

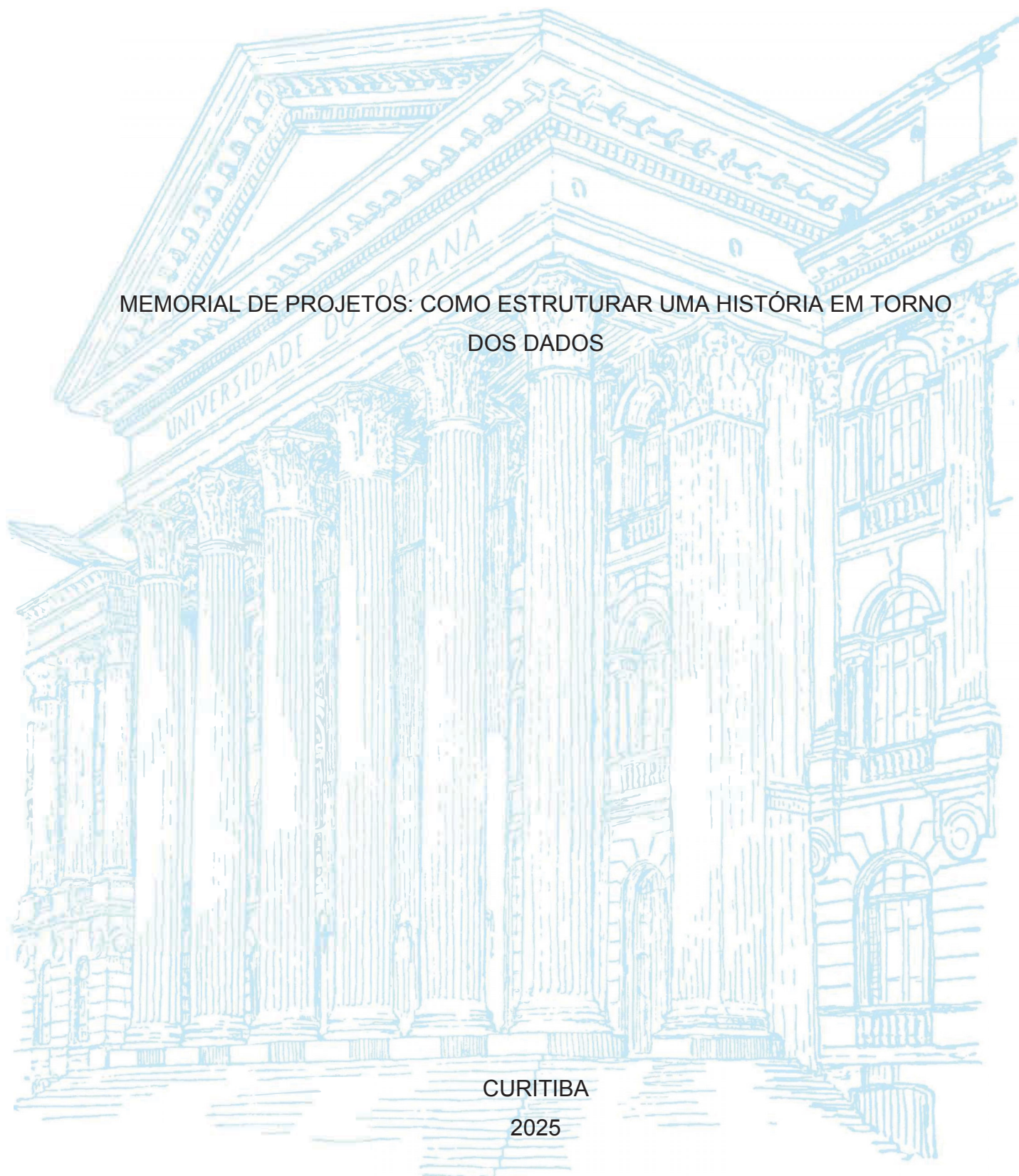
UNIVERSIDADE FEDERAL DO PARANÁ

VINÍCIUS LUÍS TEDESCO

MEMORIAL DE PROJETOS: COMO ESTRUTURAR UMA HISTÓRIA EM TORNO
DOS DADOS

CURITIBA

2025



VINÍCIUS LUÍS TEDESCO

MEMORIAL DE PROJETOS: COMO ESTRUTURAR UMA HISTÓRIA EM TORNO
DOS DADOS

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

Orientadora: Profa. Dra. Rafaela Mantovani Fontana

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **VINÍCIUS LUÍS TEDESCO**, intitulada: **MEMORIAL DE PROJETOS: COMO ESTRUTURAR UMA HISTÓRIA EM TORNO DOS DADOS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua **aprovação** no rito de defesa. A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 04 de Setembro de 2025.

RESUMO

A inteligência artificial (IA) tem modificado de forma significativa como lidamos com dados e processos decisórios, exigindo não apenas informações de qualidade, mas também uma organização que beneficie a compreensão e o aprendizado das máquinas. Deste modo, a estruturação narrativa dos dados, ou *storytelling* com esses dados, salienta como estratégia capaz de agregar contexto, temporalidade e associações ao conjunto de informações, tornando-os mais ricos e interpretáveis. Posto que tradicionalmente aplicada na comunicação visual e apresentação de resultados, essa abordagem também contribui diretamente para a construção de modelos de IA mais precisos, transparentes e éticos. Oferecer aos algoritmos uma sequência lógica de eventos permite melhorar a detecção de padrões, reduzir vieses e aumentar a confiabilidade das previsões, especialmente em áreas críticas. Desta forma, integrar o *storytelling* na modelagem de dados, não apenas aprimora o desempenho técnico, mas aumenta a confiança social na tecnologia e sua aplicação na realidade.

Palavras-chave: Inteligência Artificial; Storytelling com Dados; Estruturação Narrativa; Qualidade de Dados; Modelagem Preditiva.

ABSTRACT

Artificial Intelligence (AI) has significantly transformed the way we handle data and decision-making processes, requiring not only high-quality information but also an organization that enhances machine understanding and learning. In this context, narrative structuring of data, or storytelling with data, emerges as a strategy capable of adding context, temporality, and associations to datasets, making them richer and more interpretable. While traditionally applied to visual communication and results presentation, this approach also directly contributes to building AI models that are more accurate, transparent, and ethical. By providing algorithms with a logical sequence of events, it becomes possible to improve pattern detection, mitigate biases, and increase the reliability of predictions, especially in critical domains. Thus, integrating storytelling into data modeling not only enhances technical performance but also strengthens public trust in technology and its real-world applications.

Keywords: Artificial Intelligence; Data Storytelling; Narrative Structuring; Data Quality; Predictive Modeling.

SUMÁRIO

1 PARECER TÉCNICO.....	7
REFERÊNCIAS.....	9
APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL.....	10
APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA.....	19
APÊNDICE 3 – LINGUAGEM R.....	32
APÊNDICE 4 – ESTATÍSTICA APLICADA I.....	39
APÊNDICE 5 – ESTATÍSTICA APLICADA II.....	46
APÊNDICE 6 – ARQUITETURA DE DADOS.....	48
APÊNDICE 7 – APRENDIZADO DE MÁQUINA.....	62
APÊNDICE 8 – DEEP LEARNING.....	68
APÊNDICE 9 – BIG DATA.....	121
APÊNDICE 10 – VISÃO COMPUTACIONAL.....	127
APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	130
APÊNDICE 12 – GESTÃO DE PROJETOS DE IA.....	144
APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	148
APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING.....	172
APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	179

1 PARECER TÉCNICO

Com o grande avanço dos sistemas de inteligência artificial (IA), nos últimos anos tem se transformado radicalmente o modo em que lidamos com grandes volumes de dados e processos decisórios automatizados. Por outro lado, o desempenho e a credibilidade desses sistemas estão extremamente atrelados à qualidade e à estrutura dos dados que os sustentam. A interpretação e o pré-processamento de dados não devem se limitar somente à remoção de ruídos e inconsistências, mas também considerar a estruturação da narrativa dos dados como um passo estratégico que garante que a IA compreenda, aprenda e gere resultados mais eficazes e contextualizados (Singh, Kumar, & Gupta, 2025; Zhao et al., 2025; NVIDIA, 2023).

Dentre essas abordagens provenientes do tratamento e curadoria dos dados, a estruturação narrativa, também conhecida como *storytelling* com dados, vem se tornando destaque por contribuir não somente com a comunicação de resultados, mas de certa forma na própria construção e desempenho dos sistemas de IA. Tradicionalmente, o *storytelling* com dados é aplicado no contexto da visualização e apresentação de informações, com o objetivo de tornar gráficos e relatórios mais compreensíveis, persuasivos e acessíveis a públicos diversos (Knafllic, 2015; Dykes, 2020). Por outro lado, ao considerar que os modelos de IA aprendem padrões com base nos dados de entrada, sendo que uma estrutura narrativa pode ajudar, captando relações causais e temporais que dados desorganizados muitas vezes ocultam.

Considerando um exemplo, do qual um modelo de IA destinado a prever o risco de um paciente retornar ao hospital. Um *dataset* convencional carregaria variáveis isoladas, como tempo de internação, idade e comorbidades. De outro modo, um *dataset* estruturado de forma narrativa conforme exemplificado no Quadro 1 incluiria o histórico clínico do paciente, com dados como o diagnóstico inicial, evolução do quadro, tratamentos aplicados e tempos de resposta. Essa organização oferece à IA uma “história” que permite associar causas e consequências, construindo uma base mais sólida para previsões confiáveis.

Além disso, a estruturação narrativa de dados é fundamental em contextos sensíveis, como por exemplo finanças, justiça e saúde, sendo que nesses meios as consequências de erros algoritmos são graves. Pensar os dados como histórias

permite detectar lacunas, distorções e vieses que poderiam comprometer a equidade das decisões automatizadas (Shneiderman, 2020).

QUADRO 1 - Exemplo prático - *Dataset* com estrutura narrativa

IdPaciente	Data Admissão	Diagnóstico	Histórico	Tratamento	Eventos Adversos	Alta	Re Internado
01	15/01/2023	Insfu. Cardíaca	Hipertensão Tabagismo	Diuréticos	Sim	20/01/2023	Não
02	03/02/2023	Pneumonia	Diabetes tipo 2	Antibióticos	Não	09/02/2023	Sim

FONTE: O autor(2025)

Esse tipo de estrutura revela não apenas variáveis, mas trajetórias, e é exatamente essa lógica que a inteligência artificial precisa aprender para desta maneira oferecer respostas mais coerentes à realidade. Com aponta Miller(2019) e Doshi-Velez e Kim(2017), a explicabilidade dos modelos depende diretamente de como os dados são representados. A narrativa não serve apenas para humanos entenderem a saída da IA, mas também para que a IA aprenda de maneira mais próxima ao raciocínio humano.

Deste modo, ao estruturar uma história em torno dos dados, cria-se um ambiente de aprendizado mais rico, ético e transparente para a IA. Sendo que essa prática amplia não apenas o desempenho técnico dos modelos, mas de certa forma também a confiança social na tecnologia.

REFERÊNCIAS

DOSHI-VELEZ, Finale; KIM, Been. Towards a rigorous science of interpretable machine learning. *arXiv preprint* arXiv:1702.08608, 2017. Disponível em: <https://arxiv.org/abs/1702.08608>

DYKES, Brent. *Effective Data Storytelling: How to Drive Change with Data, Narrative and Visuals*. Wiley, 2020.

KNAFLIC, Cole Nussbaumer. *Storytelling with data: A data visualization guide for business professionals*. Wiley, 2015.

MILLER, Tim. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, v. 267, p. 1–38, 2019. DOI: <https://doi.org/10.1016/j.artint.2018.07.007>

NVIDIA. *Data Storytelling Best Practices for Data Scientists and AI Practitioners*. NVIDIA Developer Blog, 2023. Disponível em: <https://developer.nvidia.com/blog/data-storytelling-best-practices-for-data-scientists-and-ai-practitioners>. Acesso em: 22 ago. 2025.

SHNEIDERMAN, Ben. Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy. *International Journal of Human–Computer Interaction*, v. 36, n. 6, p. 495–504, 2020. DOI: <https://doi.org/10.1080/10447318.2020.1741118>

SINGH, A.; KUMAR, R.; GUPTA, M. *Data Narratives Using AI: A Framework for Automated Insight Storytelling*. *International Journal of Scientific Research in Engineering and Technology*, 2025. Disponível em: <https://ijsret.com/2025/01/03/data-narratives-using-ai-a-framework-for-automated-insight-storytelling>. Acesso em: 22 ago. 2025.

ZHAO, Y. et al. *MDSF: Context-Aware Multi-Dimensional Data Storytelling Framework based on Large language Model*. *arXiv preprint*, 2025. Disponível em: <https://arxiv.org/abs/2501.01014>. Acesso em: 22 ago. 2025

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.

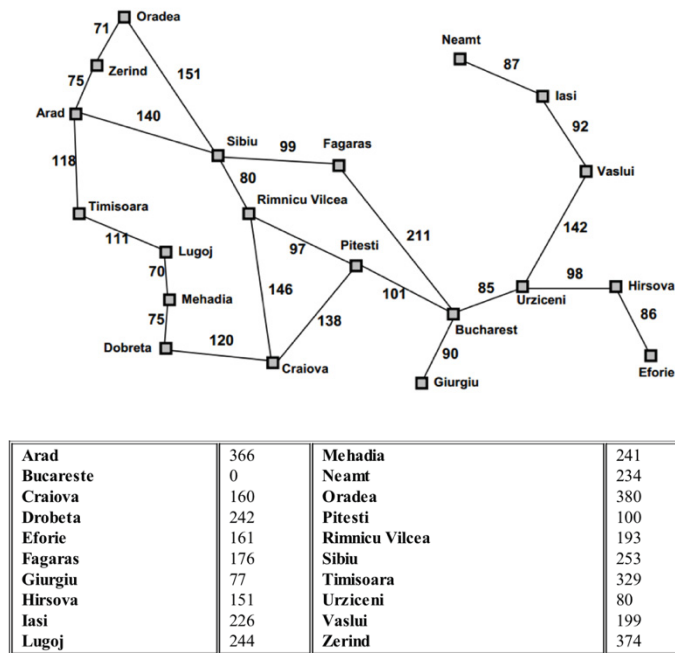


Figura 3.22 Valores de $hDLR$ — distâncias em linha reta para Bucarest.

3 Lógica

Verificar se o argumento lógico é válido.

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

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem

q: a raposa come as uvas

r: as uvas estão maduras

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

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

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

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

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

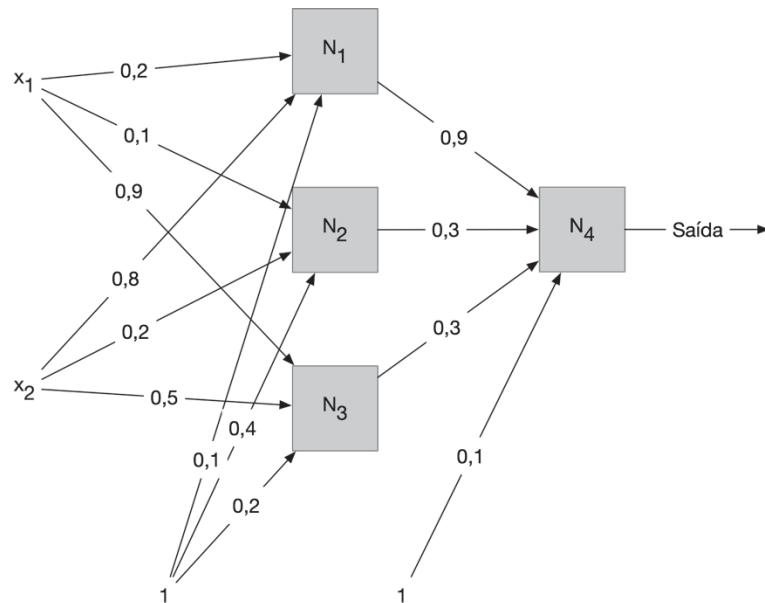
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada $x_1=-3, x_2=1$, dê:

- (6,25 pontos)** Valor de saída do neurônio N_1
- (6,25 pontos)** Valor de saída do neurônio N_2
- (6,25 pontos)** Valor de saída do neurônio N_3
- (6,25 pontos)** Valor de saída da rede como um todo

B – RESOLUÇÃO

1 - ChatGPT

a) Inteligência artificial (IA) refere-se à capacidade de um sistema computacional para realizar tarefas que normalmente requerem intervenção humana. Isso inclui coleta e análise de dados, aprendizado, reconhecimento de padrões, tomada de decisões e interação com o ambiente de uma maneira semelhante à dos seres humanos. A IA possibilita que os sistemas aprendam e se adaptem por meio da experiência, aprimorando continuamente suas capacidades.

b) A definição de IA do ChatGPT está mais alinhada com a abordagem do comportamento fiel ao comportamento humano. O ChatGPT define IA como a capacidade de agir como os Humanos, ou seja, a capacidade de realizar tarefas que

geralmente são realizadas por humanos, não explorando a questão do pensamento ou da racionalidade no sentido de buscar a solução ótima ou pensar racionalmente.

c) O ChatGPT é um chat bot alimentado por inteligência artificial que interage com seres humanos e permite obter respostas em linguagem natural a partir de comandos escritos. Ele usa técnicas avançadas de processamento de linguagem natural para entender perguntas e dar respostas aplicando um modelo de machine learning chamado de “aprendizado pré-treinado”. As limitações do ChatGPT são as mesmas de um ser humano, com a diferença de que o ser humano, além de pesquisar conteúdo para responder uma pergunta, utiliza conhecimentos pré concebidos e informações armazenadas para responder uma pergunta. O ChatGPT não tem conhecimento prévio sobre domínio ou tópicos específicos, como um humano especialista em um assunto, por exemplo.

As limitações são as mesmas de um ser humano:

- 1 – Falta de conhecimento específico, como mencionado acima;
- 2 – Possibilidade de respostas erradas;
- 3 – Sensibilidade ao contexto, o que pode levar o ChatGPT a ter dificuldade em captar o contexto ou perder a coerência na continuidade da conversa;
- 4 – Respostas vagas ou evasivas;
- 5 – Propensão a viés e linguagem inadequada.

De certa forma seu funcionamento para lidar com todos esses dados, trabalha com uma arquitetura conhecida como Transformer, introduzida em 2017, a qual se destaca pela sua capacidade de lidar com sequências de palavras de maneira eficaz.

ChatGPT é um Chat Bot criado pela OpenAI. Um chatbot é uma ferramenta de processamento de linguagem natural. Ele faz uso dos LLMs (Large Language Model) GPT3.5 e GPT4. O LLM é um tipo rede neural, a maneira como ela é construída e através de uma quantidade enorme de textos que é apresentada a rede para que esta entenda como as palavras se relacionam, criando assim padrões que são então usados para gerar textos. No entanto, os textos gerados após essa etapa inicial de treinamento nem sempre garante resultados da melhor qualidade, o que é

feito então é um refinamento dos modelos usados pela rede, esse refinamento é realizado por humanos em um processo que recebeu o nome de "Reinforcement Learning from Human Feedback". Esse método tem suas vantagens quando comparado a outros modelos de treinamento em que são usadas textos e imagens rotuladas para treinar o LLM, principalmente porque as bases com informações rotuladas não costumavam ser grandes o suficiente, pois o processo de criação de bases era caro.

