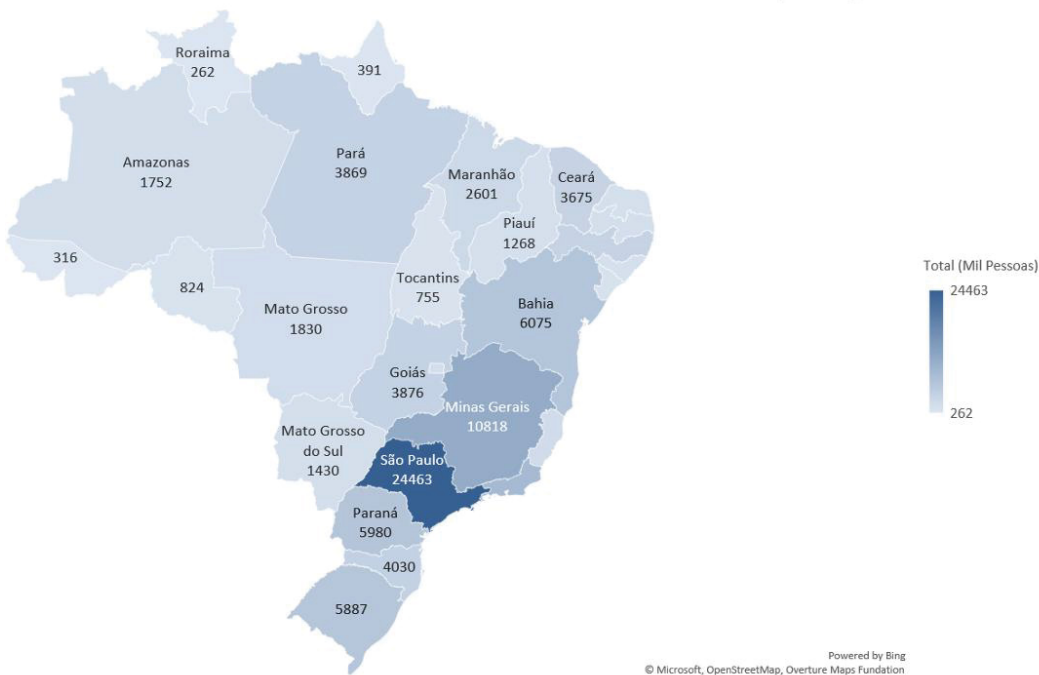
O número de pessoas com mais de 14 anos ocupadas aumentou de 91 milhões em 2013 para mais de 100 milhões em 2023, o que reflete crescimento econômico e maior inserção da população no mercado de trabalho.

Total de pessoas ocupadas entre 2013 e 2023:

Pessoas de 14 anos ou mais no mercado de trabalho (2013)



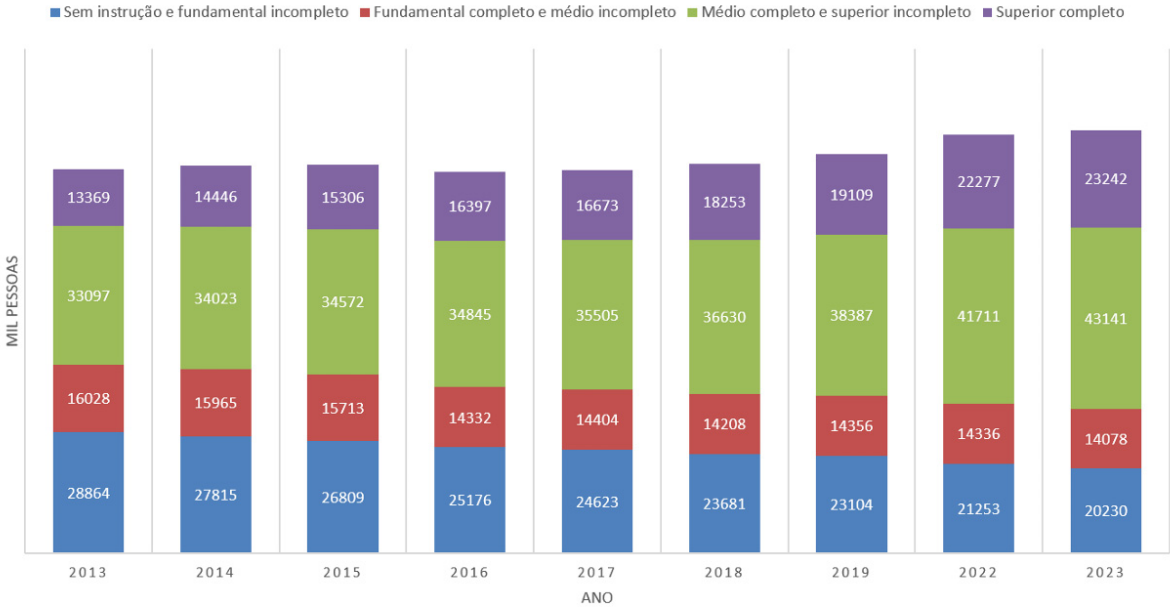
Pessoas de 14 anos ou mais no mercado de trabalho (2023)



Pessoas com ensino superior completo cresceram fortemente, o que reflete políticas públicas de incentivo ao acesso universitário e pessoas sem instrução ou com ensino fundamental incompleto diminuíram substancialmente, mostrando evidências de avanços na educação básica e diminuição do abandono escolar.