Dentre as limitações da tecnologia, podemos citar

1 - Capacidade de entender contextos: muitas vezes a ferramenta falha ao entender nuances ou tem problemas ao se deparar com questões complexas e gera respostas superficiais que não atendem a demanda;

2 - Informação defasada: o modelo foi treinado usando informações até uma certa data e ao lidar com questões relacionadas a assuntos mais recentes pode gerar respostas imprecisas;

3 - Criatividade: as respostas geradas pela ferramenta não costumam ser criativas pois se baseiam em conteúdos fornecidos previamente;

4 - Falta de senso comum: estas são máquinas que não são normais, pois estão usando recursos de máquina local;

5 - Respostas Direcionadas, sendo que essas respostas são recursos que produzem vieses locais como por exemplo a língua inglesa direcionada e outros tipos de artigos;

6 - Conhecimento limitado em certos tópicos: imagine tentar buscar aprender somente diferentes tópicos e nichos, isso pode ser complicado, mas houve uma época que o ChatGPT tentou fazer isso e pode ser complicado por causa da quantidade de conteúdo;

7 - Respostas incompletas e imprecisas: um problema pode ser a geração de respostas imprecisas, isso decorre do fato de que as respostas nem sempre trabalham com a mesma base de conhecimento, por exemplo, quando responder usam como base um conjunto de informação que é incompleto, ou que gera umas respostas parciais ou respostas que podem estar defasadas ou até em determinados momentos, não correspondem a nossa realidade;

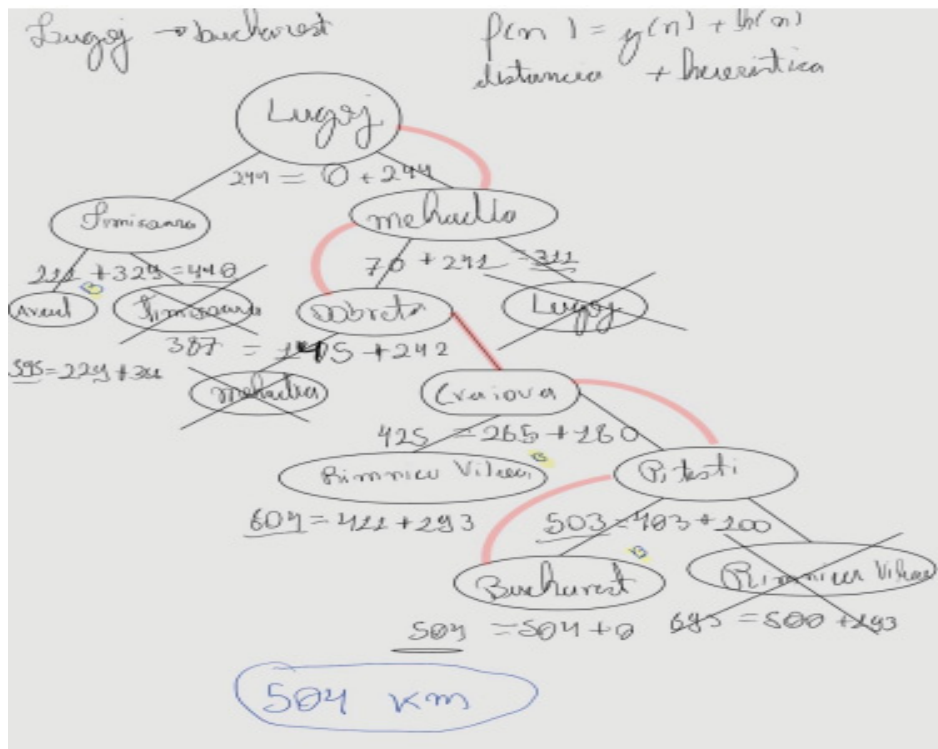
Raymond, Daniel. Top 10 Cons & Disadvantages of ChatGPT. Disponível em 10/03/2024 em: <https://projectmanagers.net/top-10-cons-disadvantages-of-chatgpt/>

Guinness, Harry. How does ChatGPT work? . Disponível em 10/03/2024 em: <https://zapier.com/blog/how-does-chatgpt-work/>

Ramponi, Marco. How ChatGPT actually works. Disponível em 10/03/2024 em: <https://www.assemblyai.com/blog/how-chatgpt-actually-works/>

d) Predominantemente o ChatGPT tenta agir como os humanos, apresentando várias características do comportamento humano, como as limitações do ser humano ao responder uma pergunta. O Chat GPT também se enquadra na abordagem descrita como agir racionalmente, dentro dos parâmetros do seu treinamento (educação) e da sua programação, ele foi projetado para compreender e gerar respostas úteis. Envolvendo entender o contexto, inferir a intenção e buscar fornecer respostas corretas e até mesmo informativa.

2 – Busca Heurística



3 – Lógica

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

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

Logo

$q \leftrightarrow p$

Prova:

3)

$R_1: p \rightarrow q$	
$R_2: q \rightarrow r$	Derivação de $q \leftrightarrow p$
$R_3: \neg r \vee p$	
$R_4: R \rightarrow P$	
$R_5: q \rightarrow P$	Silog. hipot.; R_2, R_4
$R_6: (q \rightarrow P) \wedge (P \rightarrow q)$	Conjunção; R_5, R_1
$R_7: q \leftrightarrow P$	Eg. Bicondicional; R_6

(Handwritten signature/initials over R7)

4 - Redes Neurais

$x_1 = -3, x_2 = 1$

a) $N_1 = \text{linear}(u)$

$$u = 1 * 0,1 + (-3) * 0,2 + 1 * 0,8$$

$$u = 0,1 - 0,6 + 0,8$$

$$u = 0,3$$

$$N1 = \text{linear}(0,3)$$

$$\mathbf{N1 = 0,3}$$

$$\mathbf{b) N2 = \text{linear}(u)}$$

$$u = 1 * 0,4 + (-3) * 0,1 + 1 * 0,2$$

$$u = 0,4 - 0,3 + 0,2$$

$$u = 0,3$$

$$N2 = \text{linear}(0,3)$$

$$N2 = 0,3$$

$$\mathbf{c) N3 = \text{linear}(u)}$$

$$u = 1 * 0,2 + (-3) * 0,9 + 1 * 0,5$$

$$u = 0,2 - 2,7 + 0,5$$

$$u = -2$$

$$N3 = \text{linear}(-2)$$

$$N3 = -2$$

$$\mathbf{d) N4 = \tanh(u)}$$

$$u = 1 * 0,1 + N1 * 0,9 + N2 * 0,3 + N3 * 0,3$$

$$u = 1 * 0,1 + 0,3 * 0,9 + 0,3 * 0,3 + (-2) * 0,3$$

$$u = -0,14$$

$$N4 = \tanh(-0,14)$$

$$\text{Saída} = -0,1391$$

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

Nome do campo	Descrição
year_of_reference	O preço médio corresponde a um mês de ano de referência
month_of_reference	O preço médio corresponde a um mês de referência, ou seja, a FIPE atualiza sua tabela mensalmente
fipe_code	Código único da FIPE
authentication	Código de autenticação único para consulta no site da FIPE
brand	Marca do carro
model	Modelo do carro

fuel	Tipo de combustível do carro
gear	Tipo de engrenagem do carro
engine_size	Tamanho do motor em centímetros cúbicos
year_model	Ano do modelo do carro. Pode não corresponder ao ano de fabricação
avg_price	Preço médio do carro, em reais

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

1 Análise Exploratória dos dados

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

- 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

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

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

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

- a. Escolha as variáveis **numéricas** (modelos de Regressão) para serem as variáveis independentes do modelo. A variável target é **avg_price**. **Observação:** caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no modelo. Indique **quais variáveis** foram transformadas e **como** foram transformadas
- b. Crie partições contendo 75% dos dados para treino e 25% para teste
- c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros. **Observação:** caso julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo
- d. Grave os valores preditos em variáveis criadas
- e. Realize a análise de importância das variáveis para estimar a variável target, **para cada modelo treinado**
- f. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis
- g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2
- h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

B - RESOLUÇÃO

1 - Análise Exploratória dos dados

a)

✓ 1 - Análise Exploratória dos dados

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

✓ a. Carregue a base de dados `media_precos_carros_brasil.csv`

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.impute import SimpleImputer
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

#Carregando a base de dados.

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

b)

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

```
[2] # Verificando a existencia de elementos sem valor nenhum
dados.isna().sum()

# Metodo para remover linhas que contenham linhas com valores faltantes

# A função dropna() apaga as linhas que contem entradas com dados "não disponíveis"
dados = dados.dropna()
```

```
dados.head()
```

	year_of_reference	month_of_reference	fiipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
0	2021.0	January	004001-0	cfzktzhercp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2002.0	9162.0
1	2021.0	January	004001-0	cdgxxwpe3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2001.0	8832.0
2	2021.0	January	004001-0	cb1t3xwejt1p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000.0	8388.0
3	2021.0	January	004001-0	ct8gctf65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000.0	8453.0
4	2021.0	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL / Chevro 1.6 MBEI / EFI	Gasoline	manual	1.6	2001.0	12525.0

✓ c. Verifique se há dados duplicados nos dados

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

2

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

```
# Criando duas categorias para armazenar as colunas categoricas e numericas

numeric_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']
```

```
# Imprimindo o resumo de informações das variáveis numéricas
dados[numeric_cols].describe()
```

	year_of_reference	year_model	avg_price_brl
count	202297.000000	202297.000000	202297.000000
mean	2021.564694	2011.271527	52756.909153
std	0.571903	6.378234	51628.677716
min	2021.000000	2000.000000	6647.000000
25%	2021.000000	2006.000000	22855.000000
50%	2022.000000	2012.000000	38027.000000
75%	2022.000000	2016.000000	64064.000000
max	2023.000000	2023.000000	979358.000000

```
[7] # Imprimindo o resumo de informações das variáveis categóricas
dados[categoricas_cols].describe()
```

	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size
count	202297	202297	202297	202297	202297	202297	202297	202297
unique	12	2001	202296	6	2112	3	2	20
top	January	003281-6	3r6c277cnqcb	Fiat	Palo Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6
freq	24260	425	2	44962	425	168685	161885	47420

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

```
[8] # Imprimindo a contagem de valores - Modelo
dados['model'].value_counts()
```

```
Palo Week. Adv/Adv TRYON 1.8 mpi Flex    425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p    425
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.    400
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V      400
Corvette 5.7/ 6.0, 6.2 Targa/Stingray    375
...
STEPWAY Zen Flex 1.0 12V Mec.              2
Saveiro Robust 1.6 Total Flex 16V CD       2
Saveiro Robust 1.6 Total Flex 16V         2
Gol Last Edition 1.0 Flex 12V 5p          2
Polo Track 1.0 Flex 12V 5p                2
Name: model, Length: 2112, dtype: int64
```

```
[9] # Imprimindo a contagem de valores - Marca
dados['brand'].value_counts()
```

```
Fiat                44962
VW - Volkswagen     44312
GM - Chevrolet      38590
Ford                33151
Renault             29192
Nissan               12090
Name: brand, dtype: int64
```

f) Dentro dos resultados obtidos na Análise Exploratória, podemos dizer que a base de dados original possui 11 colunas e 202297 dados, destes, 65245 são valores que possuem dados faltantes. Depois do tratamento para eliminar valores faltantes obtivemos uma quantidade total de 137052 registros. Podemos constatar também que após os tratamentos, há 2 dados duplicados. O modelo de carro que aparece com mais frequência é um empate entre o "Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p" e o "Palio Week. Adv/Adv TRYON 1.8 mpi Flex" que aparecem ambos 425 vezes, ao passo que a marca que mais aparece é a Fiat com 44962 ocorrências.

2 - Visualização dos dados

2 Visualização dos dados

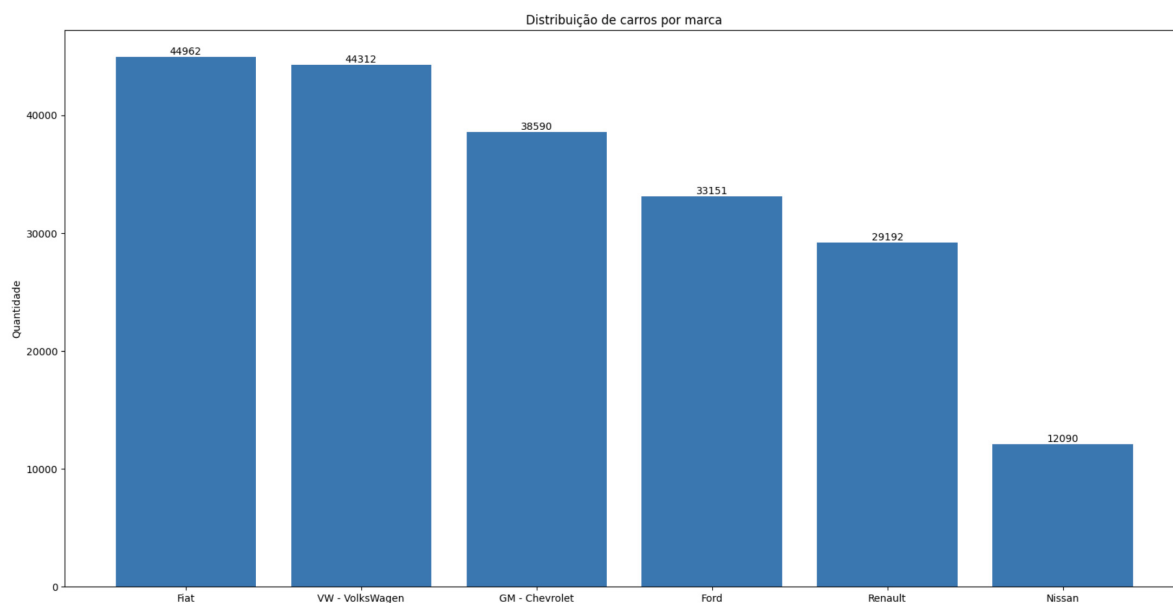
A partir da base de dados `precos_carros_brasil.csv`, execute as seguintes tarefas:

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

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

plt.figure(figsize=(20,10))
grafico_1 = plt.bar(valores_contados.index, valores_contados.values)
plt.title('Distribuição de carros por marca')
plt.ylabel('Quantidade')
plt.bar_label(grafico_1, size=10)

[Text(0, 0, '44962'),
 Text(0, 0, '44312'),
 Text(0, 0, '38590'),
 Text(0, 0, '33151'),
 Text(0, 0, '29192'),
 Text(0, 0, '12090')]
```

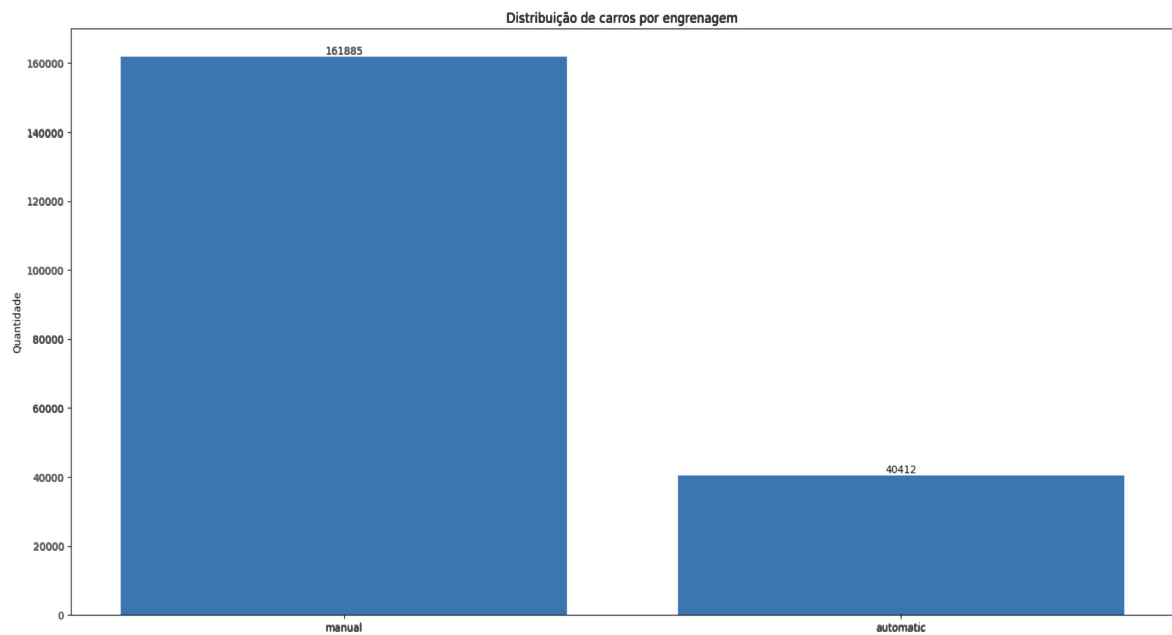


✓ b. Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro

```
valores_contados_2 = dados['gear'].value_counts()

plt.figure(figsize=(20,10))
grafico_2 = plt.bar(valores_contados_2.index, valores_contados_2.values)
plt.title('Distribuição de carros por engrenagem')
plt.ylabel('Quantidade')
plt.bar_label(grafico_2, size=10)
```

```
[Text(0, 0, '161885'), Text(0, 0, '40412')]
```

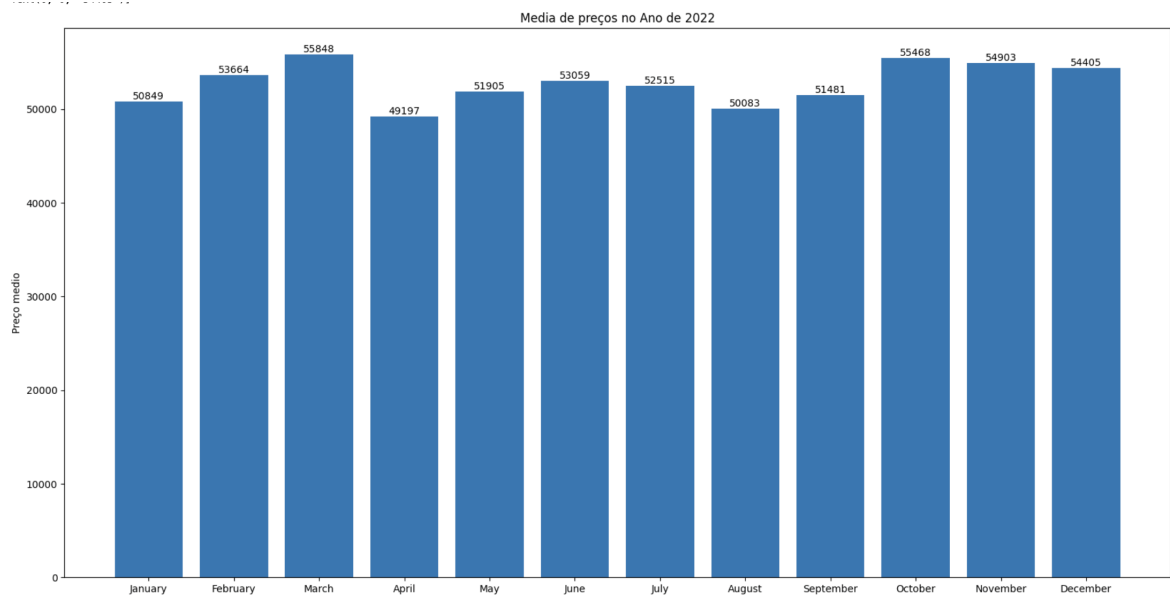


✓ c. Gere um gráfico da evolução da média de preço dos carros ao longo dos meses de 2022 (variável de tempo no eixo X)

```
ano_2022 = dados[dados.year_of_reference == 2022]

plt.figure(figsize=(20,10))
grafico_3 = plt.bar(ano_2022['month_of_reference'].unique(), dados.groupby('month_of_reference')['avg_price_brl'].mean().round())
plt.ylabel('Preço médio')
plt.title('Média de preços no Ano de 2022')
plt.bar_label(grafico_3, size=10)
```

```
Text(0, 0, '49197'),
Text(0, 0, '51905'),
Text(0, 0, '53059'),
Text(0, 0, '52515'),
Text(0, 0, '50083'),
Text(0, 0, '51481'),
Text(0, 0, '55468'),
Text(0, 0, '54983'),
Text(0, 0, '54405')]
```

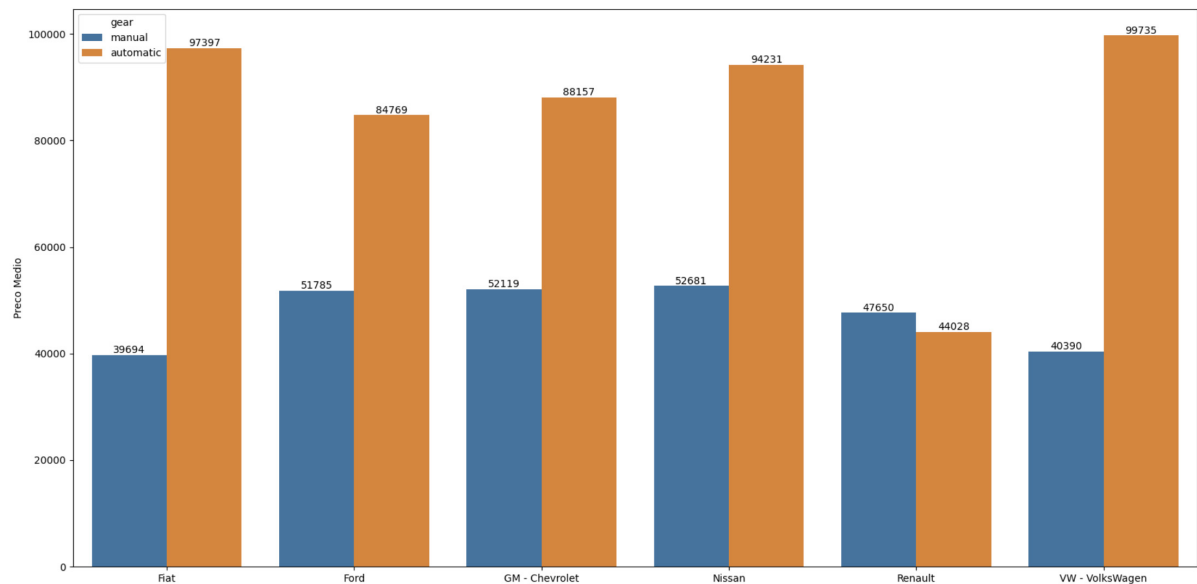


✓ d. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem

```
media_marca_engrenagem = dados.groupby(['brand', 'gear'])['avg_price_brl'].mean().round()
media_marca_engrenagem = media_marca_engrenagem.reset_index(name='Preço Médio')

plt.figure(figsize=(20,10))
ax = sns.barplot(data=media_marca_engrenagem, x='brand', y='Preço Médio', hue='gear', hue_order=['manual', 'automatic'])
ax.bar_label(ax.containers[0], size=10)
ax.bar_label(ax.containers[1], size=10)
```

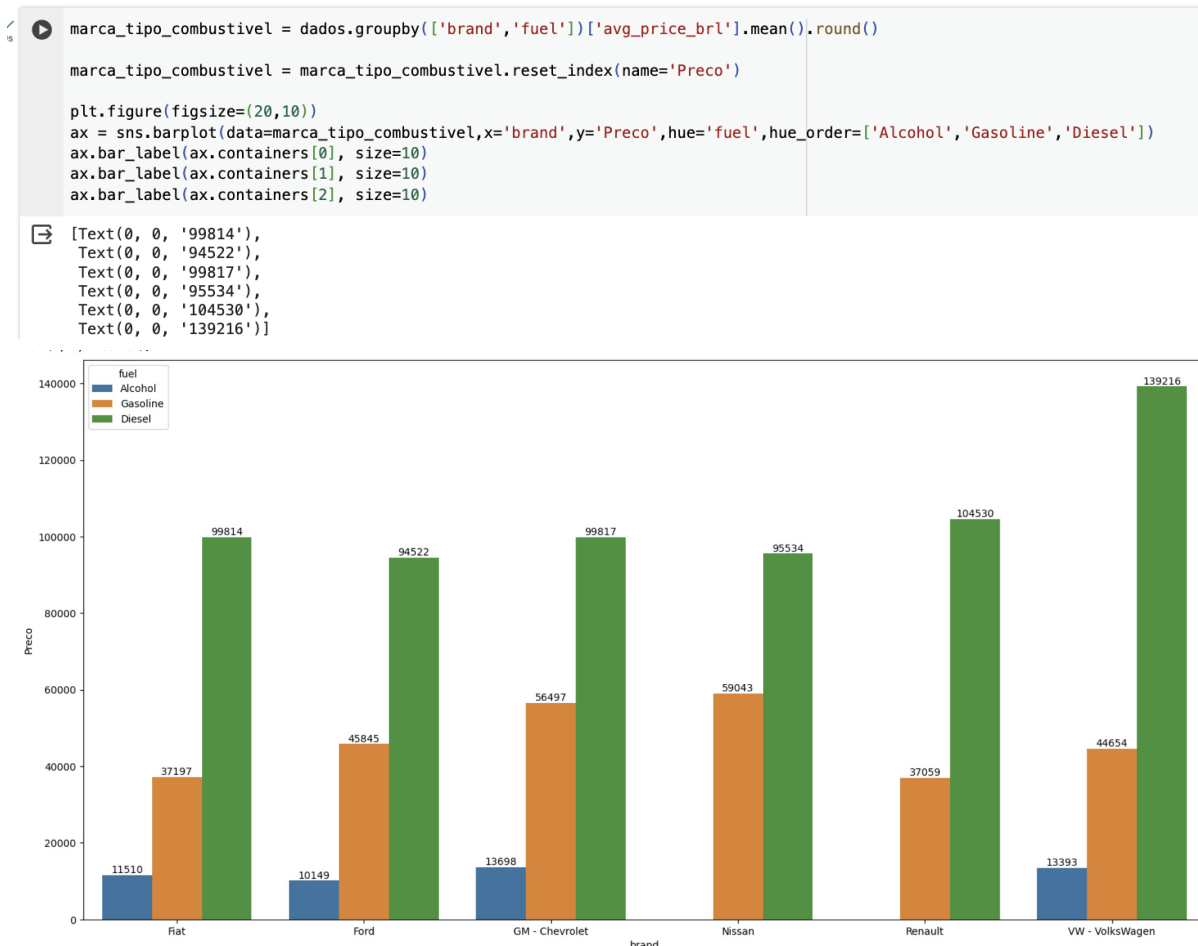
```
[Text(0, 0, '97397'),
Text(0, 0, '84769'),
Text(0, 0, '88157'),
Text(0, 0, '94231'),
Text(0, 0, '44028'),
Text(0, 0, '99735')]
```



e) No gráfico gerado no item d podemos verificar que todas as marcas oferecem tanto modelos de carros com câmbios (engrenagens) manuais como automáticas. Podemos também concluir que os carros com câmbio automáticos tem um preço muito maior que o de carros com engrenagens manuais, em alguns casos,

como o exemplo da Fiat e da Volkswagen, essa média de preço do carro automático para o manual, pode chegar a ser o dobro do valor.

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



g) O que o gráfico gerado no item f pode nos informar é que o valor dos carros que utilizam o Diesel como combustível, em média, são os mais caros. Aqueles que possuem o segundo valor médio seriam os carros a gasolina, com os carros a álcool no terceiro e último lugar, sendo que existem marcas, como o caso da Nissan e da Renault, que de acordo com a base de dados nem chega a fornecer uma opção de carro a álcool.

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

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

Aqui será feita a alteração das variáveis da coluna 'gear', elas passarão a ser do tipo numérico com valores variando entre 0 (automatic) e 1 (manual).

```
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])
dados.tail()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
202292	2023.0	January	005538-7	ccv3mvmxsz0dqw	VW - VolksWagen	Saveiro Robust 1.6 Total Flex 16V	Gasoline	1	1,6	2023.0	86038.0
202293	2023.0	January	005539-5	chmwlg3l5hbp	VW - VolksWagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	95997.0
202294	2023.0	January	005539-5	cdj27srcvcodqw	VW - VolksWagen	Gol Last Edition 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	87828.0
202295	2023.0	January	005540-9	9w64fg6dhpq	VW - VolksWagen	Polo Track 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	80845.0
202296	2023.0	January	005540-9	7hbnjmj9z5dqw	VW - VolksWagen	Polo Track 1.0 Flex 12V 5p	Gasoline	1	1	2023.0	74458.0

Aqui será feita a alteração das variáveis da coluna 'fuel', elas passarão a ser do tipo numérico com valores variando entre 0 (Alcohol), 1 (Diesel) e 2 (Gasoline).

```
[16] dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])
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	cfzlcztzwrp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2002.0	9162.0
1	2021.0	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2001.0	8832.0
2	2021.0	January	004001-0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2000.0	8388.0
3	2021.0	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	0	1	1	2000.0	8453.0
4	2021.0	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	2	1	1,6	2001.0	12525.0

Aqui será feita a alteração das variáveis da coluna 'brand', elas passarão a ser do tipo numérico com um numero limitado de opções. A marcas assumirão valores: 0 (Fiat), 1 (Ford), 2 (GM), 3 (Nissan), 4 (Renault) e 5 (VW).

```
[17] dados['brand'] = LabelEncoder().fit_transform(dados['brand'])
```

Aqui é feito a alteração da coluna 'engine_size', o primeiro passo é alterar o caracter ',' pelo caracter '.', pois o metodo que faz a conversão de texto para valores numericos não soube lidar com a maneira como o Português lida com casas decimais, pois isso, é necessário alterar o caracter para que o Python entenda que se trata de um numero decimal. Em seguida o metodo 'pandas.to_numeric()' é usado para converter os valores da coluna 'engine_size'.

```
[18] dados['engine_size'] = dados['engine_size'].str.replace(',', '.')
dados['engine_size - numeric'] = pd.to_numeric(dados['engine_size'])
dados.head()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl	engine_size - numeric
0	2021.0	January	004001-0	cfzlcztzwrp	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2002.0	9162.0	1.0
1	2021.0	January	004001-0	cdqwxwpw3y2p	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2001.0	8832.0	1.0
2	2021.0	January	004001-0	cb1t3xwwj1xp	2	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1	2000.0	8388.0	1.0
3	2021.0	January	004001-0	cb9gct6j65r0	2	Corsa Wind 1.0 MPFI / EFI 2p	0	1	1	2000.0	8453.0	1.0
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	1.6

Aqui é criado um novo dataset utilizando somente os valores que serão utilizados no treinamento dos modelos.

```
dados_num = dados.drop(['month_of_reference', 'fipe_code', 'authentication', 'model', 'engine_size'], axis=1)
dados_num
```

	year_of_reference	brand	fuel	gear	year_model	avg_price_brl	engine_size - numeric
0	2021.0	2	2	1	2002.0	9162.0	1.0
1	2021.0	2	2	1	2001.0	8832.0	1.0
2	2021.0	2	2	1	2000.0	8388.0	1.0
3	2021.0	2	0	1	2000.0	8453.0	1.0
4	2021.0	2	2	1	2001.0	12525.0	1.6
...	...	...	...	...	...	...	...
202292	2023.0	5	2	1	2023.0	86038.0	1.6
202293	2023.0	5	2	1	2023.0	95997.0	1.0
202294	2023.0	5	2	1	2023.0	87828.0	1.0
202295	2023.0	5	2	1	2023.0	80845.0	1.0
202296	2023.0	5	2	1	2023.0	74458.0	1.0

202297 rows x 7 columns

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

Aqui eu vou definir as variáveis numéricas que serão utilizadas na análise com exceção da variável target.

```
✓ 0s [20] X = dados_num.drop('avg_price_brl',axis=1)
```

Aqui vai é definida a variável Y apenas com a variável target.

```
✓ 0s [21] Y = dados_num['avg_price_brl']
```

Aqui é feita a divisão do dados entre a parte que será usada para treinar o modelo e a parte que será usada para testar o modelo.

```
✓ 0s [22] X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.25, random_state=42)
```

c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost(biblioteca XGBRegressor)

- ✓ para predição dos preços dos carros. 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

```
✓ 25s ▶ # Preencher os valores ausentes em X_train e X_test
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test)

# Preencher os valores ausentes em Y_train
Y_train_imputed = imputer.fit_transform(Y_train.values.reshape(-1, 1)).ravel()

# Modelo RandomForest
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train_imputed, Y_train_imputed)

# Modelo XGBoost
xgb_model = XGBRegressor(random_state=42)
xgb_model.fit(X_train_imputed, Y_train_imputed)

# Modelo RandomForest com parâmetros ajustados
rf_model_tuned = RandomForestRegressor(n_estimators=100, max_depth=5, random_state=42)
rf_model_tuned.fit(X_train_imputed, Y_train_imputed)

# Modelo XGBoost com parâmetros ajustados
xgb_model_tuned = XGBRegressor(n_estimators=100, max_depth=3, learning_rate=0.1, random_state=42)
xgb_model_tuned.fit(X_train_imputed, Y_train_imputed)

# Avaliação dos modelos (opcional)
# (Você pode adicionar aqui a avaliação dos modelos usando X_test_imputed e Y_test)
```



XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=3, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=100, n_jobs=None,
              num_parallel_tree=None, random_state=42, ...)
```

✓ d. Grave os valores preditos em variáveis criadas

```
✓ [24] Y_pred_rf = rf_model.predict(X_test_imputed)
3s      Y_pred_xgb = xgb_model.predict(X_test_imputed)
      Y_pred_rf_tuned = rf_model_tuned.predict(X_test_imputed)
      Y_pred_xgb_tuned = xgb_model_tuned.predict(X_test_imputed)
```

e)

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

```
✓ [25] # Análise de importância das variáveis para o modelo RandomForest
0s      importances_rf = rf_model.feature_importances_
      feature_names = X.columns

      # Criar um DataFrame para visualização
      importances_df_rf = pd.DataFrame({'Feature': feature_names, 'Importance': importances_rf})
      importances_df_rf = importances_df_rf.sort_values(by='Importance', ascending=False)

      # Visualizar a importância das variáveis
      print("Importância das variáveis para o modelo RandomForest:")
      print(importances_df_rf)

      # Análise de importância das variáveis para o modelo XGBoost
      importances_xgb = xgb_model.feature_importances_

      # Criar um DataFrame para visualização
      importances_df_xgb = pd.DataFrame({'Feature': feature_names, 'Importance': importances_xgb})
      importances_df_xgb = importances_df_xgb.sort_values(by='Importance', ascending=False)

      # Visualizar a importância das variáveis
      print("Importância das variáveis para o modelo XGBoost:")
      print(importances_df_xgb)
```

Importância das variáveis para o modelo RandomForest:

	Feature	Importance
5	engine_size - numeric	0.481831
4	year_model	0.407889
3	gear	0.037368
2	fuel	0.033648
1	brand	0.025986
0	year_of_reference	0.013278

Importância das variáveis para o modelo XGBoost:

	Feature	Importance
5	engine_size - numeric	0.407052
4	year_model	0.275560
2	fuel	0.133859
3	gear	0.129480
1	brand	0.034835
0	year_of_reference	0.019213

f) Nos dois modelos analisados, notou-se uma maior importância para as variáveis `engine_size` e `year_model`. As demais variáveis, `fuel` e `gear` afetam de formas diferentes cada modelo. Já `brand` e `year_of_reference` são variáveis que têm uma menor influência nos dois modelos.

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

```

mse = mean_squared_error(Y_test, Y_pred_rf)
mae = mean_absolute_error(Y_test, Y_pred_rf)
r2 = r2_score(Y_test, Y_pred_rf)

print("RandomForest - MSE - ", mse)
print("RandomForest - MAE -", mae)
print("RandomForest - R^2 -", r2)

mse = mean_squared_error(Y_test, Y_pred_rf_tuned)
mae = mean_absolute_error(Y_test, Y_pred_rf_tuned)
r2 = r2_score(Y_test, Y_pred_rf_tuned)

print("\nRandomForest Tuned - MSE - ", mse)
print("RandomForest Tuned - MAE -", mae)
print("RandomForest Tuned - R^2 -", r2)

mse = mean_squared_error(Y_test, Y_pred_xgb)
mae = mean_absolute_error(Y_test, Y_pred_xgb)
r2 = r2_score(Y_test, Y_pred_xgb)

print("\nXGBoost - MSE - ", mse)
print("XGBoost - MAE -", mae)
print("XGBoost - R^2 -", r2)

mse = mean_squared_error(Y_test, Y_pred_xgb_tuned)
mae = mean_absolute_error(Y_test, Y_pred_xgb_tuned)
r2 = r2_score(Y_test, Y_pred_xgb_tuned)

print("\nXGBoost Tuned - MSE - ", mse)
print("XGBoost Tuned - MAE -", mae)
print("XGBoost Tuned - R^2 -", r2)

# O melhor modelo é o RandomForest
# RandomForest - MSE - 109260296.40292338
# RandomForest - MAE - 5577.5337231235135
# RandomForest - R^2 - 0.9588152066232576

```

```

➡ RandomForest - MSE - 109260296.40292338
RandomForest - MAE - 5577.5337231235135
RandomForest - R^2 - 0.9588152066232576

RandomForest Tuned - MSE - 424656702.10737133
RandomForest Tuned - MAE - 11730.560029104634
RandomForest Tuned - R^2 - 0.8399290583301677

XGBoost - MSE - 110781845.68670912
XGBoost - MAE - 5690.20555838328
XGBoost - R^2 - 0.9582416708108143

XGBoost Tuned - MSE - 205312790.36170554
XGBoost Tuned - MAE - 7709.001292769865
XGBoost Tuned - R^2 - 0.9226089885618961

```

h) O modelo que gerou o melhor resultado foi o RandomForest, utilizando os métodos de avaliação MSE (menor média de erro quadrático 109260296.40292338), MAE (menor média de erro absoluto 5577.5337231235135) e R^2 (maior precisão 0.9588152066232576).

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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)

Comandos

```
#Exercício 1
# 1 Carregue a base de dados Satellite
# install.packages("mlbench")
library(mlbench)
library(caret)
data(Satellite)

# 2 Crie partições contendo 80% para treino e 20% para teste
set.seed(7)
Satellite <- subset(Satellite, select = c(x.17,x.18,x.19,x.20,classes))
indices <- createDataPartition(Satellite$classes, p=0.80, list=FALSE)
treino <- Satellite[indices,]
teste <- Satellite[-indices,]

# Treine modelos RandomForest, SVM e RNA para predição destes dados.
rf <- train(classes~., data=treino, method="rf")
svm <- train(classes~., data=treino, method="svmRadial")
rna <- train(classes~., data=treino, method="nnet", trace=FALSE)

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

# Escolha o melhor modelo com base em suas matrizes de confusão.
confusionMatrix(predict.rf, teste$classes) #0.8419
confusionMatrix(predict.svm, teste$classes) #0.8707
confusionMatrix(predict.rna, teste$classes) #0.8084
#

# Indique qual modelo dá o melhor o resultado e a métrica utilizada
# O modelo que apresentou o melhor resultado foi o SVM com 87,07% de acurácia.
```

Outputs: Random Forest

```
> confusionMatrix(predict.rf, teste$classes)
Confusion Matrix and Statistics
```

Prediction	Reference				
	red soil	cotton crop	grey soil	damp grey soil	vegetation stubble
red soil	296	1	5	2	8
cotton crop	0	123	0	0	2
grey soil	3	0	238	29	1
damp grey soil	1	0	21	62	2
vegetation stubble	6	10	0	0	120
very damp grey soil	0	6	7	32	8

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

Overall Statistics

Accuracy : 0.8419
 95% CI : (0.8208, 0.8614)
 No Information Rate : 0.2383
 P-value [Acc > NIR] : < 2.2e-16

Kappa : 0.8047

McNemar's Test P-value : NA

Statistics by Class:

SVM

```
> confusionMatrix(predict.svm, teste$classes)
Confusion Matrix and Statistics
```

Prediction	Reference				
	red soil	cotton crop	grey soil	damp grey soil	vegetation stubble
red soil	298	1	4	2	7
cotton crop	1	120	0	0	4
grey soil	4	0	260	29	1
damp grey soil	0	1	7	69	2
vegetation stubble	3	14	0	2	117
very damp grey soil	0	4	0	23	10

Prediction	Reference				
	very damp grey soil				
red soil	0				
cotton crop	0				
grey soil	12				
damp grey soil	32				
vegetation stubble	3				
very damp grey soil	254				

Overall Statistics

Accuracy : 0.8707
 95% CI : (0.8511, 0.8886)
 No Information Rate : 0.2383
 P-value [Acc > NIR] : < 2.2e-16

Kappa : 0.8399

McNemar's Test P-value : NA

RNA