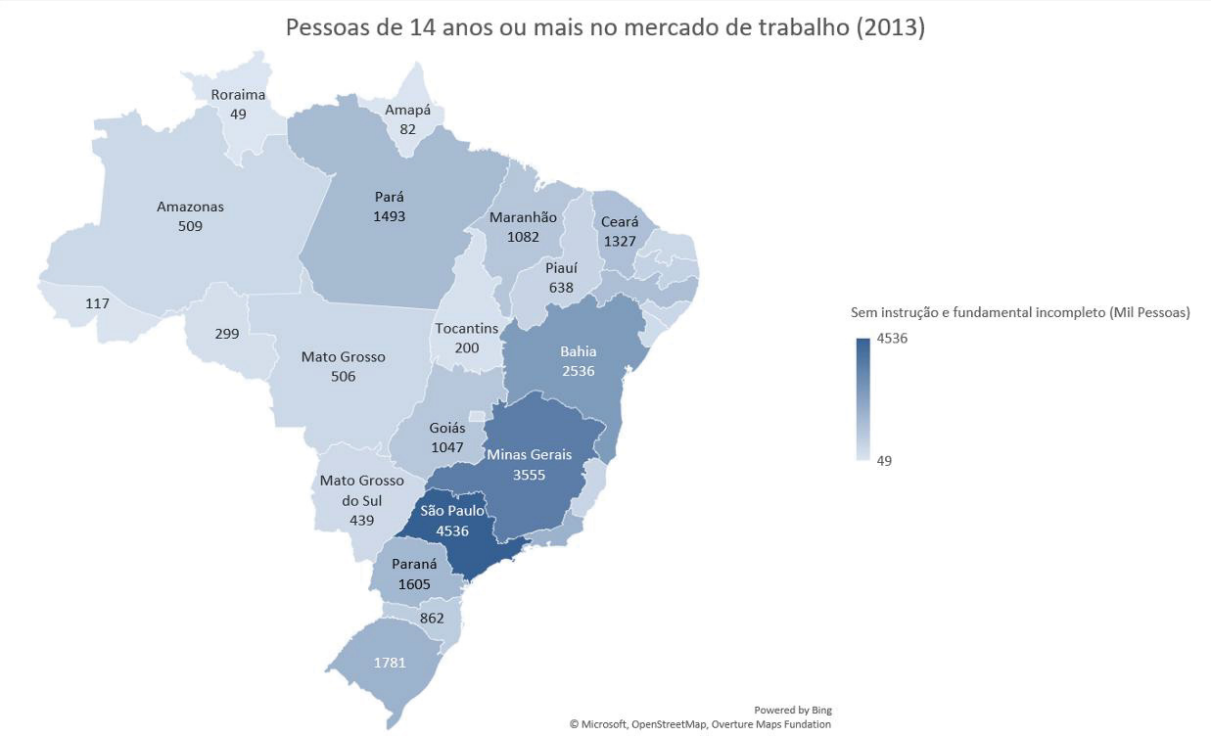
Chart Area

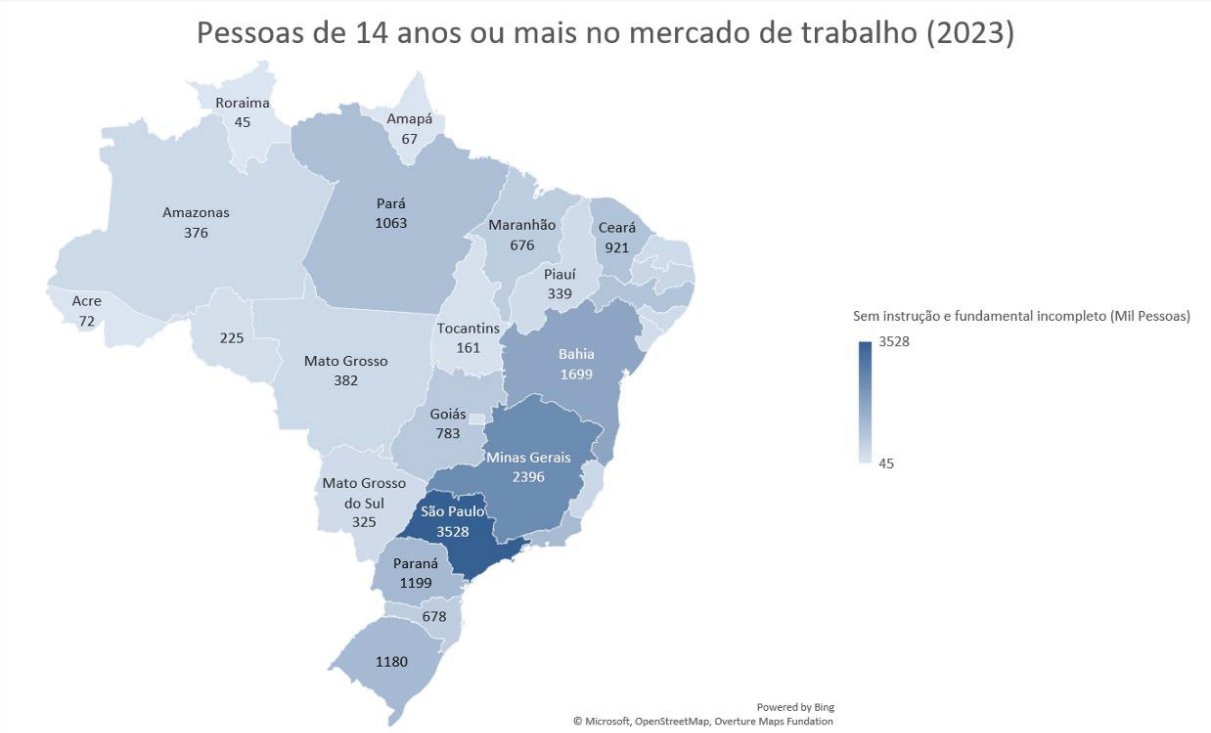
PESSOAS COM MAIS DE 14 ANOS NO MERCADO DE TRABALHO
BRASILEIRO - POR NÍVEL DE INSTRUÇÃO