```
> confusionMatrix(predict.rna, teste$classes)
Confusion Matrix and Statistics
```

Prediction \ Reference	red soil	cotton crop	grey soil	damp grey soil	vegetation stubble
red soil	292	2	11	2	12
cotton crop	7	124	0	0	13
grey soil	3	0	259	67	1
damp grey soil	0	0	0	0	0
vegetation stubble	3	5	0	0	100
very damp grey soil	1	9	1	56	15

```

Prediction \ Reference
red soil      very damp grey soil
red soil      1
cotton crop    0
grey soil     25
damp grey soil 0
vegetation stubble 12
very damp grey soil 263

Overall Statistics

          Accuracy : 0.8084
          95% CI   : (0.7858, 0.8296)
    No Information Rate : 0.2383
    P-value [Acc > NIR] : < 2.2e-16

          Kappa   : 0.7595

McNemar's Test P-Value : NA
```

Avaliação do melhor modelo

O modelo que apresentou o melhor resultado foi o SVM com 87,07% de acurácia.

2 - Estimativa de Volumes de Árvores

Comandos

```
# Exercício 2
# 1 Carregar o arquivo Volumes.csv
(http://www.razer.net.br/datasets/Volumes.csv)
#install.packages("RSNNS")
library("ggplot2")
library("lattice")
library("caret")
library(RSNNS)
set.seed(7)

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

# 2 Eliminar a coluna NR, que só apresenta um número sequencial
df$NR <- NULL

# 3 Criar partição de dados: treinamento 80%, teste 20%
indices <- createDataPartition(df$VOL, p=0.80, list=FALSE)
treino <- df[indices,]
teste <- df[-indices,]
ene <- nrow(teste)
```

```

# 4 Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM
(svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de SPURR
rf <- caret::train(VOL~.,data=treino, method="rf")
svm <- caret::train(VOL~.,data=treino, method="svmRadial")
rna <- caret::train(VOL~.,data=treino, method="nnet", trace=FALSE, linout =
TRUE)
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, treino, start=list(b0=0.5,b1=0.5))

# 5 Efetue as previsões nos dados de teste
predicoes.rf <- predict(rf, teste)
predicoes.svm <- predict(svm, teste)
predicoes.rna <- predict(rna, teste)
predicoes.alom <- predict(alom, teste)

# 6 Crie suas próprias funções (UDF) e calcule as seguintes métricas entre a
predição e os dados observados

rquadrado <- function(y1,y2){
  ymean <- mean(y1)
  rr <- 1-(sum((y1-y2)^2)/sum((y1-ymean)^2))
  return(rr)
}

erroPadrao <- function(y1,y2,n){
  erro <- sqrt((sum((y1-y2)^2))/(n-2))
  return(erro)
}

erroPorcento <- function(y1,y2,n){
  erroP <- erroPadrao(y1,y2,n)
  ymean <- mean(y1)
  volta <- (erroP/ymean)*100
  return(volta)
}

print('-----Rando Forest-----')
cat("Valor R² calculado: ")
print(rquadrado(teste$VOL,predicoes.rf))
cat("Valor do Erro padrão da estimativa: ")
print(erroPadrao(teste$VOL,predicoes.rf,ene))
cat("Valor da porcentagem de erro: ")
print(erroPorcento(teste$VOL,predicoes.rf,ene))

print('-----SVM-----')

```

```

cat("Valor R² calculado: ")
print(rquadrado(teste$VOL,predicoes.svm))
cat("Valor do Erro padrão da estimativa: ")
print(erroPadrao(teste$VOL,predicoes.svm,ene))
cat("Valor da porcentagem de erro: ")
print(erroPorcento(teste$VOL,predicoes.svm,ene))

print('-----RNA-----')
cat("Valor R² calculado: ")
print(rquadrado(teste$VOL,predicoes.rna))
cat("Valor do Erro padrão da estimativa: ")
print(erroPadrao(teste$VOL,predicoes.rna,ene))
cat("Valor da porcentagem de erro: ")
print(erroPorcento(teste$VOL,predicoes.rna,ene))

print('-----ALOM-----')
cat("Valor R² calculado: ")
print(rquadrado(teste$VOL,predicoes.alom))
cat("Valor do Erro padrão da estimativa: ")
print(erroPadrao(teste$VOL,predicoes.alom,ene))
cat("Valor da porcentagem de erro: ")
print(erroPorcento(teste$VOL,predicoes.alom,ene))

```

Outputs

```

[1] "-----Rando Forest-----"
Valor R² calculado: [1] 0.8535647
Valor do Erro padrão da estimativa: [1] 0.1445527
Valor da porcentagem de erro: [1] 11.07658
[1] "-----SVM-----"
Valor R² calculado: [1] 0.8484652
Valor do Erro padrão da estimativa: [1] 0.1470481
Valor da porcentagem de erro: [1] 11.2678
[1] "-----RNA-----"
Valor R² calculado: [1] 0.8811983
Valor do Erro padrão da estimativa: [1] 0.1302011
Valor da porcentagem de erro: [1] 9.97687
[1] "-----ALOM-----"
Valor R² calculado: [1] 0.8263134
Valor do Erro padrão da estimativa: [1] 0.1574296
Valor da porcentagem de erro: [1] 12.0633

```

Escolha do melhor modelo

Avaliando todos os modelos a partir de seus resultados, podemos concluir que o modelo que obteve melhor resultado é o RNA, apresentando os seguintes números:

R2 = 0.8811983

Erro padrão = 0.1302011

Porcentagem de erro = 9.97687%

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

(15 pontos) Elaborar a tabela de 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

a)

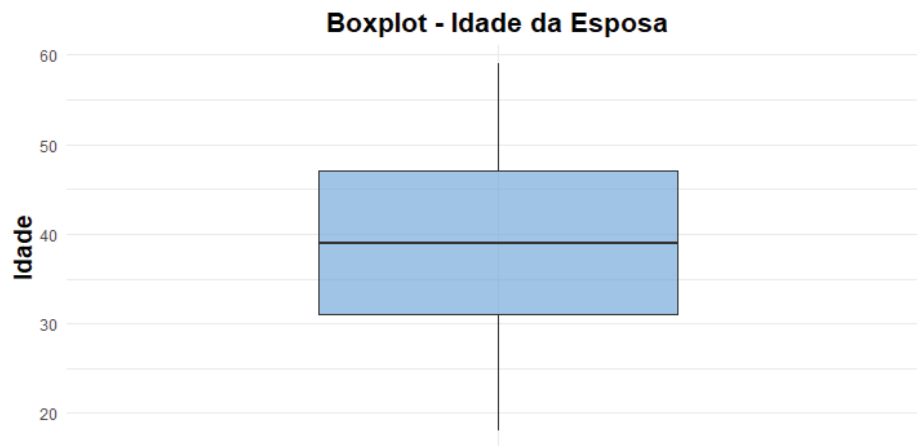


Figura 1 - Distribuição de idades das esposas usando box-plot

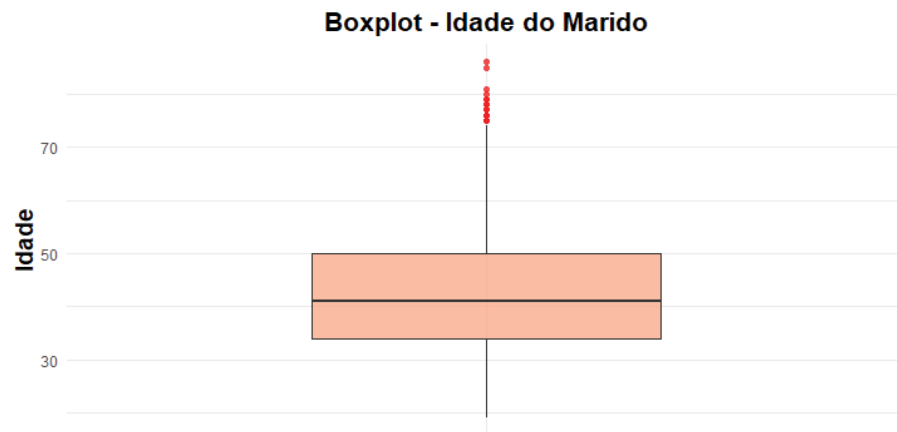


Figura 2 - Distribuição de idades dos maridos usando box-plot

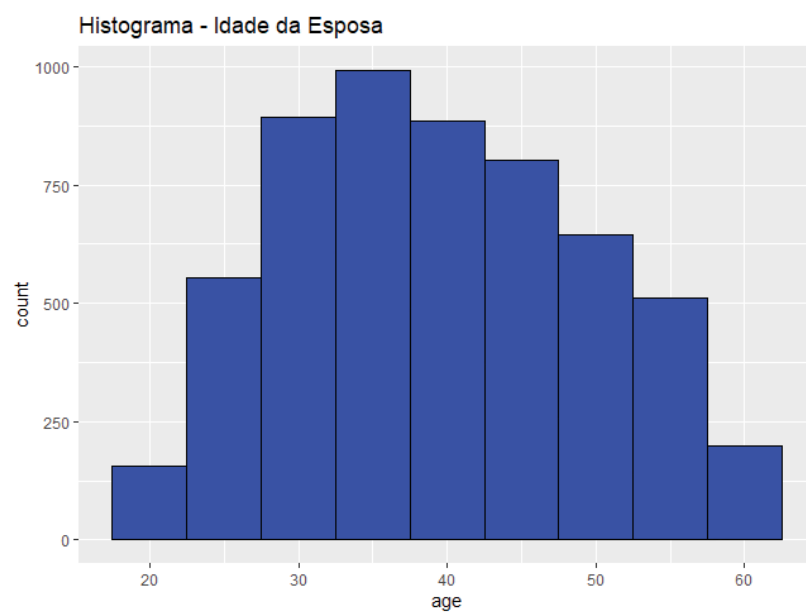


Figura 3 - Histograma idade das esposas

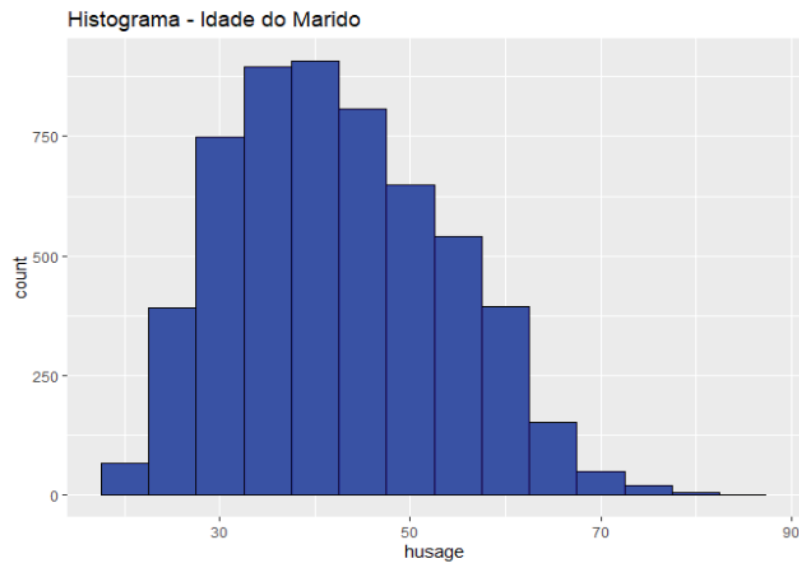


Figura 4 - Histograma idade dos maridos

Análise: Analisando os gráficos é possível notar que os maridos possuem uma idade maior em relação as esposas analisando pela mediana, o mesmo comportamento é observado no primeiro e terceiro quartil.

b)

Tabela de frequência idade marido

```
[1] "Tabela de Frequência da Idade do Marido:"
> print(df_husage)
```

	Idade	Frequência	Percentual
1	19	5	0.08874689
2	20	6	0.10649627
3	21	25	0.44373447
4	22	31	0.55023074
5	23	35	0.62122826
6	24	71	1.26020589
7	25	59	1.04721335
8	26	115	2.04117856
9	27	113	2.00567980
10	28	108	1.91693291
11	29	137	2.43166489
12	30	157	2.78665247
13	31	172	3.05289315
14	32	174	3.08839191
15	33	169	2.99964501
16	34	196	3.47887824
17	35	161	2.85764998
18	36	174	3.08839191
19	37	195	3.46112886
20	38	169	2.99964501
21	39	200	3.54987575
22	40	175	3.10614129
23	41	192	3.40788072
24	42	172	3.05289315
25	43	178	3.15938942
26	44	201	3.56762513
27	45	149	2.64465744
28	46	149	2.64465744
29	47	130	2.30741924
30	48	147	2.60915868
31	49	135	2.39616613
--	--	---	-----
32	50	127	2.25417110
33	51	112	1.98793042
34	52	128	2.27192048
35	53	102	1.81043663
36	54	104	1.84593539
37	55	126	2.23642173
38	56	103	1.82818601
39	57	106	1.88143415
40	58	86	1.52644657
41	59	91	1.61519347
42	60	81	1.43769968
43	61	74	1.31345403
44	62	62	1.10046148
45	63	45	0.79872204
46	64	45	0.79872204
47	65	30	0.53248136
48	66	18	0.31948882
49	67	14	0.24849130
50	68	14	0.24849130
51	69	13	0.23074192
52	70	8	0.14199503
53	71	7	0.12424565
54	72	9	0.15974441
55	73	6	0.10649627
56	74	2	0.03549876
57	75	7	0.12424565
58	76	4	0.07099752
59	77	2	0.03549876
60	78	2	0.03549876
61	79	2	0.03549876
62	80	1	0.01774938
63	81	1	0.01774938
64	85	1	0.01774938
65	86	1	0.01774938

Tabela de frequência idade esposas

	Idade	Frequência	Percentual				
1	18	12	0.2129925				
2	19	18	0.3194888				
3	20	31	0.5502307				
4	21	47	0.8342208				
5	22	47	0.8342208				
6	23	67	1.1892084				
7	24	84	1.4909478				
8	25	114	2.0234292				
9	26	114	2.0234292				
10	27	174	3.0883919				
11	28	161	2.8576500				
12	29	170	3.0173944				
13	30	184	3.2658857				
14	31	191	3.3901313				
15	32	187	3.3191338				
16	33	180	3.1948882				
17	34	204	3.6208733				
18	35	187	3.3191338				
19	36	205	3.6386226				
20	37	217	3.8516152				
21	38	202	3.5853745				
22	39	171	3.0351438				
23	40	181	3.2126376				
24	41	158	2.8044018				
25	42	172	3.0528931	34	51	117	2.0766773
26	43	185	3.2836351	35	52	138	2.4494143
27	44	185	3.2836351	36	53	103	1.8281860
28	45	140	2.4849130	37	54	109	1.9346823
29	46	164	2.9108981	38	55	104	1.8459354
30	47	128	2.2719205	39	56	91	1.6151935
31	48	131	2.3251686	40	57	104	1.8459354
32	49	125	2.2186723	41	58	104	1.8459354
33	50	133	2.3606674	42	59	95	1.6861910

Observações Principais:

1. Idade: A distribuição etária segue uma curva em forma de sino, com o pico ocorrendo entre os 20 e 30 anos e diminuindo à medida que a idade avança. Isso é típico para uma população trabalhadora.

2. Salário Familiar: A distribuição dos salários familiares é assimétrica à direita,

indicando que há mais famílias com salários baixos e menos com salários muito altos.

3. Tendências Centrais:

- **Idade:** A faixa etária mais comum (moda) está entre 20 e 30 anos.
- **Salário Familiar:** A moda (categoria de salário mais frequente)

provavelmente está na parte inferior da faixa.

4. Variabilidade:

- **Idade:** Existe uma boa variedade de idades, sugerindo que o conjunto de dados inclui pessoas em diferentes estágios de carreira.
- **Salário Familiar:** A variabilidade nos salários é provavelmente maior do que a da idade, como é comum em distribuições de renda desiguais.

2) Medidas de posição e dispersão

a) Medidas “age” – esposa

Média: 39.42758

Mediana: 39

Moda: 37

Medidas “husage” – Marido

Média: 42.45296

Mediana: 41

Moda: 44

Observações Gerais:

Distribuição de Idade: A média e a mediana para "idade" são relativamente próximas, sugerindo uma distribuição bastante simétrica.

Idade do Chefe de Família: A média para "husage" é ligeiramente superior à mediana, indicando que a distribuição pode estar levemente inclinada para a direita (algumas famílias com chefes de família mais velhos estão elevando a média).

b) Medidas “age” - esposa

Variância: 99.75234

Desvio padrão: 9.98761

Coeficiente de variação: 25.33153

Medidas “husage” - marido

Variância: 126.0717

Desvio padrão: 11.22817

Coeficiente de variação: 26.44849

Variância:

- **Idade:** A variância da idade é 99,75. Isso significa que há um certo grau de dispersão nas idades dentro do seu conjunto de dados.

- **Idade Chefe Família:** A variância da idade do chefe da família é 126,07. Isso indica uma dispersão ligeiramente maior das idades dos chefes de família em comparação com a distribuição geral de idade.

Desvio Padrão:

- **Idade:** O desvio padrão da idade é 9,99. Em média, as idades individuais em seu conjunto de dados desviam-se da idade média em cerca de 9,99 anos.
- **Idade Chefe Família:** O desvio padrão da idade do chefe da família é 11,23. Em média, as idades dos chefes de família desviam-se da média da idade do chefe da família em cerca de 11,23 anos.

Coeficiente de Variação (CV):

- **Idade:** O CV para idade é de 25,33%, indicando um nível moderado de variabilidade relativa nos dados de idade.
- **Idade Chefe Família:** O CV para a idade do chefe da família é de 26,45%, indicando também um nível moderado de variabilidade relativa. A variabilidade da idade dos chefes de família é ligeiramente superior à variabilidade geral da idade.

3 - Testes paramétricos e não paramétricos

- a) 1) Testada normalidade com Shapiro-Wilk, utilizando uma subamostra e também com Kolmogorov-Smirnov utilizando a amostra completa. Ambos obtendo o resultado de ***p-value < 2.2e-16***.

2) Utilizado o teste Levene com centro na mediana, foi obtido o seguinte resultado:

```
> leveneTest(salarios$age, salarios$husage, center = "median")
Levene's Test for Homogeneity of Variance (center = "median")
      Df F value    Pr(>F)
group  64  4.0244 < 2.2e-16 ***
      5569
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3) Devido a este resultado de normalidade, e o resultado do teste de Levine onde apontou variâncias desiguais, ou seja, o valor p extremamente pequeno (<2,2e-16) no teste de Levene é uma forte indicação de que as variâncias dos grupos que em comparação não são iguais. Significando assim que há uma diferença significativa na distribuição dos dados entre os grupos. Por isso foi escolhido o *Welch Two Sample t-test*.

```
> welch_test_result <- t.test(salarios$age, salarios$husage, var.equal = FALSE)
> print(welch_test_result)

Welch Two Sample t-test

data:  salarios$age and salarios$husage
t = -15.111, df = 11115, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.417821 -2.632943
sample estimates:
mean of x mean of y
 39.42758  42.45296
```

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

husage = 40	(anos – idade do marido)
husunion = 0	(marido não possui união estável)
husearns = 600	(US\$ renda do marido por semana)
huseduc = 13	(anos de estudo do marido)
husblk = 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

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

Tabela de resultados

Modelo	Treino		Teste	
	RMSE	R^2	RMSE	R^2
Lasso	0,8388284	0,2960251	0,8610461	0,2324235
Ridge	0,8384184	0,296713	0,861604	0,2314283
ElasticNet	0,8381841	0,297106	0,8627815	0,2293262

Para podermos determinar o melhor modelo com base nos resultados, é possível analisar as métricas de desempenho RMSE e R^2 tanto para os conjuntos de treino quanto de teste. De certa forma o RMSE quanto menor, melhor, indicando o desvio padrão dos erros de predição. Já o R^2 quanto maior, melhor ao qual mede a proporção da variabilidade nos dados que vêm ao modelo.

Quanto os resultados:

- **RMSE (Treino)** - ElasticNet tem o menor RMSE (0.8381841), seguido por Ridge (0.8384184) e Lasso (0.8388284).
- **R^2 (Treino)** - ElasticNet tem o maior R^2 (0.297106), seguido por Ridge (0.296713) e Lasso (0.2960251).
- **RMSE (Teste)** - Lasso tem o menor RMSE (0.8610461), seguido por Ridge (0.861604) e ElasticNet (0.8627815).
- **R^2 (Teste)** - Lasso tem o maior R^2 (0.2324235), seguido por Ridge (0.2314283) e ElasticNet (0.2293262).

APÊNDICE 6 – ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

Parta do exemplo (notebook produzido no Colab) que foi 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

Entrada: "texto qualquer"

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

O processo de Reconhecimento de Padrões

O objetivo deste trabalho é demonstrar o processo de "construção de atributos" e como ele é fundamental para o Reconhecimento de Padrões (RP).

Primeiro um conjunto de "amostras" previamente conhecido (classificado)

```
#  
# amostras de texto em diferentes línguas  
#  
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.",
]

```

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

Um padrão é um conjunto de características, geralmente representado por um vetor e um conjunto de padrões no formato de tabela. Onde cada linha é um padrão e as colunas as características e, geralmente, na última coluna a classe.

```
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)
```

Exemplo de saída: [['Estou ansioso para as férias de verão.', 'português'], ['Estoy emocionado por el concierto.', 'espanhol'], ['El perro está jugando.', 'espanhol'], ['Can you help me with this?']

O DataFrame do pandas facilita a visualização.

```
import pandas as pd
dados = pd.DataFrame(pre_padroes)
dados
```

	0	1
0	Estou ansioso para as férias de verão.	português
1	Estoy emocionado por el concierto.	espanhol
2	El perro está jugando.	espanhol
3	Can you help me with this?	inglês
4	A comida estava deliciosa!	português
...	...	...
87	Me encanta leer libros.	espanhol
88	She sings beautifully.	inglês
89	Do you speak English?	inglês
90	Adoro explorar novos lugares.	português
91	Let's go for a walk.	inglês
92 rows x 2 columns		

Construção dos atributos

Esse é o coração desse trabalho e que deverá ser desenvolvido por vocês. Pensem em como podemos "medir" cada frase/sentença e extrair características que melhorem o resultado do processo de identificação.

Após a criação de cada novo atributo, execute as etapas seguintes e registre as métricas da matriz de confusão. Principalmente acurácia e a precisão.

```
# A ENTRADA É O VETOR PRE_PADROES E A SAÍDA DESSE PASSO DEVERÁ SER "PADRÕES"
IMPORT RE

DEF TAMANHOMEDIOFRASES (TEXTO) :
    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 TEMIm (FRASE) :
    IF FRASE.FIND("I'M") != -1 OR FRASE.FIND("I'M") != -1 OR FRASE.FIND("I ") != -1:
        RETURN 1
    RETURN 0

DEF TEMTos (FRASE) :
    IF FRASE.FIND(" TO ") != -1 OR FRASE.FIND(" IS ") != -1 OR FRASE.FIND(" ARE ") != -1:
        RETURN 1
    RETURN 0

DEF TEMIng (FRASE) :
    IF FRASE.FIND("ING ") != -1:
        RETURN 1
    RETURN 0

DEF TEMAtil (FRASE) :
    IF FRASE.FIND("Â") != -1:
        RETURN 1
```

```

    RETURN 0

DEF TEMCEDILHA (FRASE) :
    IF FRASE.FIND ("Ç") != -1 :
        RETURN 1
    RETURN 0

DEF TEMEU (FRASE) :
    IF FRASE.FIND ("EU") != -1 OR FRASE.FIND (" DE ") != -1 OR FRASE.FIND (" UMA ") != -1 OR
FRASE.FIND (" AO ") != -1 :
        RETURN 1
    RETURN 0

DEF TEMOMAIUSCULO (FRASE) :
    IF FRASE.FIND ("O ") != -1 OR FRASE.FIND (" o ") != -1 :
        RETURN 1
    RETURN 0

DEF TEMSS (FRASE) :
    IF FRASE.FIND ("ss") != -1 :
        RETURN 1
    RETURN 0

DEF TEMINTERROGA (FRASE) :
    IF FRASE.FIND ("¿") != -1 OR FRASE.FIND ("¡") != -1 :
        RETURN 1
    RETURN 0

DEF TEMELAL (FRASE) :
    IF FRASE.FIND (" EL ") != -1 OR FRASE.FIND (" AL ") != -1 OR FRASE.FIND (" LA ") != -1
OR FRASE.FIND (" LAS ") != -1 OR FRASE.FIND (" ES ") != -1 :
        RETURN 1
    RETURN 0

DEF TEMPARA (FRASE) :
    IF FRASE.FIND ("PARA") != -1 :
        RETURN 1
    RETURN 0

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=TEMIM (TEXTO)

    CARACTERISTICA3=TEMING (TEXTO)

    CARACTERISTICA4=TEMATIL (TEXTO)

    CARACTERISTICA5=TEMCEDILHA (TEXTO)

    CARACTERISTICA6=TEMEU (TEXTO)

    CARACTERISTICA7=TEMOIAUSCULO (TEXTO)

    CARACTERISTICA8=TEMSS (TEXTO)

    CARACTERISTICA9=TEMINTERROGA (TEXTO)

    CARACTERISTICA10=TEMEIAL (TEXTO)

    CARACTERISTICA11=TEMPARA (TEXTO)

    CARACTERISTICA12=TEMTOS (TEXTO)

    # ACRESCENTE AS SUAS FUNCOES NO VETOR PADRAO

    PADRAO = [CARACTERISTICA1, CARACTERISTICA2, CARACTERISTICA3,
CARACTERISTICA4, CARACTERISTICA5, CARACTERISTICA6, CARACTERISTICA7, CARACTERISTICA8, CARACTERISTIC
A9, CARACTERISTICA10, CARACTERISTICA11, CARACTERISTICA12, 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

```

```
[[4.428571428571429, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 'português'], [5.8, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 'espanhol']]
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	4.428571	0	0	0	0	1	0	0	0	0	1	0	português
1	5.800000	0	0	0	0	0	0	0	0	1	0	0	espanhol
2	4.500000	0	0	0	0	0	0	0	0	0	0	0	espanhol
3	3.333333	0	0	0	0	0	0	0	0	0	0	0	inglês
4	5.500000	0	0	0	0	0	0	0	0	0	0	0	português
...	...	...	...	...	...	...	...	...	...	...	...	...	...
87	4.750000	0	0	0	0	0	0	0	0	0	0	0	espanhol
88	6.333333	0	0	0	0	0	0	0	0	0	0	0	inglês
89	4.250000	0	0	0	0	0	0	0	0	0	0	0	inglês
90	6.250000	0	0	0	0	0	0	0	0	0	0	0	português
91	2.333333	0	0	0	0	0	0	0	0	0	0	0	inglês

92 rows x 13 columns

Treinando o modelo com SVM

Separando o conjunto de treinamento do conjunto de testes

```
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)
```

Com os conjuntos separados, podemos "treinar" o modelo usando a SVM.

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

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

#
# score com os dados de treinamento
```

```

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

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

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

```

Acurácia nos dados de treinamento: 82.61%

```

[[12  2  8]
 [ 1 21  1]
 [ 0  0 24]]

```

	precision	recall	f1-score	support
espanhol	0.92	0.55	0.69	22
inglês	0.91	0.91	0.91	23
português	0.73	1.00	0.84	24
accuracy			0.83	69
macro avg	0.85	0.82	0.81	69
weighted avg	0.85	0.83	0.82	69

métricas mais confiáveis

```

[[7 0 1]
 [2 4 1]
 [1 1 6]]

```

	precision	recall	f1-score	support
espanhol	0.70	0.88	0.78	8
inglês	0.80	0.57	0.67	7
português	0.75	0.75	0.75	8
accuracy			0.74	23
macro avg	0.75	0.73	0.73	23
weighted avg	0.75	0.74	0.73	23

2 - Melhore uma base de dados ruim

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

# DADOS
url = "https://archive.ics.uci.edu/dataset/544/estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition"
obesidade = pd.read_csv('/content/sample_data/ObesityDataSet_raw_and_data_synthetic.csv', header=0)
```

```
# Codificação
obesidade["Gender"].replace({'Female':0, 'Male':1}, inplace=True)
obesidade["family_history_with_overweight"].replace({'no':0, 'yes':1}, inplace=True)
obesidade["FAVC"].replace({'no':0, 'yes':1}, inplace=True)
obesidade["CAEC"].replace({'Sometimes':0, 'Frequently':1, 'Always':2, 'no':3}, inplace=True)
obesidade["SMOKE"].replace({'no':0, 'yes':1}, inplace=True)
obesidade["SCC"].replace({'no':0, 'yes':1}, inplace=True)
obesidade["CALC"].replace({'Sometimes':0, 'Frequently':1, 'Always':2, 'no':3}, inplace=True)
obesidade["MTRANS"].replace({'Public_Transportation':0, 'Walking':1, 'Automobile':2, 'Motorbike':3, 'Bike':4}, inplace=True)
obesidade["NObeyesdad"].replace({'Normal_Weight':0, 'Overweight_Level_I':1, 'Overweight_Level_II':2, 'Obesity_Type_I':3, 'Insufficient_Weight':4, 'Obesity_Type_II':5, 'Obesity_Type_III':6}, inplace=True)

#obesidade.head()
```

```
#
X = obesidade.iloc[:, :16]
X_orig = obesidade.iloc[:, :16]
Y = obesidade["NObeyesdad"]
```

```
Y_orig = obesidade["NObesidad"]
```

```
# Normalização
```

```
X = pd.DataFrame( minmax_scale(X) )
```

```
# Separação dos dados entre a base de treino e teste originais
```

```
X_oring_train, X_orig_test, y_orig_train, y_orig_test =  
train_test_split(X_orig,Y_orig, test_size=0.25,  
stratify=Y_orig,random_state=10)
```

```
# Separação dos dados entre a base de treino e teste alterado
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y,  
test_size=0.25,stratify=Y,random_state=10)
```

```
treinador = svm.SVC() #algoritmo escolhido
```

```
modelo_orig = treinador.fit(X_oring_train, y_orig_train)
```

```
# predição com os mesmos dados usados para treinar
```

```
y_orig_pred = modelo_orig.predict(X_oring_train)
```

```
cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
```

```
print('Matriz de confusão - com os dados ORIGINAIS usados no  
TREINAMENTO')
```

```
print(cm_orig_train)
```

```
print(classification_report(y_orig_train, y_orig_pred))
```

```
# 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_orig_test)
```

```
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
```

```
print(cm_orig_test)
```

```
print(classification_report(y_orig_test, y2_orig_pred))
```

Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO

```
[[ 91 48 13  1 62  0  0]
 [ 41 111 53  9  4  0  0]
 [ 18 21 129 49  0  0  0]
 [  0  3 100 85  0 18 57]
 [ 30  0  0  0 174  0  0]
 [  0  0  0 31  0 112 80]
 [  0  0  0  0  0  0 243]]
```

	precision	recall	f1-score	support
0	0.51	0.42	0.46	215
1	0.61	0.51	0.55	218
2	0.44	0.59	0.50	217
3	0.49	0.32	0.39	263
4	0.72	0.85	0.78	204
5	0.86	0.50	0.63	223
6	0.64	1.00	0.78	243
accuracy			0.60	1583
macro avg	0.61	0.60	0.59	1583
weighted avg	0.61	0.60	0.58	1583

Matriz de confusão - com os dados ORIGINAIS usados para TESTES

```
[[25 14  1  0 32  0  0]
 [12 38 13  6  3  0  0]
 [ 7 11 41 13  0  0  1]
 [ 0  1 36 33  0  5 13]
 [ 6  0  0  0 62  0  0]
 [ 0  0  0  6  0 29 39]
 [ 0  0  0  0  0  0 81]]
```

	precision	recall	f1-score	support
0	0.50	0.35	0.41	72
1	0.59	0.53	0.56	72
2	0.45	0.56	0.50	73
3	0.57	0.38	0.45	88
4	0.64	0.91	0.75	68
5	0.85	0.39	0.54	74
6	0.60	1.00	0.75	81
accuracy			0.59	528
macro avg	0.60	0.59	0.57	528
weighted avg	0.60	0.59	0.56	528

```
treinador = svm.SVC()

modelo = treinador.fit(X_train, y_train)

y_pred = modelo.predict(X_train)
cm_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no
TREINAMENTO')
```

```

print(cm_train)
print(classification_report(y_train, y_pred))

print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_pred = modelo.predict(X_test)
cm_test = confusion_matrix(y_test, y2_pred)
print(cm_test)
print(classification_report(y_test, y2_pred))

```

➡ Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO

[	155	15	14	4	27	0	0]
[	20	159	22	13	4	0	0]
[	10	18	167	22	0	0	0]
[	4	4	15	211	0	26	3]
[	12	1	0	0	191	0	0]
[	1	0	0	3	0	219	0]
[	0	0	0	0	0	1	242]]
				precision	recall	f1-score	support
	0			0.77	0.72	0.74	215
	1			0.81	0.73	0.77	218
	2			0.77	0.77	0.77	217
	3			0.83	0.80	0.82	263
	4			0.86	0.94	0.90	204
	5			0.89	0.98	0.93	223
	6			0.99	1.00	0.99	243
	accuracy					0.85	1583
	macro avg			0.84	0.85	0.85	1583
	weighted avg			0.85	0.85	0.85	1583

Matriz de confusão - com os dados ORIGINAIS usados para TESTES

[	37	16	7	1	11	0	0]
[	6	50	9	5	2	0	0]
[	3	9	53	8	0	0	0]
[	1	0	7	68	0	11	1]
[	4	0	0	0	64	0	0]
[	0	0	2	2	0	70	0]
[	0	0	1	0	0	0	80]]
				precision	recall	f1-score	support
	0			0.73	0.51	0.60	72
	1			0.67	0.69	0.68	72
	2			0.67	0.73	0.70	73
	3			0.81	0.77	0.79	88
	4			0.83	0.94	0.88	68
	5			0.86	0.95	0.90	74
	6			0.99	0.99	0.99	81
	accuracy					0.80	528
	macro avg			0.79	0.80	0.79	528
	weighted avg			0.80	0.80	0.79	528

APÊNDICE 7 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

Seed utilizado: 202483

(Ano atual com 4 dígitos + 2 algarismos do dígito verificador do CPF de um dos integrantes)

CLASSIFICAÇÃO

Para o experimento de Classificação:

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

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	C=1 Sigma=0.078	0.766	<pre> Reference Prediction bus opel saab van bus 43 0 1 1 opel 0 20 13 0 saab 0 21 27 0 van 0 1 2 38 </pre>
SVM – Hold-out	C=1 Sigma=0.065	0.743	<pre> Reference Prediction bus opel saab van bus 43 1 1 1 opel 0 16 13 0 saab 0 24 27 0 van 0 1 2 38 </pre>

RF – Hold-out	mtry=2	0.713	<table><tr><th colspan="5">Reference</th></tr><tr><th>Prediction</th><th>bus</th><th>opel</th><th>saab</th><th>van</th></tr><tr><td>bus</td><td>43</td><td>3</td><td>2</td><td>2</td></tr><tr><td>opel</td><td>0</td><td>19</td><td>18</td><td>0</td></tr><tr><td>saab</td><td>0</td><td>19</td><td>20</td><td>0</td></tr><tr><td>van</td><td>0</td><td>1</td><td>3</td><td>37</td></tr></table>	Reference					Prediction	bus	opel	saab	van	bus	43	3	2	2	opel	0	19	18	0	saab	0	19	20	0	van	0	1	3	37
Reference																																	
Prediction	bus	opel	saab	van																													
bus	43	3	2	2																													
opel	0	19	18	0																													
saab	0	19	20	0																													
van	0	1	3	37																													
RF – CV	mtry=10	0.713	<table><tr><th colspan="5">Reference</th></tr><tr><th>Prediction</th><th>bus</th><th>opel</th><th>saab</th><th>van</th></tr><tr><td>bus</td><td>43</td><td>3</td><td>0</td><td>1</td></tr><tr><td>opel</td><td>0</td><td>21</td><td>22</td><td>0</td></tr><tr><td>saab</td><td>0</td><td>17</td><td>17</td><td>0</td></tr><tr><td>van</td><td>0</td><td>1</td><td>4</td><td>38</td></tr></table>	Reference					Prediction	bus	opel	saab	van	bus	43	3	0	1	opel	0	21	22	0	saab	0	17	17	0	van	0	1	4	38
Reference																																	
Prediction	bus	opel	saab	van																													
bus	43	3	0	1																													
opel	0	21	22	0																													
saab	0	17	17	0																													
van	0	1	4	38																													
KNN	k=1	0.635	<table><tr><th colspan="5">Reference</th></tr><tr><th>Prediction</th><th>bus</th><th>opel</th><th>saab</th><th>van</th></tr><tr><td>bus</td><td>38</td><td>2</td><td>2</td><td>3</td></tr><tr><td>opel</td><td>3</td><td>19</td><td>23</td><td>0</td></tr><tr><td>saab</td><td>2</td><td>20</td><td>15</td><td>2</td></tr><tr><td>van</td><td>0</td><td>1</td><td>3</td><td>34</td></tr></table>	Reference					Prediction	bus	opel	saab	van	bus	38	2	2	3	opel	3	19	23	0	saab	2	20	15	2	van	0	1	3	34
Reference																																	
Prediction	bus	opel	saab	van																													
bus	38	2	2	3																													
opel	3	19	23	0																													
saab	2	20	15	2																													
van	0	1	3	34																													
RNA – Hold-out	size=3 decay=0.1	0.461	<table><tr><th colspan="5">Reference</th></tr><tr><th>Prediction</th><th>bus</th><th>opel</th><th>saab</th><th>van</th></tr><tr><td>bus</td><td>40</td><td>3</td><td>2</td><td>1</td></tr><tr><td>opel</td><td>1</td><td>34</td><td>38</td><td>37</td></tr><tr><td>saab</td><td>2</td><td>5</td><td>3</td><td>1</td></tr><tr><td>van</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	Reference					Prediction	bus	opel	saab	van	bus	40	3	2	1	opel	1	34	38	37	saab	2	5	3	1	van	0	0	0	0
Reference																																	
Prediction	bus	opel	saab	van																													
bus	40	3	2	1																													
opel	1	34	38	37																													
saab	2	5	3	1																													
van	0	0	0	0																													
RNA – CV	size=5 decay=0.1	0.515	<table><tr><th colspan="5">Reference</th></tr><tr><th>Prediction</th><th>bus</th><th>opel</th><th>saab</th><th>van</th></tr><tr><td>bus</td><td>42</td><td>1</td><td>2</td><td>25</td></tr><tr><td>opel</td><td>0</td><td>17</td><td>12</td><td>2</td></tr><tr><td>saab</td><td>1</td><td>23</td><td>27</td><td>12</td></tr><tr><td>van</td><td>0</td><td>1</td><td>2</td><td>0</td></tr></table>	Reference					Prediction	bus	opel	saab	van	bus	42	1	2	25	opel	0	17	12	2	saab	1	23	27	12	van	0	1	2	0
Reference																																	
Prediction	bus	opel	saab	van																													
bus	42	1	2	25																													
opel	0	17	12	2																													
saab	1	23	27	12																													
van	0	1	2	0																													

Resultado com 3 Linhas:

```

"", "Comp", "Circ", "DCirc", "RadRa", "PrAxisRa", "MaxLRa", "ScatRa", "Elong", "PrAxisRec
t", "MaxLRect", "ScVarMaxis", "ScVarmaxis", "RaGyr", "SkewMaxis", "Skewmaxis", "Kurt
maxis", "KurtMaxis", "HollRa", "predict.svm_novos"
"1", 90, 44, 80, 150, 60, 8, 175, 50, 17, 144, 180, 355, 175, 66, 7, 15, 171, 188, "opel"
"2", 80, 49, 76, 148, 66, 9, 173, 53, 13, 134, 179, 343, 176, 54, 5, 12, 189, 176, "opel"
"3", 86, 54, 64, 161, 53, 6, 136, 43, 18, 132, 168, 321, 163, 45, 8, 14, 174, 181, "opel"

```

Lista de comandos:

```

#### Treinamento de SVM com Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
svm_cv <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl)
svm_cv
predict.svm_cv <- predict(svm_cv, teste)

```

```
confusionMatrix(predict.svm_cv, as.factor(teste$tipo))
```

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

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	C=0.5 Sigma=0.13	0.765	Reference Prediction neg pos neg 88 24 pos 12 29
RNA – Hold-out	size=5 decay=0.1	0.745	Reference Prediction neg pos neg 80 19 pos 20 34
SVM – Hold-out	C=0.25 Sigma=0.13	0.745	Reference Prediction neg pos neg 88 27 pos 12 26
RF – Hold-out	mtry=5	0.739	Reference Prediction neg pos neg 82 22 pos 18 31
KNN	k=9	0.725	Reference Prediction neg pos neg 78 20 pos 22 33
RF – CV	mtry=2	0.725	Reference Prediction neg pos neg 83 25 pos 17 28
RNA – CV	size=3 decay=0.1	0.719	Reference Prediction neg pos neg 80 23 pos 20 30

Resultado com 3 Linhas:

```
"", "num", "preg0nt", "glucose", "pressure", "triceps", "insulin", "mass", "pedigree", "age", "predict.svm"  
"1", 1, 7, 150, 80, 40, 0, 25.4, 0.781, 43, "pos"  
"2", 2, 4, 147, 77, 32, 100, 13, 0.931, 50, "neg"  
"3", 3, 9, 169, 51, 45, 40, 30.6, 0.554, 61, "pos"
```

Lista de comandos:

```
#### Treinamento SVM com Cross-validation  
ctrl <- trainControl(method = "cv", number = 10)
```

```
svm_cv <- train(diabetes~., data=treino, method="svmRadial", trControl=ctrl)
svm_cv
predict.svm_cv <- predict(svm_cv, teste)
confusionMatrix(predict.svm_cv, as.factor(teste$diabetes))
#####
#####
```

Regressão

Para o experimento de Regressão:

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

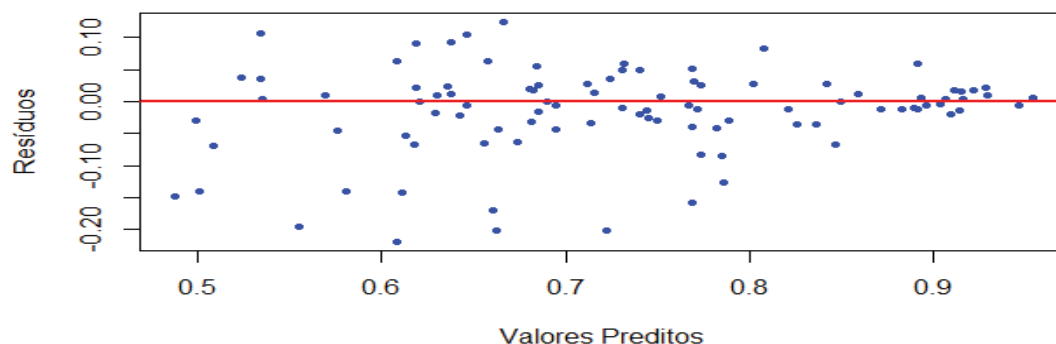
Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – CV	mtry=2	0.78	0.0014	0.89	0.07	0.048
RF – Hold-out	mtry=2	0.78	0.0014	0.89	0.069	0.048
SVM – CV	C=1 Sigma=0.16	0.76	0.0015	0.89	0.073	0.049
RNA – CV	size=10 decay=0.1	0.77	0.0014	0.89	0.071	0.05
SVM – Hold-out	C=0.5 Sigma=0.19	0.75	0.0015	0.89	0.075	0.05
RNA – Hold-out	size=5 decay=0.1	0.75	0.0015	0.88	0.075	0.053
KNN	K=9	0.68	0.0017	0.83	0.084	0.06

Resultado com 3 Linhas:

```
,"GRE.Score","TOEFL.Score","University.Rating","SOP","LOR","CGPA","Research","predicoes.rf_cv_novos"
1,300,100,3,2,4,5.3,0,0.52
2,250,120,4,3,6.8,7.2,1,0.64
3,350,110,4.5,5.8,6.1,1,1,0.76
```

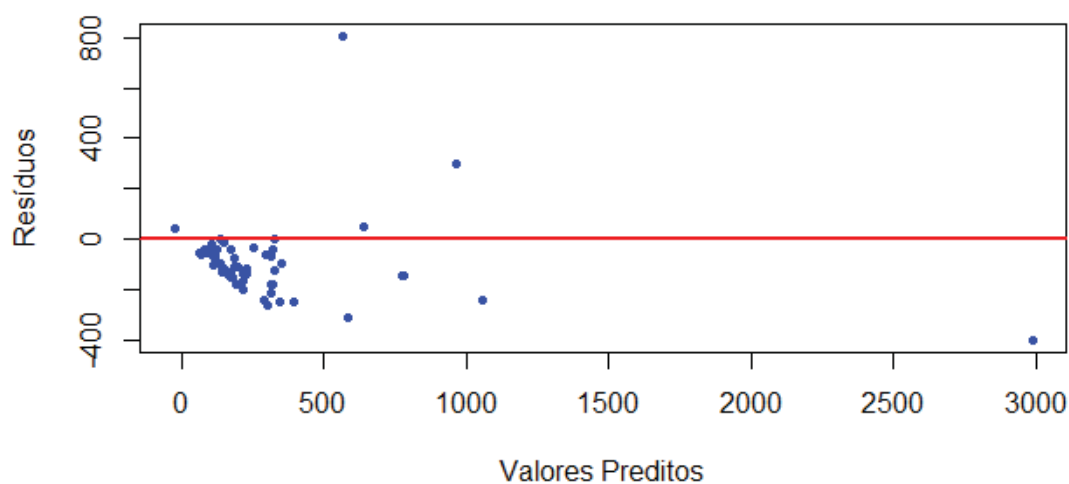
Gráfico de Resíduos - RF com $R^2 = 0.78$



Lista de comandos:

```
##### Treinamento de Random Forest com CrossValidation
ctrl <- trainControl(method = "cv", number = 10)
rf_cv <- train(ChanceOfAdmit~., data=treino, method="rf", trControl=ctrl)
rf_cv
predicoes.rf_cv <- predict(rf_cv, teste)
```

Gráfico de Resíduos - SVM com $R^2 = 0.81$



Lista de comandos:

```
#### Treinamento de SVM com HoldOut
svm <- train(biomassa~., data=treino, method="svmRadial")
svm
predicoes.svm <- predict(svm, teste)
# Calcular metricas
r2(predicoes.svm, teste$biomassa)
rmse(teste$biomassa, predicoes.svm)
MAE <- mean(abs(teste$biomassa - predicoes.svm))
MAE
Pearson <- cor(teste$biomassa, predicoes.svm)
```

Pearson
`syx(teste$biomassa, predicoes.svm,dados,3)`

AGRUPAMENTO

Veículo

Lista de Clusters gerados:

10 primeiras linhas do arquivo com o cluster correspondente.

Usado 10 clusters no experimento.

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

```
Cluster modes:
  Comp  Circ  DCirc  RadRa  PrAxisRa  MaxLRa  ScatRa  Elong  PrAxisRect  MaxLRect  ScvarMaxis  Scvarmaxis  RaGyr
1    89    36    66   127      56      7   135    50          18      125      155      266   139
2    82    44    73   141      64      7   151    44          19      143      175      312   177
3    89    47    85   162      64     11   157    43          20      155      173      354   186
4    85    45    70   120      56      7   149    45          19      145      170      322   171
5   104    53   100   197      60     11   213    31          24      162      226      669   214
6   100    55   101   206      62     10   222    30          25      156      214      706   218
7    93    37    66   125      59      8   130    51          18      142      159      259   119
8    85    40    66   110      55      6   122    57          17      128      137      203   127
9    86    42    75   169      68      7   150    44          19      138      169      324   173
10   93    43    85   166      66      9   144    44          19      147      168      309   143

  skewMaxis  skewmaxis  kurtmaxis  KurtMaxis  HollRa  tipo
1         66          1         2        180    184  saab
2         75          7        14        183    187  bus
3         75          1         9        183    195  van
4         85          4         4        179    183  bus
5         68          0        11        188    199  saab
6         72          0         5        187    198  ope1
7         65          1        12        201    201  van
8         64          7         7        180    183  van
9         71          2        11        192    197  bus
10        67          3         5        186    197  van
```

Lista de comandos:

```
cluster.results <- kmodes(dados, 10, iter.max = 10, weighted = FALSE )
cluster.results
```

```
resultado <- cbind(dados, cluster.results$cluster)
resultado
```

```
write.csv(resultado,"saida_Cluster_Agrupamento.csv")
```

REGRAS DE ASSOCIAÇÃO

Musculação

Regras geradas com uma configuração de Suporte e Confiança.

Regra: mínimo de 0.001, confiança mínima de 0.7, e comprimento mínimo das regras de 2.

rules	support	confidence	coverage	lift	count
1 {Gemeos=Gemeos}=>{Crucifixo=Esteira}	0.0263157		1 0.0263157894736842	316.666.666.666.667	1
2 {Gemeos=Gemeos}=>{Afundo=Bicicleta}	0.0263157		1 0.0263157894736842	316.666.666.666.667	1
3 {Gemeos=Gemeos}=>{LegPress=Extensor}	0.0263157		1 0.0263157894736842	292.307.692.307.692	1
4 {Crucifixo=Crucifixo}=>{Afundo=Afundo}	0.0263157		1 0.0263157894736842	7.6	1
5 {Crucifixo=Crucifixo}=>{Gemeos=Gemeos}	0.0263157		1 0.0263157894736842	633.333.333.333.333	1
6 {Crucifixo=Crucifixo}=>{LegPress=LegPress}	0.0263157		1 0.0263157894736842	271.428.571.428.571	1
7 {Afundo=Gemeos}=>{Gemeos=AgachamentoSmith}	0.0526315		1 0.0526315789473684	3.8	2
8 {Afundo=Gemeos}=>{Crucifixo=Esteira}	0.0526315		1 0.0526315789473684	316.666.666.666.667	2
9 {Gemeos=Flexor}=>{Crucifixo=Esteira}	0.0526315		1 0.0526315789473684	316.666.666.666.667	2
10 {Gemeos=Flexor}=>{Afundo=Bicicleta}	0.0526315		1 0.0526315789473684	316.666.666.666.667	2

Lista de comandos:

```
#summary(dados)
```

```
set.seed(202483)
```

```
rules <- apriori(dados, parameter = list(supp = 0.001, conf = 0.7, minlen=2))
#summary(rules)
```

```
options(digits=2)
#inspect(sort(rules, by=c("confidence","support")))
```

```
rules_df <- as(rules,"data.frame")
write.csv(rules_df,"regras_Associacao.csv")
```

APÊNDICE 8 – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

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

CNN:

```

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

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

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

```

```

x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)
# Model ( lista entrada, lista saída)
model = Model(i, x)

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

Model: "model"

```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 32, 32, 3)]	0
conv2d (Conv2D)	(None, 15, 15, 32)	896
conv2d_1 (Conv2D)	(None, 7, 7, 64)	18496
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dropout (Dropout)	(None, 1152)	0
dense (Dense)	(None, 1024)	1180672
dropout_1 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 10)	10250

```

=====
Total params: 1,284,170
Trainable params: 1,284,170
Non-trainable params: 0
=====

print(tf.config.list_physical_devices())
[PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'),
PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]

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

```

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

Epoch 1/15
1563/1563 [=====] - 9s 4ms/step - loss: 1.5647
- accuracy: 0.4283 - val_loss: 1.3140 - val_accuracy: 0.5185
Epoch 2/15
1563/1563 [=====] - 6s 4ms/step - loss: 1.2792
- accuracy: 0.5422 - val_loss: 1.1444 - val_accuracy: 0.5865
Epoch 3/15
1563/1563 [=====] - 5s 3ms/step - loss: 1.1440
- accuracy: 0.5904 - val_loss: 1.0369 - val_accuracy: 0.6295
Epoch 4/15
1563/1563 [=====] - 6s 4ms/step - loss: 1.0514
- accuracy: 0.6236 - val_loss: 1.0127 - val_accuracy: 0.6411
Epoch 5/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.9767
- accuracy: 0.6510 - val_loss: 0.9221 - val_accuracy: 0.6752
Epoch 6/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.9113
- accuracy: 0.6745 - val_loss: 0.9016 - val_accuracy: 0.6862
Epoch 7/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.8639
- accuracy: 0.6927 - val_loss: 0.8673 - val_accuracy: 0.6959
Epoch 8/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.8150
- accuracy: 0.7100 - val_loss: 0.8753 - val_accuracy: 0.7013
Epoch 9/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.7798
- accuracy: 0.7218 - val_loss: 0.8655 - val_accuracy: 0.7008
Epoch 10/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.7435
- accuracy: 0.7350 - val_loss: 0.8608 - val_accuracy: 0.7011
Epoch 11/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.7201
- accuracy: 0.7441 - val_loss: 0.8302 - val_accuracy: 0.7136
Epoch 12/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.6865
- accuracy: 0.7533 - val_loss: 0.8149 - val_accuracy: 0.7153
Epoch 13/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.6647
- accuracy: 0.7618 - val_loss: 0.8303 - val_accuracy: 0.7188
```

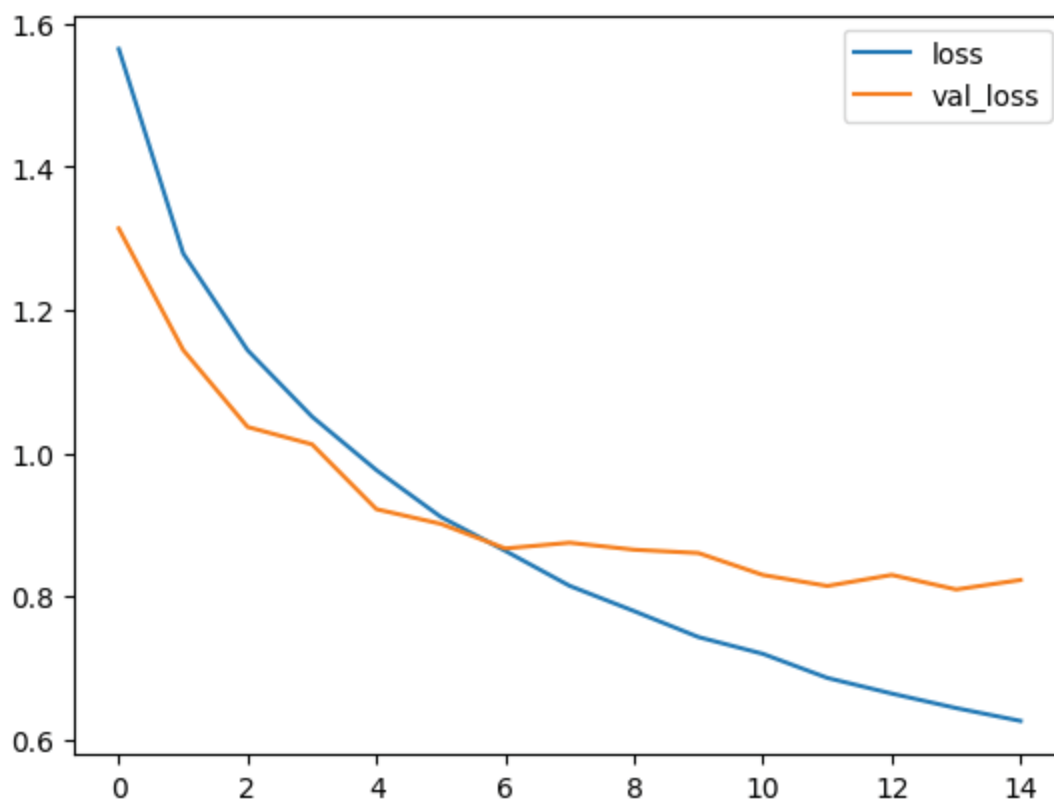
```

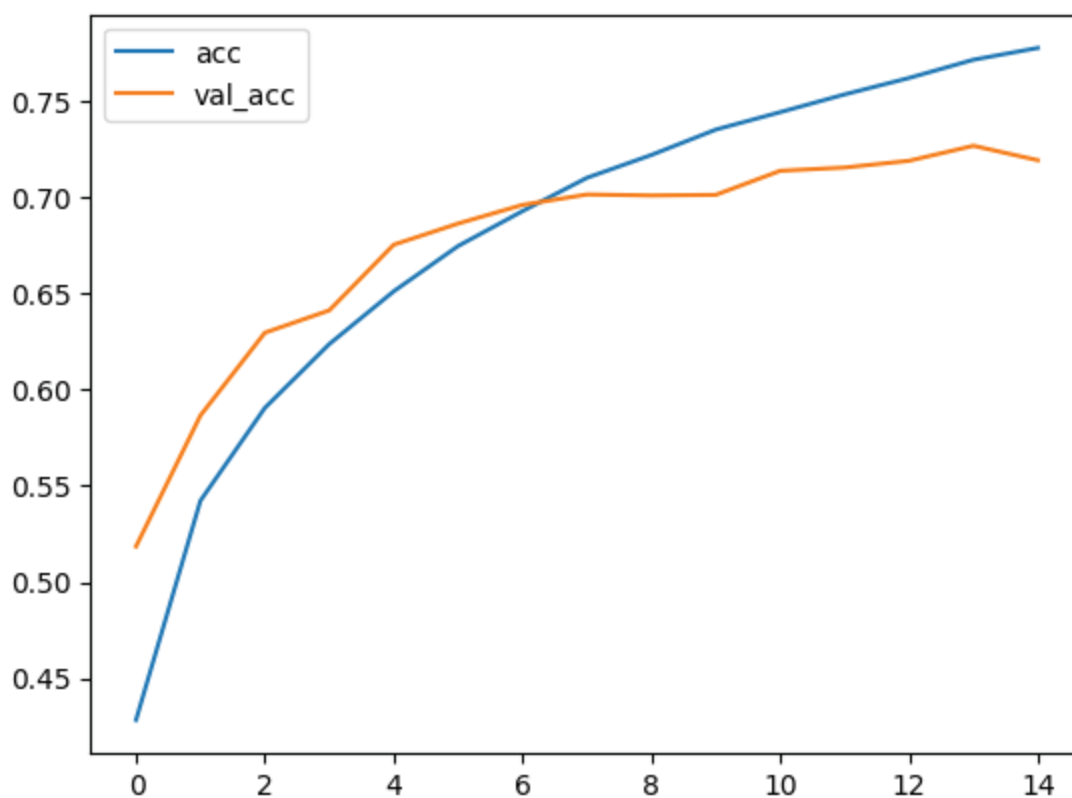
Epoch 14/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.6444
- accuracy: 0.7713 - val_loss: 0.8100 - val_accuracy: 0.7266
Epoch 15/15
1563/1563 [=====] - 6s 4ms/step - loss: 0.6264
- accuracy: 0.7774 - val_loss: 0.8232 - val_accuracy: 0.7191

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

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