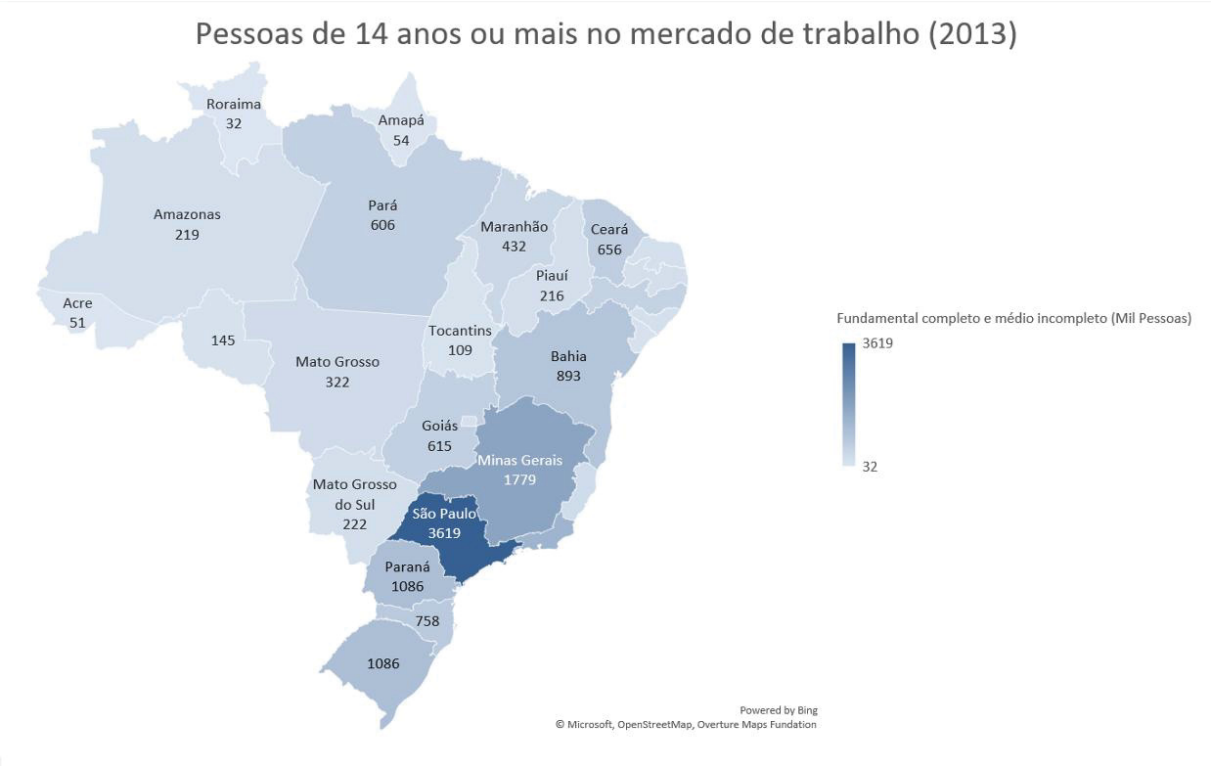
Foi possível observar algumas desigualdades regionais. São Paulo, Minas Gerais e Paraná se destacam como os estados com maior número absoluto em todos os níveis educacionais. Enquanto as regiões Norte e Nordeste ainda apresentam desafios em termos de menor nível de escolaridade, o que indica a necessidade de ações direcionadas para essas áreas.

Pessoas ocupadas sem instrução e ensino fundamental incompleto entre 2013 e 2023:

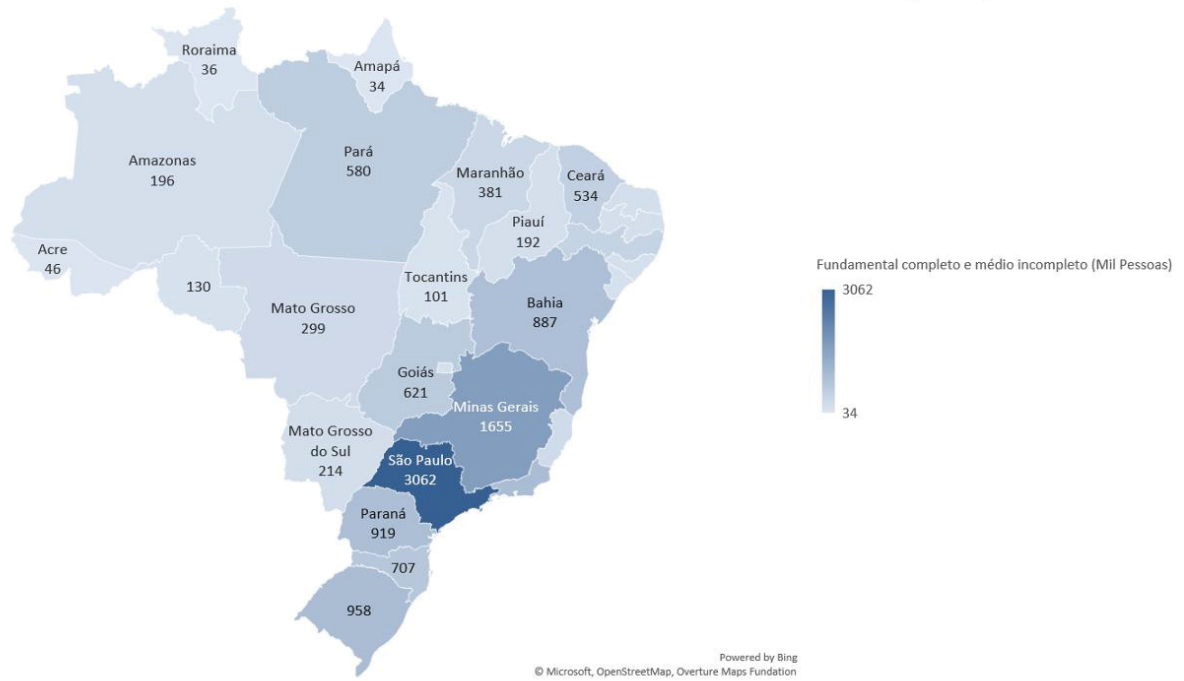




Pessoas ocupadas com ensino fundamental completo e ensino médio incompleto entre 2013 e 2023:

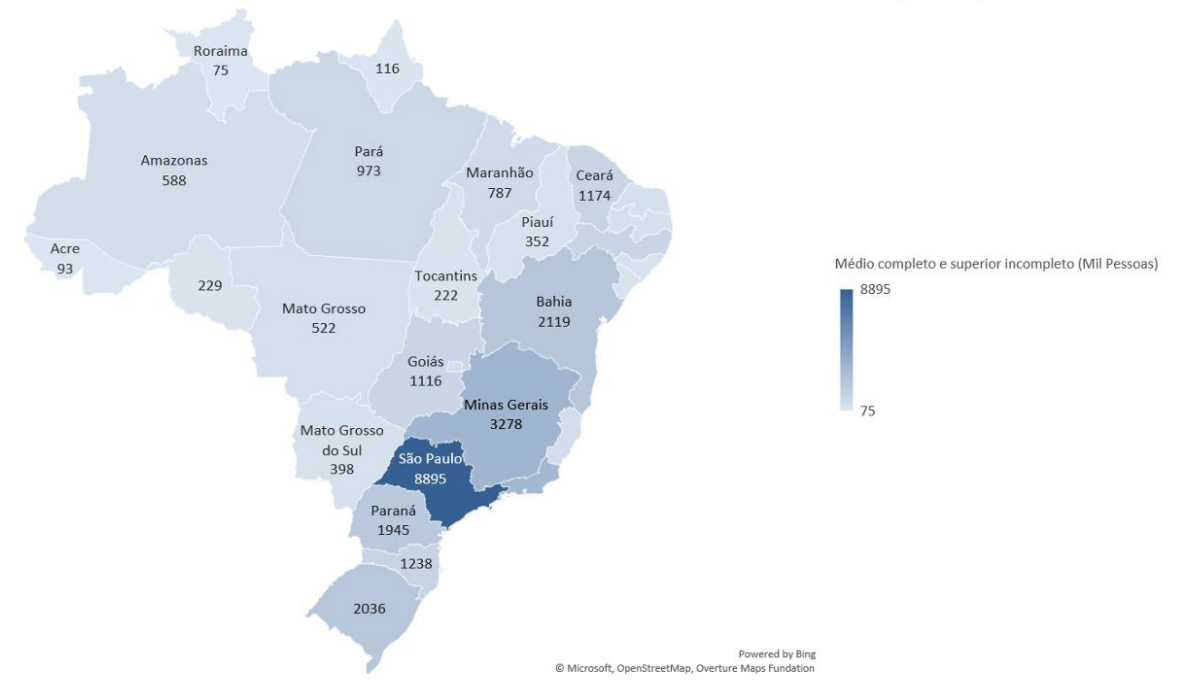


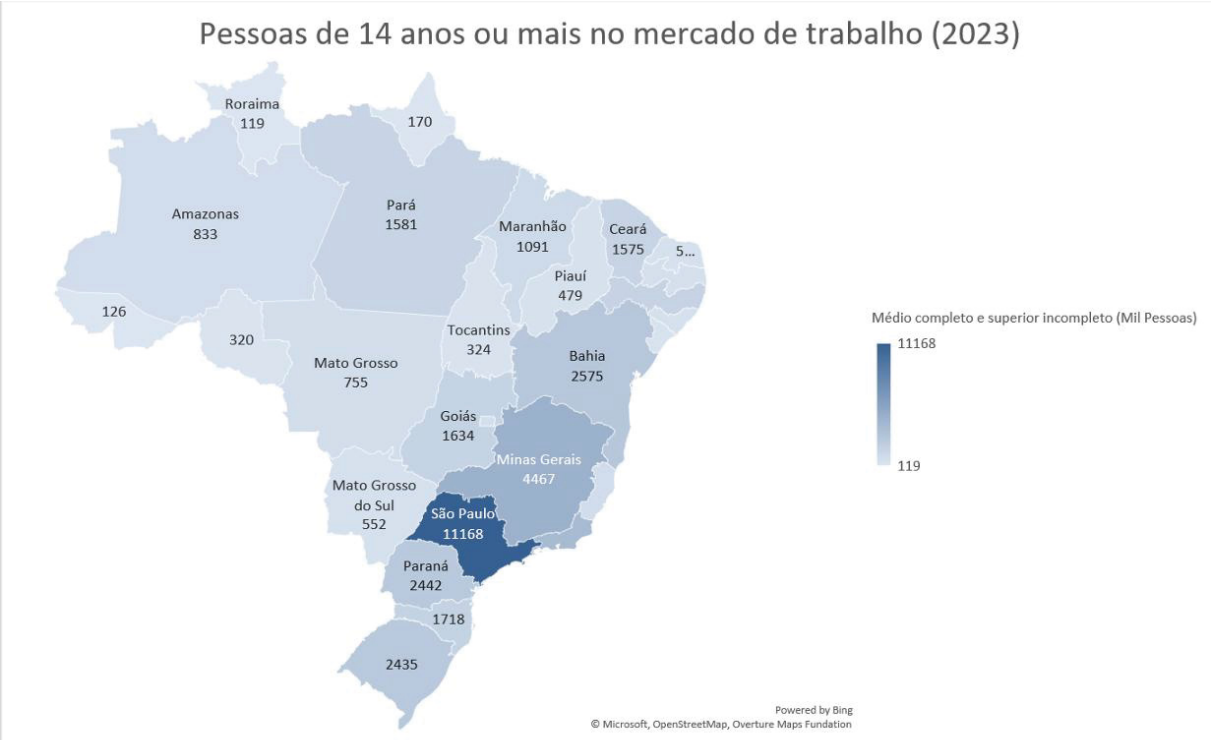
Pessoas de 14 anos ou mais no mercado de trabalho (2023)



Pessoas ocupadas com ensino médio completo e ensino superior incompleto entre 2013 e 2023:

Pessoas de 14 anos ou mais no mercado de trabalho (2013)

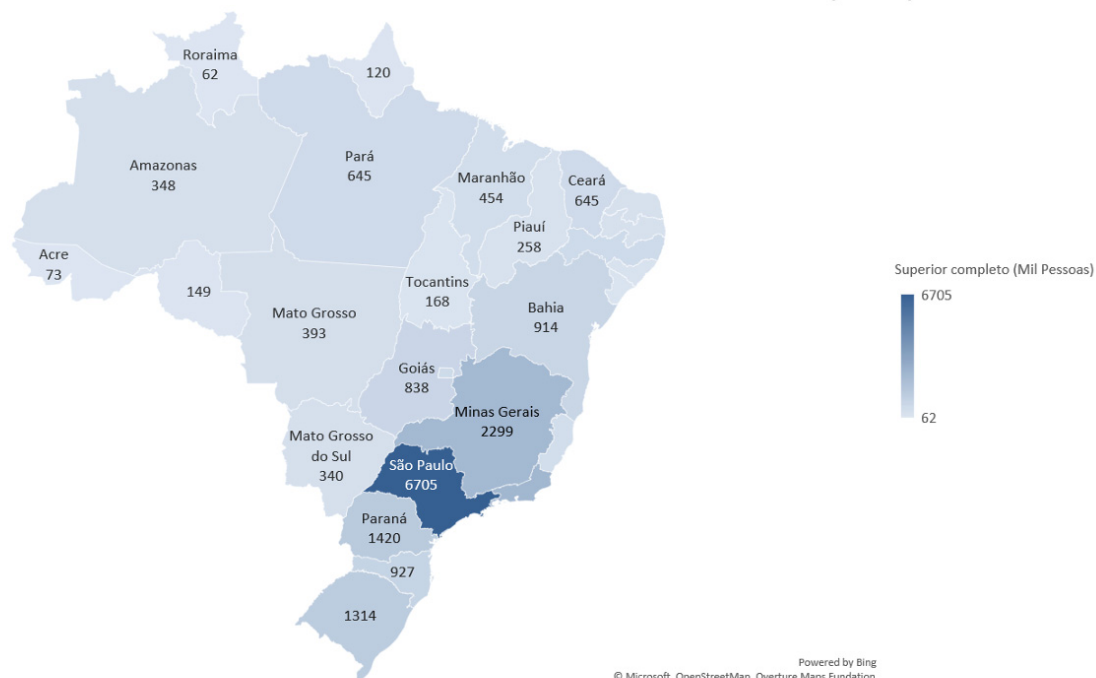




Pessoas ocupadas com ensino superior completo entre 2013 e 2023:



Pessoas de 14 anos ou mais no mercado de trabalho (2023)



Desta forma podemos sugerir que se continue investindo em programas de educação superior e profissionalizante e que o governo e gestores públicos realizem ações mais assertivas em regiões com baixa escolaridade.