```



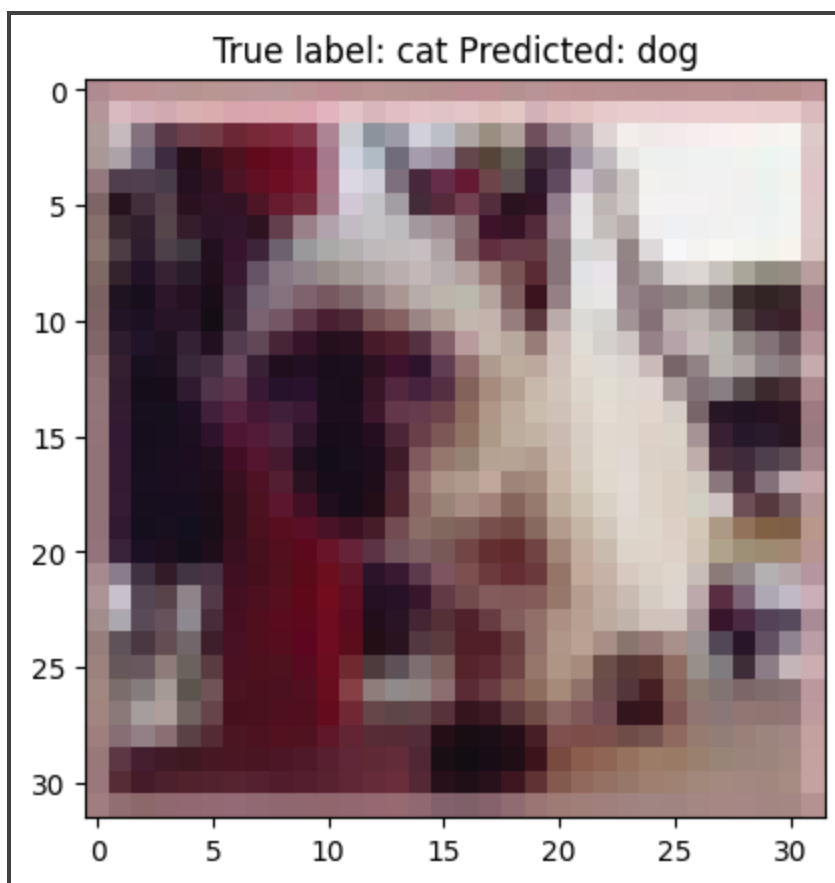


```
# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
show_normed=True)
```

true label	0	742 (0.74)	10 (0.01)	47 (0.05)	28 (0.03)	22 (0.02)	11 (0.01)	12 (0.01)	17 (0.02)	65 (0.07)	46 (0.05)
	1	17 (0.02)	766 (0.77)	2 (0.00)	11 (0.01)	4 (0.00)	3 (0.00)	12 (0.01)	10 (0.01)	45 (0.04)	130 (0.13)
	2	48 (0.05)	4 (0.00)	594 (0.59)	62 (0.06)	111 (0.11)	94 (0.09)	43 (0.04)	29 (0.03)	6 (0.01)	9 (0.01)
	3	14 (0.01)	7 (0.01)	58 (0.06)	538 (0.54)	69 (0.07)	205 (0.20)	59 (0.06)	29 (0.03)	11 (0.01)	10 (0.01)
	4	21 (0.02)	0 (0.00)	62 (0.06)	66 (0.07)	691 (0.69)	43 (0.04)	31 (0.03)	76 (0.08)	7 (0.01)	3 (0.00)
	5	8 (0.01)	2 (0.00)	43 (0.04)	187 (0.19)	49 (0.05)	647 (0.65)	15 (0.01)	39 (0.04)	5 (0.01)	5 (0.01)
	6	3 (0.00)	4 (0.00)	37 (0.04)	66 (0.07)	59 (0.06)	40 (0.04)	774 (0.77)	10 (0.01)	4 (0.00)	3 (0.00)
	7	10 (0.01)	2 (0.00)	18 (0.02)	35 (0.04)	55 (0.06)	79 (0.08)	8 (0.01)	785 (0.79)	3 (0.00)	5 (0.01)
	8	43 (0.04)	18 (0.02)	10 (0.01)	25 (0.03)	16 (0.02)	11 (0.01)	8 (0.01)	6 (0.01)	837 (0.84)	26 (0.03)
	9	30 (0.03)	48 (0.05)	9 (0.01)	25 (0.03)	5 (0.01)	16 (0.02)	10 (0.01)	19 (0.02)	21 (0.02)	817 (0.82)
		predicted label									

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

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



GAN:

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

Requirement already satisfied: imageio in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (2.35.1)
Requirement already satisfied: numpy in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from imageio)
(1.26.4)
Requirement already satisfied: pillow>=8.3.2 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from imageio)
(10.4.0)
Running command git clone --filter=blob:none --quiet
https://github.com/tensorflow/docs
'C:\Users\felip\AppData\Local\Temp\pip-req-build-5psmf5ar'
Collecting git+https://github.com/tensorflow/docs
  Cloning https://github.com/tensorflow/docs to
c:\users\felip\appdata\local\temp\pip-req-build-5psmf5ar
  Resolved https://github.com/tensorflow/docs to commit
d1ab9f4ece4bfd8b45679261931fec2d41959dd7
```

```

Preparing metadata (setup.py): started
Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: astor in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
tensorflow-docs==2024.7.15.51478) (0.8.1)
Requirement already satisfied: absl-py in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
tensorflow-docs==2024.7.15.51478) (2.1.0)
Requirement already satisfied: Jinja2 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
tensorflow-docs==2024.7.15.51478) (3.1.4)
Requirement already satisfied: nbformat in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
tensorflow-docs==2024.7.15.51478) (5.10.4)
Requirement already satisfied: protobuf>=3.12 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
tensorflow-docs==2024.7.15.51478) (3.19.6)
Requirement already satisfied: pyyaml in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
tensorflow-docs==2024.7.15.51478) (6.0.2)
Requirement already satisfied: MarkupSafe>=2.0 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
Jinja2->tensorflow-docs==2024.7.15.51478) (2.1.5)
Requirement already satisfied: fastjsonschema>=2.15 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
nbformat->tensorflow-docs==2024.7.15.51478) (2.20.0)
Requirement already satisfied: jsonschema>=2.6 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
nbformat->tensorflow-docs==2024.7.15.51478) (4.23.0)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
nbformat->tensorflow-docs==2024.7.15.51478) (5.7.2)
Requirement already satisfied: traitlets>=5.1 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
nbformat->tensorflow-docs==2024.7.15.51478) (5.14.3)
Requirement already satisfied: attrs>=22.2.0 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
jsonschema>=2.6->nbformat->tensorflow-docs==2024.7.15.51478) (24.2.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
jsonschema>=2.6->nbformat->tensorflow-docs==2024.7.15.51478)
(2023.12.1)

```

```

Requirement already satisfied: referencing>=0.28.4 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
jsonschema>=2.6->nbformat->tensorflow-docs==2024.7.15.51478) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
jsonschema>=2.6->nbformat->tensorflow-docs==2024.7.15.51478) (0.20.0)
Requirement already satisfied: platformdirs>=2.5 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
jupyter-core!=5.0.*,>=4.12->nbformat->tensorflow-docs==2024.7.15.51478)
(4.2.2)
Requirement already satisfied: pywin32>=300 in
c:\users\felip\anaconda3\envs\my_env\lib\site-packages (from
jupyter-core!=5.0.*,>=4.12->nbformat->tensorflow-docs==2024.7.15.51478)
(306)

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

# Carregar a base de dados
(train_images, train_labels), (_, _) =
tf.keras.datasets.mnist.load_data()

# Normalização
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5 # Normaliza entre [-1, 1]

# Gera o banco em partes e randomiza
BUFFER_SIZE = 60000
BATCH_SIZE = 256

# Cria o dataset (from_tensor_slices)
# Randomiza (shuffle)
# Combina elementos consecutivos em lotes (batch)
train_dataset =
tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).b
atch(BATCH_SIZE)

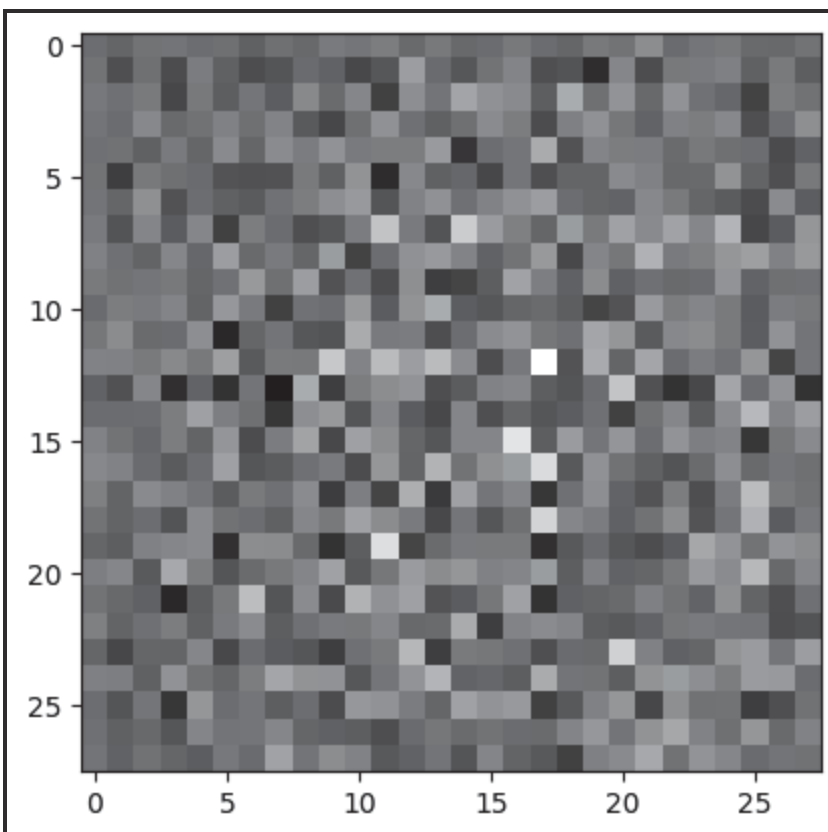
```

```

# Cria o GERADOR
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False,
input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())
    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)
# Note: None is the batch size
    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())
    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())
    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2),
padding='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)
    return model

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

```



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

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

tf.Tensor([[0.00025507]], shape=(1, 1), dtype=float32)
```

```

# Define as funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
# Perda do DISCRIMINADOR
def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss
# Perda do GERADOR
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)

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

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

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

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.Tensor: shape=(), dtype=float32, numpy=36.0>, <tf.Tensor:
shape=(), dtype=float32, numpy=10.0>]

# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

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

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

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

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

        generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))

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

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

        for image_batch in dataset:
            train_step(image_batch)

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

        # Save the model every 15 epochs
        if (epoch + 1) % 15 == 0:
```

```

        checkpoint.save(file_prefix = checkpoint_prefix)

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

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

# Gerar e salvar imagens
def generate_and_save_images(model, epoch, test_input):
    # Notice `training` is set to False.
    # This is so all layers run in inference mode (batchnorm).
    predictions = model(test_input, training=False)
    fig = plt.figure(figsize=(4, 4))
    for i in range(predictions.shape[0]):
        plt.subplot(4, 4, i+1)
        plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5,
cmap='gray')
        plt.axis('off')
    plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
    plt.show()

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

```



```

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

display_image(EPOCHS)
anim_file = 'dcgan.gif'

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

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

```



RNN:

```

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

# carrega e arruma a base
#!wget http://www.razer.net.br/datasets/spam.csv
df = pd.read_csv("C:\\Users\\felip\\OneDrive\\Documentos\\Deep
Learning\\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

# Separa a base em treino e teste
x_train, x_test, y_train, y_test = train_test_split(df["data"], y,
test_size=0.33)

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

7233 tokens

# Acerta o tamanho das sequências (padding)
data_train = pad_sequences(sequences_train) # usa o tamanho da maior
seq.

```

```

T = data_train.shape[1] # tamanho da sequência
data_test = pad_sequences(sequences_test, maxlen=T)
print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)

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

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

model.summary()
Model: "model"

```

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 189)]	0
embedding (Embedding)	(None, 189, 20)	144680
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6

```

Total params: 145,206
Trainable params: 145,206
Non-trainable params: 0

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

```

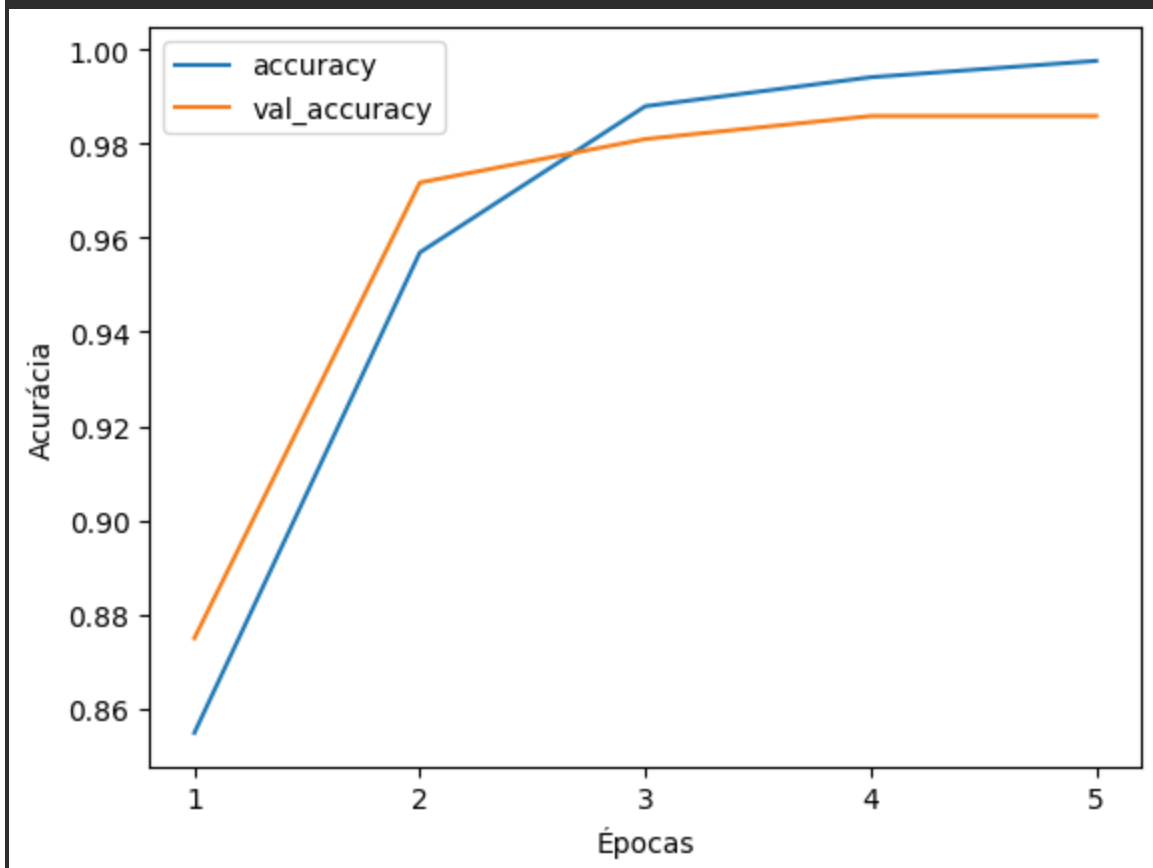
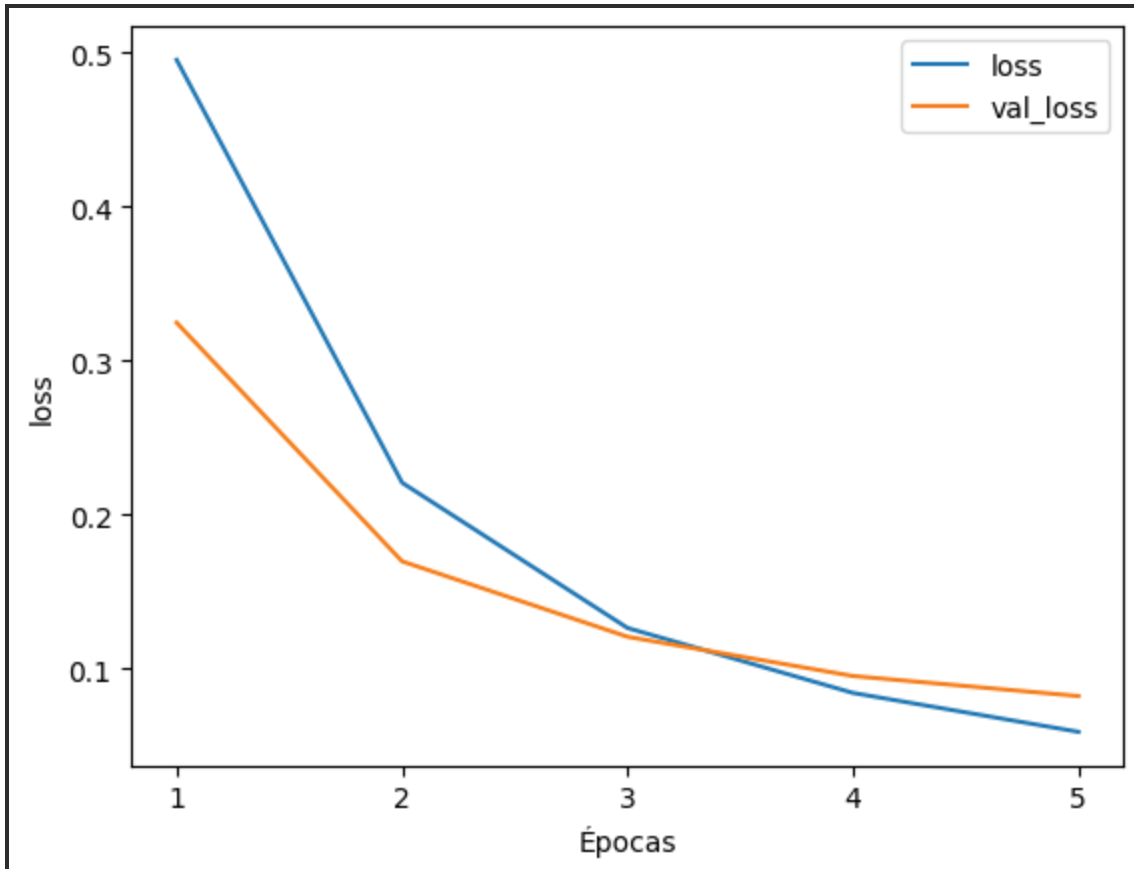
```

Epoch 1/5
117/117 [=====] - 8s 21ms/step - loss: 0.4955
- accuracy: 0.8548 - val_loss: 0.3248 - val_accuracy: 0.8749
Epoch 2/5
117/117 [=====] - 2s 17ms/step - loss: 0.2208
- accuracy: 0.9569 - val_loss: 0.1697 - val_accuracy: 0.9717
Epoch 3/5
117/117 [=====] - 2s 18ms/step - loss: 0.1264
- accuracy: 0.9879 - val_loss: 0.1207 - val_accuracy: 0.9810
Epoch 4/5
117/117 [=====] - 2s 17ms/step - loss: 0.0842
- accuracy: 0.9941 - val_loss: 0.0952 - val_accuracy: 0.9859
Epoch 5/5
117/117 [=====] - 2s 18ms/step - loss: 0.0589
- accuracy: 0.9976 - val_loss: 0.0822 - val_accuracy: 0.9859

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

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

```



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

1/1 [=====] - 0s 227ms/step
[[0.6726951]]
SPAM
```

Transformer:

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

Found existing installation: tensorflow 2.15.0
Uninstalling tensorflow-2.15.0:
  Would remove:
    /usr/local/bin/estimator_ckpt_converter
    /usr/local/bin/import_pb_to_tensorboard
    /usr/local/bin/saved_model_cli
    /usr/local/bin/tensorboard
    /usr/local/bin/tf_upgrade_v2
    /usr/local/bin/tflite_convert
    /usr/local/bin/toco
    /usr/local/bin/toco_from_protos

/usr/local/lib/python3.10/dist-packages/tensorflow-2.15.0.dist-info/*
/usr/local/lib/python3.10/dist-packages/tensorflow/*
Proceed (Y/n)? y
  Successfully uninstalled tensorflow-2.15.0
Collecting tensorflow==2.15.0
  Using cached
tensorflow-2.15.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.4 kB)
```

```

Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(1.26.4)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(24.1)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev
,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from
tensorflow==2.15.0) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(71.0.4)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(1.16.0)

```

```
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(0.37.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(1.64.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow==2.15.0)
(2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from
astunparse>=1.6.0->tensorflow==2.15.0) (0.44.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow==2.15.0) (1.2.1)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow==2.15.0) (3.7)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2.32.3)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow==2.15.0) (0.7.2)
```

```

Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow==2.15.0) (3.0.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from
google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow==2.15.0)
(5.5.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from
google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow==2.15.0)
(0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from
google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow==2.15.0)
(4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow==2.15
.0) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow==2.15.0)
(2024.7.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow==2.15.0) (2.1.5)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from
pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->
tensorflow==2.15.0) (0.6.0)