As empresas devem aproveitar o aumento da mão-de-obra qualificada para impulsionar inovação e produtividade e investir em capacitação contínua para maximizar o potencial dos trabalhadores.

Assim como também educadores devem promover estudos adicionais sobre a eficácia de políticas educacionais e avaliar se o aumento na escolaridade se traduz efetivamente em melhores condições de trabalho e salários.

APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

1 Algoritmo Genético

```

import numpy as np
import random
import math
import matplotlib.pyplot as plt

# Função para calcular a distância euclidiana entre duas cidades
def distancia(cidade1, cidade2):
    """
    Calcula a distância euclidiana entre duas cidades representadas por
    coordenadas (x, y).
    """
    return math.sqrt((cidade2[0] - cidade1[0])**2 + (cidade2[1] -
    cidade1[1])**2)

# Função para calcular o comprimento total do percurso
def calcular_distancia_total(cromossomo, cidades):
    """
    Calcula a distância total de um percurso com base no cromossomo.
    O cromossomo representa a sequência das cidades a serem visitadas.
    """
    distancia_total = 0
    # A distância entre cada cidade consecutiva do cromossomo
    for i in range(len(cromossomo) - 1):
        distancia_total += distancia(cidades[cromossomo[i]],
        cidades[cromossomo[i+1]])
    # A distância de retorno à cidade inicial
    distancia_total += distancia(cidades[cromossomo[-1]],
    cidades[cromossomo[0]]) # Retorno à cidade inicial
    return distancia_total

# Geração da população inicial (com cidade 0 fixada)
def gerar_populacao_inicial(tamanho_populacao, num_cidades):
    """
    Gera a população inicial do algoritmo genético.
    A cidade de origem (índice 0) é fixada na primeira e última posição
    do cromossomo.
    """
    populacao = []
    for _ in range(tamanho_populacao):

```

```

        cidades = list(range(1, num_cidades - 1)) # Exclui cidade 0 e
cidade final
        random.shuffle(cidades) # Embaralha as cidades para criar uma
solução aleatória
        cromossomo = [0] + cidades + [num_cidades-1] # A cidade inicial
(índice 0) está nas posições 0 e 100
        populacao.append(cromossomo)
    return populacao

# Cruzamento OX (Order Crossover)
def crossover_ox(pai1, pai2):
    """
    Implementa o operador de cruzamento OX (Order Crossover).
    Combina dois cromossomos pais para gerar um filho.
    """
    tamanho = len(pai1)
    corte1 = random.randint(1, tamanho - 2)
    corte2 = random.randint(corte1 + 1, tamanho - 1)

    filho = [-1] * tamanho
    filho[corte1:corte2] = pai1[corte1:corte2] # Mantém a ordem das
cidades entre os cortes

    # Preencher o restante do cromossomo
    indice_filho = 0
    for gene in pai2:
        if gene not in filho:
            while filho[indice_filho] != -1:
                indice_filho += 1
            filho[indice_filho] = gene

    return filho

# Mutação (troca de duas cidades)
def mutacao(cromossomo):
    """
    Aplica o operador de mutação trocando duas cidades de lugar no
cromossomo.

    A mutação ocorre com uma probabilidade pequena (1%).
    """
    i, j = random.sample(range(1, len(cromossomo)-1), 2)

```

```

        cromossomo[i], cromossomo[j] = cromossomo[j], cromossomo[i] # Troca
as duas cidades
    return cromossomo

# Função de seleção por torneio
def selecao_torneio(populacao, cidades, tamanho_torneio=3):
    """
    Seleciona um indivíduo da população utilizando o método de seleção
    por torneio.
    Um número aleatório de indivíduos é escolhido e o melhor entre eles é
    retornado.
    """
    torneio = random.sample(populacao, tamanho_torneio)
    melhor_individuo = min(torneio, key=lambda x:
calcular_distancia_total(x, cidades)) # Indivíduo com menor distância
    return melhor_individuo

# Função principal do Algoritmo Genético
def algoritmo_genetico(num_cidades=100, tamanho_populacao=100,
num_geracoes=1000):
    """
    Função principal do algoritmo genético.
    Ela gerencia a evolução das soluções ao longo das gerações e aplica
    os operadores de cruzamento e mutação.
    """
    # Gera as cidades aleatoriamente dentro do espaço 100x100
    cidades = [(random.randint(0, 100), random.randint(0, 100)) for _ in
range(num_cidades)]
    populacao = gerar_populacao_inicial(tamanho_populacao, num_cidades)
    # Gera a população inicial

    # Obtém a melhor solução inicial (geração 0)
    melhor_solucao_inicial = min(populacao, key=lambda x:
calcular_distancia_total(x, cidades))
    melhor_distancia_inicial =
calcular_distancia_total(melhor_solucao_inicial, cidades)

    # A melhor solução é inicialmente a melhor solução encontrada na
    geração 0
    melhor_solucao = melhor_solucao_inicial

```



```

for geracao in range(num_geracoes):
    nova_populacao = []

    # Cria novos indivíduos por cruzamento e mutação
    while len(nova_populacao) < tamanho_populacao:
        pai1 = selecao_torneio(populacao, cidades) # Seleciona o
primeiro pai
        pai2 = selecao_torneio(populacao, cidades) # Seleciona o
segundo pai

        filho = crossover_ox(pai1, pai2) # Aplica cruzamento para
gerar o filho

        if random.random() < 0.01: # 1% de chance de aplicar mutação
            filho = mutacao(filho)
        nova_populacao.append(filho) # Adiciona o filho à nova
população

    # Preservar a melhor solução da geração anterior
    melhor_individuo = min(nova_populacao, key=lambda x:
calcular_distancia_total(x, cidades))
    nova_populacao[0] = melhor_individuo # A melhor solução de cada
geração é preservada

    populacao = nova_populacao # Atualiza a população para a próxima
geração

    # Atualiza a melhor solução encontrada
    melhor_solucao = min(populacao, key=lambda x:
calcular_distancia_total(x, cidades))

    # Retorna a melhor solução inicial e final, além das distâncias
associadas
    melhor_distancia_final = calcular_distancia_total(melhor_solucao,
cidades)

    return melhor_solucao_inicial, melhor_distancia_inicial,
melhor_solucao, melhor_distancia_final, cidades

# Rodando o algoritmo genético
melhor_solucao_inicial, melhor_distancia_inicial, melhor_solucao,
melhor_distancia_final, cidades = algoritmo_genetico()

```

```

# Exibindo a primeira solução (inicial)
print("Primeira melhor solução encontrada (geração inicial):")
print("Caminho:", melhor_solucao_inicial)
print("Distância total:", melhor_distancia_inicial)

# Exibindo a melhor solução após 1000 gerações
print("Melhor solução após 1000 gerações:")
print("Caminho:", melhor_solucao)
print("Distância total:", melhor_distancia_final)

# Gerar gráfico 2D para a primeira solução (inicial)
x_inicial = [cidades[i][0] for i in melhor_solucao_inicial]
y_inicial = [cidades[i][1] for i in melhor_solucao_inicial]

# Ajustar o tamanho da figura
plt.figure(figsize=(16, 8)) # Figura maior

plt.subplot(1, 2, 1)
plt.scatter(x_inicial, y_inicial, color='green')
plt.plot(x_inicial, y_inicial, color='purple')

# Destacar o ponto inicial (vermelho) e o ponto final (azul)
plt.scatter([cidades[melhor_solucao_inicial[0]][0]],
            [cidades[melhor_solucao_inicial[0]][1]], color='red', s=100,
            label='Início', zorder=5)
plt.scatter([cidades[melhor_solucao_inicial[-1]][0]],
            [cidades[melhor_solucao_inicial[-1]][1]], color='blue', s=100,
            label='Final', zorder=5)

# Garantir que a linha volta à cidade inicial
x_inicial.append(cidades[melhor_solucao_inicial[0]][0]) # Adicionando a
cidade inicial no final
y_inicial.append(cidades[melhor_solucao_inicial[0]][1]) # Adicionando a
cidade inicial no final

# Destacar a última linha (do final de volta à cidade inicial)
plt.plot([cidades[melhor_solucao_inicial[-1]][0],
          cidades[melhor_solucao_inicial[0]][0]],
         [cidades[melhor_solucao_inicial[-1]][1],
          cidades[melhor_solucao_inicial[0]][1]],

```

```

        color='green', linewidth=2, label="Última Linha (Retorno)",
        zorder=5)

plt.title("Primeira Solução Encontrada")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")
plt.legend()

# Gerar gráfico 2D para a melhor solução final
x_final = [cidades[i][0] for i in melhor_solucao]
y_final = [cidades[i][1] for i in melhor_solucao]
plt.subplot(1, 2, 2)
plt.scatter(x_final, y_final, color='green')
plt.plot(x_final, y_final, color='purple')

# Destacar o ponto inicial (vermelho) e o ponto final (azul)
plt.scatter([cidades[melhor_solucao[0]][0]],
            [cidades[melhor_solucao[0]][1]], color='red', s=100, label='Início',
            zorder=5)
plt.scatter([cidades[melhor_solucao[-1]][0]], [cidades[melhor_solucao[
1]][1]], color='blue', s=100, label='Final', zorder=5)

# Garantir que a linha volta à cidade inicial
x_final.append(cidades[melhor_solucao[0]][0]) # Adicionando a cidade
inicial no final
y_final.append(cidades[melhor_solucao[0]][1]) # Adicionando a cidade
inicial no final

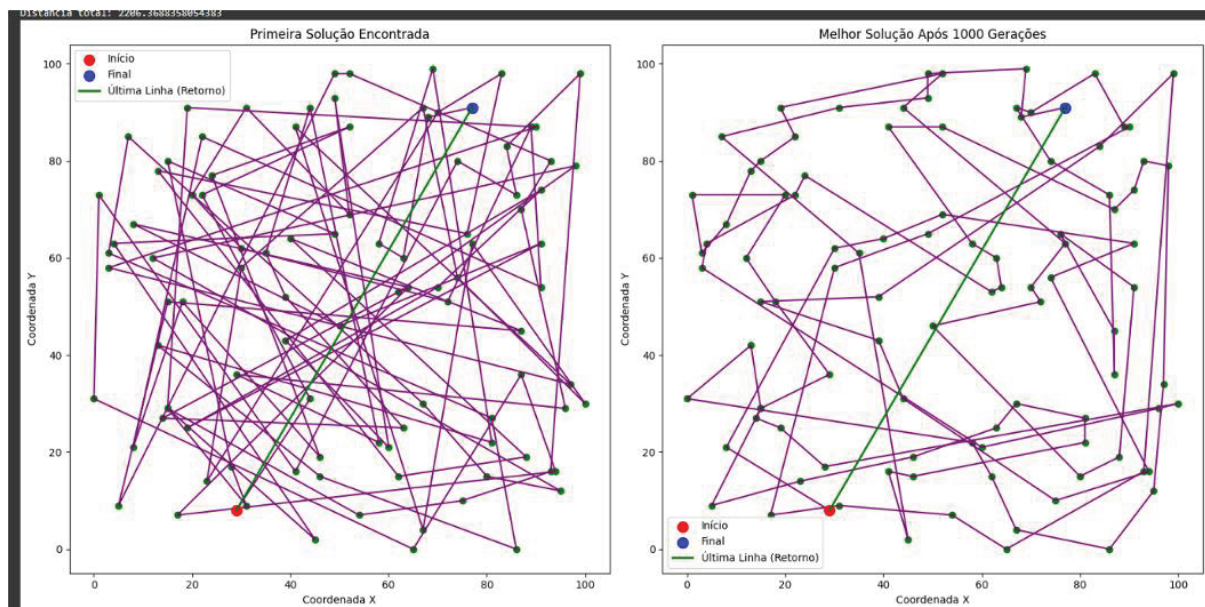
# Destacar a última linha (do final de volta à cidade inicial)
plt.plot([cidades[melhor_solucao[-1]][0], cidades[melhor_solucao[0]][0]],
        [cidades[melhor_solucao[-1]][1], cidades[melhor_solucao[0]][1]],
        color='green', linewidth=2, label="Última Linha (Retorno)",
        zorder=5)

plt.title("Melhor Solução Após 1000 Gerações")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")
plt.legend()

# Exibir os gráficos com espaçamento ajustado
plt.tight_layout()

```

```
plt.show()
```



2 Compare a representação de dois modelos vetoriais

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import StandardScaler
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import unicodedata
from nltk.stem import WordNetLemmatizer
# Baixando os recursos necessários do NLTK
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
# Inicializando a lista de stopwords em português
stopwords_portuguese = stopwords.words('portuguese')

# Inicializando o lematizador
lemmatizer = WordNetLemmatizer()

# Função para remover acentos das palavras
```

```

def remover_acentos(texto):
    return ''.join(c for c in unicodedata.normalize('NFD', texto) if
unicodedata.category(c) != 'Mn')

# Função para aplicar tokenização, remoção de stopwords e lematização
def processar_texto_com_lemmatizacao(texto):
    # Tokeniza o texto em palavras
    palavras_tokenizadas = word_tokenize(texto, language='portuguese')

    # Removendo stopwords e lematizando as palavras
    palavras_processadas = [
        lemmatizer.lemmatize(remover_acentos(palavra.lower()))
        for palavra in palavras_tokenizadas
        if palavra.lower() not in stopwords_portuguese and
palavra.isalpha()
    ]

    # Junta as palavras processadas de volta em uma frase
    return " ".join(palavras_processadas)

# Exemplo de texto com várias sentenças em português
texto = [
    "Eu adoro aprender novas linguagens de programação.",
    "Programar em Python é divertido e fácil.",
    "Análise de dados é uma ferramenta poderosa.",
    "Estou aprendendo como analisar dados de forma eficaz.",
    "Aprendizado de máquina pode ajudar a fazer previsões.",
    "Ciência de dados envolve extrair insights dos dados."
]

# Aplicando o pré-processamento (tokenização, remoção de stopwords e
lemmatização) em cada sentença
texto_processado = [processar_texto_com_lemmatizacao(frase) for frase in
texto]

print('TEXTO PROCESSADO COM LEMMATIZAÇÃO: ', texto_processado)

# 1. Vetorização de sentenças com TF-IDF (sem stopwords específicas)
vectorizer = TfidfVectorizer(stop_words=None) # Não estamos usando
stopwords para português
X = vectorizer.fit_transform(texto_processado)

```

```

# Pegando as sentenças que foram vetorizadas
sentencas = texto # Mantemos as sentenças originais

# Convertendo para um formato denso para aplicar o PCA
X_dense = X.toarray() # Usamos as sentenças diretamente

# 2. Redução de Dimensionalidade com PCA
# Padronizando os dados antes de aplicar o PCA
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_dense)

# Aplicando PCA para redução de dimensionalidade para 2 componentes
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# 3. Visualização dos dados projetados no gráfico
plt.figure(figsize=(12, 8)) # Aumentando o tamanho da figura
plt.scatter(X_pca[:, 0], X_pca[:, 1], color='blue')

# Ajustando os limites dos eixos (para garantir que todos os pontos
caibam)
plt.xlim(X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 7)
plt.ylim(X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1)

# Adicionando os rótulos das sentenças no gráfico de forma mais flexível
com annotate
for i, sentenca in enumerate(sentencas):
    plt.annotate(sentenca,
                  (X_pca[i, 0], X_pca[i, 1]),
                  fontsize=10,
                  textcoords="offset points",
                  xytext=(0, 10))
# Título e rótulos
plt.title("Projeção das Sentenças usando PCA")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
# Exibindo a grade e o gráfico
plt.grid(True)
plt.show()

```