```

```

Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from
requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.1
6,>=2.15->tensorflow==2.15.0) (3.2.2)
Using cached
tensorflow-2.15.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (475.2 MB)
Installing collected packages: tensorflow
Successfully installed tensorflow-2.15.0
Requirement already satisfied: tensorflow_datasets in
/usr/local/lib/python3.10/dist-packages (4.9.6)
Requirement already satisfied: absl-py in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(1.4.0)
Requirement already satisfied: click in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(8.1.7)
Requirement already satisfied: dm-tree in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(0.1.8)
Requirement already satisfied: immutabledict in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(4.2.0)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(1.26.4)
Requirement already satisfied: promise in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(2.3)
Requirement already satisfied: protobuf>=3.20 in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(3.20.3)
Requirement already satisfied: psutil in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(5.9.5)
Requirement already satisfied: pyarrow in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(14.0.2)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(2.32.3)

```

```

Requirement already satisfied: simple-parsing in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(0.1.5)
Requirement already satisfied: tensorflow-metadata in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(1.15.0)
Requirement already satisfied: termcolor in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(2.4.0)
Requirement already satisfied: toml in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(0.10.2)
Requirement already satisfied: tqdm in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(4.66.5)
Requirement already satisfied: wrapt in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(1.14.1)
Requirement already satisfied: array-record>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow_datasets)
(0.5.1)
Requirement already satisfied: etils>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from
etils[enp,epath,epy,etree]>=1.6.0; python_version <
"3.11"->tensorflow_datasets) (1.7.0)
Requirement already satisfied: typing_extensions in
/usr/local/lib/python3.10/dist-packages (from
etils[enp,epath,epy,etree]>=1.6.0; python_version <
"3.11"->tensorflow_datasets) (4.12.2)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from
etils[enp,epath,epy,etree]>=1.6.0; python_version <
"3.11"->tensorflow_datasets) (2024.6.1)
Requirement already satisfied: importlib_resources in
/usr/local/lib/python3.10/dist-packages (from
etils[enp,epath,epy,etree]>=1.6.0; python_version <
"3.11"->tensorflow_datasets) (6.4.3)
Requirement already satisfied: zipp in
/usr/local/lib/python3.10/dist-packages (from
etils[enp,epath,epy,etree]>=1.6.0; python_version <
"3.11"->tensorflow_datasets) (3.20.0)

```

```

Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.19.0->tensorflow_datasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.19.0->tensorflow_datasets) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.19.0->tensorflow_datasets) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests>=2.19.0->tensorflow_datasets) (2024.7.4)
Requirement already satisfied: six in
/usr/local/lib/python3.10/dist-packages (from
promise->tensorflow_datasets) (1.16.0)
Requirement already satisfied: docstring-parser~=0.15 in
/usr/local/lib/python3.10/dist-packages (from
simple-parsing->tensorflow_datasets) (0.16)
Requirement already satisfied: tensorflow-text==2.15.0 in
/usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: tensorflow-hub>=0.13.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow-text==2.15.0)
(0.16.1)
Requirement already satisfied: tensorflow<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow-text==2.15.0)
(2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (0.2.0)

```

```

Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (1.26.4)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (24.1)
Requirement already satisfied:
protobuf!=4.21.0,!>=4.21.1,!>=4.21.2,!>=4.21.3,!>=4.21.4,!>=4.21.5,<5.0.0dev
,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (71.0.4)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (0.37.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (1.64.1)

```

```

Requirement already satisfied: tensorboard<2.16,>=2.15 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (2.15.0)
Requirement already satisfied: tf-keras>=2.14.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorflow-hub>=0.13.0->tensorflow-text==2.15.0) (2.15.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from
astunparse>=1.6.0->tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0)
(0.44.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0->tensorflow-text==2.1
5.0) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0->tensorflow-text==2.1
5.0) (1.2.1)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0->tensorflow-text==2.1
5.0) (3.7)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0->tensorflow-text==2.1
5.0) (2.32.3)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0->tensorflow-text==2.1
5.0) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0->tensorflow-text==2.1
5.0) (3.0.3)

```

```

Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from
google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.
0->tensorflow-text==2.15.0) (5.5.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from
google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.
0->tensorflow-text==2.15.0) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from
google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.
0->tensorflow-text==2.15.0) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow<2.16,
>=2.15.0->tensorflow-text==2.15.0) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0-
>tensorflow-text==2.15.0) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0-
>tensorflow-text==2.15.0) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0-
>tensorflow-text==2.15.0) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0-
>tensorflow-text==2.15.0) (2024.7.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow<2.16,>=2.15.0->ten
sorflow-text==2.15.0) (2.1.5)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from
pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->
tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (0.6.0)

```

```
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from
requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.1
6,>=2.15->tensorflow<2.16,>=2.15.0->tensorflow-text==2.15.0) (3.2.2)
```

```
import collections
import logging
import os
import re
import sys
import time

import numpy as np
import matplotlib.pyplot as plt

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

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

# Carregar a base de dados
examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en',
with_info=True, as_supervised=True)

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

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

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

e quando melhoramos a procura , tiramos a única vantagem da impressão ,
que é a serendipidade .

mas e se estes fatores fossem ativos ?

mas eles não tinham a curiosidade de me testar .

and when you improve searchability , you actually take away the one advantage of print , which is serendipity .

but what if it were active ?

but they did n't test for curiosity .

Tokenização e Destokenização do texto

model_name = "ted_hrlr_translate_pt_en_converter"

tf.keras.utils.get_file(f"{model_name}.zip",

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

Tem 2 tokenizers: um pt outro em en

tokenizers.en tokeniza e detokeniza

tokenizers = tf.saved_model.load(model_name)

PIPELINE DE ENTRADA

Codificar/tokenizar lotes de texto puro

def tokenize_pairs(pt, en):

pt = tokenizers.pt.tokenize(pt)

Converte ragged (irregular, tam variável) para dense

Faz padding com zeros.

pt = pt.to_tensor()

en = tokenizers.en.tokenize(en)

ragged -> dense

en = en.to_tensor()

return pt, en

Pipeline simples: processa, embaralha, agrupa os dados, prefetch

Datasets de entrada terminam com prefetch

BUFFER_SIZE = 20000

BATCH_SIZE = 64

def make_batches(ds):

return (

ds

.cache()

.shuffle(BUFFER_SIZE)

.batch(BATCH_SIZE)

.map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)

.prefetch(tf.data.AUTOTUNE))

train_batches = make_batches(train_examples)

val_batches = make_batches(val_examples)

```

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

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

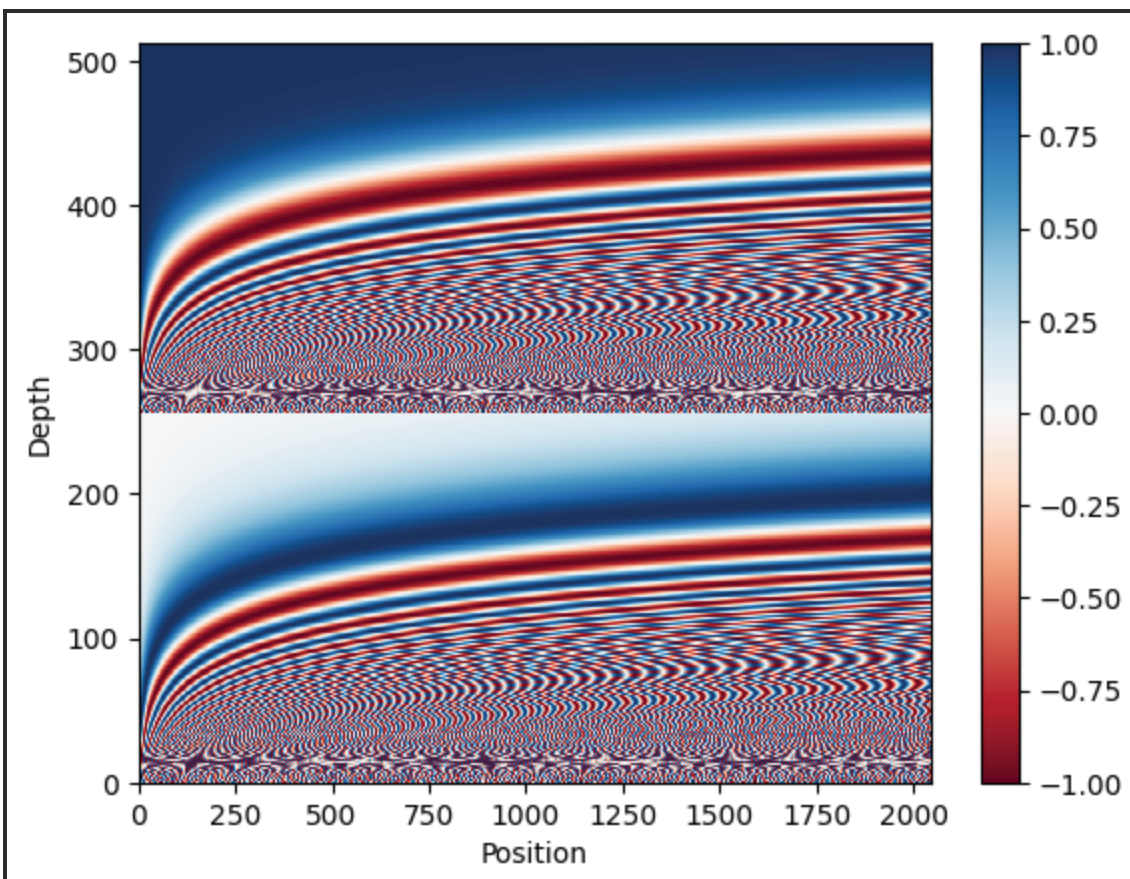
    # sin em índices pares no array; 2i
    angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])

    # cos em índices ímpares no array; 2i+1
    angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])

    # newaxis, aumenta a dimensão [] -> [ [] ]
    pos_encoding = angle_rads[np.newaxis, ...]
    return tf.cast(pos_encoding, dtype=tf.float32)

# CODIFICAÇÃO POSICIONAL
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]
# Arrumar as dimensões
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))
plt.pcolormesh(pos_encoding, cmap='RdBu')
plt.ylabel('Depth')
plt.xlabel('Position')
plt.colorbar()
plt.show()

```



```
# Cria uma máscara de 0 e 1, 0 para quando há valor e 1 quando não há
def create_padding_mask(seq):
    seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
    # add extra dimensions to add the padding
    # to the attention logits.
    return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_len)
# Máscara futura, usada no decoder
def create_look_ahead_mask(size):
    # zera o triângulo inferior
    mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
    return mask # (seq_len, seq_len)

# Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):
    # Q K^T
    matmul_qk = tf.matmul(q, k, transpose_b=True) # (... , seq_len_q,
    seq_len_k)
    # converte matmul_qk para float32
    dk = tf.cast(tf.shape(k)[-1], tf.float32)
    # divide por sqrt(d_k)
    scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
```

```

# Soma a máscara, e os valores faltantes serão um número próximo a
-inf
if mask is not None:
    scaled_attention_logits += (mask * -1e9)
# softmax normaliza os dados, somam 1. // (... , seq_len_q, seq_len_k)
attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
output = tf.matmul(attention_weights, v) # (... , seq_len_q, depth_v)
return output, attention_weights

# Atenção Multi-cabeças
class MultiHeadAttention(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.num_heads = num_heads
        self.d_model = d_model
        assert d_model % self.num_heads == 0
        self.depth = d_model // self.num_heads
        self.wq = tf.keras.layers.Dense(d_model)
        self.wk = tf.keras.layers.Dense(d_model)
        self.wv = tf.keras.layers.Dense(d_model)
        self.dense = tf.keras.layers.Dense(d_model)

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

    def call(self, v, k, q, mask):
        batch_size = tf.shape(q)[0]
        q = self.wq(q) # (batch_size, seq_len, d_model)
        k = self.wk(k) # (batch_size, seq_len, d_model)
        v = self.wv(v) # (batch_size, seq_len, d_model)

        q = self.split_heads(q, batch_size) # (batch_size, num_heads,
        seq_len_q, depth)
        k = self.split_heads(k, batch_size) # (batch_size, num_heads,
        seq_len_k, depth)
        v = self.split_heads(v, batch_size) # (batch_size, num_heads,
        seq_len_v, depth)
        # Calcula a atenção para cada cabeça (de forma matricial)

```

```

        # scaled_attention.shape == (batch_size, num_heads, seq_len_q,
depth)
        # attention_weights.shape == (batch_size, num_heads, seq_len_q,
seq_len_k)
        scaled_attention, attention_weights =
scaled_dot_product_attention(q, k, v, mask)
        # Troca a dimensão 2 com 1, para acertar o num_heads
        # (batch_size, seq_len_q, num_heads, depth)
        scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1,
3])
        # Concatena os valores em: (batch_size, seq_len_q, d_model)
        concat_attention = tf.reshape(scaled_attention, (batch_size, -1,
self.d_model))
        output = self.dense(concat_attention) # (batch_size, seq_len_q,
d_model)
        return output, attention_weights

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

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

    def call(self, x, training, mask):
        attn_output, _ = self.mha(x, x, x, mask) # (batch_size,
input_seq_len, d_model)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(x + attn_output) # (batch_size,
input_seq_len, d_model)
        ffn_output = self.ffn(out1) # (batch_size, input_seq_len, d_model)
        ffn_output = self.dropout2(ffn_output, training=training)

```

```

        out2 = self.layernorm2(out1 + ffn_output) # (batch_size,
input_seq_len, d_model)
        return out2

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

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

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

```

```

        self.embedding = tf.keras.layers.Embedding(input_vocab_size,
d_model)
        self.pos_encoding = positional_encoding(maximum_position_encoding,
self.d_model)
        self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate) for
_ in range(num_layers)]
        self.dropout = tf.keras.layers.Dropout(rate)

    def call(self, x, training, mask):
        seq_len = tf.shape(x)[1]
        # adding embedding and position encoding.
        x = self.embedding(x) # (batch_size, input_seq_len, d_model)
        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
        x += self.pos_encoding[:, :seq_len, :]
        x = self.dropout(x, training=training)
        for i in range(self.num_layers):
            x = self.enc_layers[i](x, training, mask)
        return x # (batch_size, input_seq_len, d_model)

class Decoder(tf.keras.layers.Layer):
    def __init__(self, num_layers, d_model, num_heads, dff,
target_vocab_size,
        maximum_position_encoding, rate=0.1):
        super(Decoder, self).__init__()
        self.d_model = d_model
        self.num_layers = num_layers
        self.embedding = tf.keras.layers.Embedding(target_vocab_size,
d_model)
        self.pos_encoding = positional_encoding(maximum_position_encoding,
d_model)
        self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate) for
_ in range(num_layers)]
        self.dropout = tf.keras.layers.Dropout(rate)

    def call(self, x, enc_output, training, look_ahead_mask,
padding_mask):
        seq_len = tf.shape(x)[1]
        attention_weights = {}
        x = self.embedding(x) # (batch_size, target_seq_len, d_model)
        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
        x += self.pos_encoding[:, :seq_len, :]
        x = self.dropout(x, training=training)
        for i in range(self.num_layers):

```

```

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

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

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

    def call(self, inputs, training):
        # Keras models prefer if you pass all your inputs in the first
        argument
        inp, tar = inputs
        enc_padding_mask, look_ahead_mask, dec_padding_mask =
self.create_masks(inp, tar)
        # (batch_size, inp_seq_len, d_model)
        enc_output = self.encoder(inp, training, enc_padding_mask)
        # dec_output.shape == (batch_size, tar_seq_len, d_model)
        dec_output, attention_weights = self.decoder(
        tar, enc_output, training, look_ahead_mask, dec_padding_mask)
        # (batch_size, tar_seq_len, target_vocab_size)
        final_output = self.final_layer(dec_output)
        return final_output, attention_weights

    def create_masks(self, inp, tar):
        # Encoder padding mask
        enc_padding_mask = create_padding_mask(inp)
        # Used in the 2nd attention block in the decoder.
        # This padding mask is used to mask the encoder outputs.
        dec_padding_mask = create_padding_mask(inp)
        # Used in the 1st attention block in the decoder.

```

```

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

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

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

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

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

loss_object =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask
    return tf.reduce_sum(loss_) / tf.reduce_sum(mask)

```

```

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

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

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

# Checkpoint
checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(transformer=transformer,
optimizer=optimizer)
ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path,
max_to_keep=5)
# if a checkpoint exists, restore the latest checkpoint.

if ckpt_manager.latest_checkpoint:
    ckpt.restore(ckpt_manager.latest_checkpoint)
    print('Latest checkpoint restored!!')

EPOCHS = 20
train_step_signature = [
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]
@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
    tar_inp = tar[:, :-1]
    tar_real = tar[:, 1:]

```

```

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

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

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

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

Epoch 1 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 650 Loss 0.0000 Accuracy 0.0000

```

```
Epoch 1 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 1 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 1 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 9.07 secs
```

```
Epoch 2 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 2 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 2 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.78 secs
```

```
Epoch 3 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 3 Batch 800 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 3 Loss 0.0000 Accuracy 0.0000
```

```
Time taken for 1 epoch: 10.56 secs
```

```
Epoch 4 Batch 0 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 50 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 100 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 150 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 200 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 250 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 300 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 350 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 400 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 450 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 500 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 550 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 600 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 650 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 700 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 750 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Batch 800 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 4 Loss 0.0000 Accuracy 0.0000
```

```
Time taken for 1 epoch: 10.57 secs
```

```
Epoch 5 Batch 0 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 50 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 100 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 150 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 200 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 250 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 300 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 350 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 400 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 450 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 500 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 550 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 600 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 650 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 700 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 750 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 5 Batch 800 Loss 0.0000 Accuracy 0.0000
```

```
Saving checkpoint for epoch 5 at ./checkpoints/train/ckpt-5
```

```
Epoch 5 Loss 0.0000 Accuracy 0.0000
```

```
Time taken for 1 epoch: 10.82 secs
```

```
Epoch 6 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 6 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 6 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.56 secs
```

```
Epoch 7 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 7 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 7 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.58 secs
```

```
Epoch 8 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 50 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 8 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 8 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 8 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 6.18 secs
```

```
Epoch 9 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 9 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 9 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 7.60 secs
```

```
Epoch 10 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 200 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 10 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 10 Batch 800 Loss 0.0000 Accuracy 0.0000
Saving checkpoint for epoch 10 at ./checkpoints/train/ckpt-6
Epoch 10 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.81 secs

Epoch 11 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 11 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 11 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 6.27 secs

Epoch 12 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 300 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 12 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 12 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 12 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 7.85 secs
```

```
Epoch 13 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 13 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 13 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 6.08 secs
```

```
Epoch 14 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 450 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 14 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 14 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 14 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 7.90 secs
```

```
Epoch 15 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 15 Batch 800 Loss 0.0000 Accuracy 0.0000
Saving checkpoint for epoch 15 at ./checkpoints/train/ckpt-7
Epoch 15 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 6.42 secs
```

```
Epoch 16 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 550 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 16 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 16 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 16 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.61 secs
```

```
Epoch 17 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 17 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 17 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.57 secs
```

```
Epoch 18 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 700 Loss 0.0000 Accuracy 0.0000
```

```
Epoch 18 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 18 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 18 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.59 secs
```

```
Epoch 19 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 19 Batch 800 Loss 0.0000 Accuracy 0.0000
Epoch 19 Loss 0.0000 Accuracy 0.0000
Time taken for 1 epoch: 10.59 secs
```

```
Epoch 20 Batch 0 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 50 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 100 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 150 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 200 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 250 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 300 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 350 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 400 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 450 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 500 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 550 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 600 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 650 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 700 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 750 Loss 0.0000 Accuracy 0.0000
Epoch 20 Batch 800 Loss 0.0000 Accuracy 0.0000
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-8
```

```
Epoch 20 Loss 0.0000 Accuracy 0.0000
```

```
Time taken for 1 epoch: 10.80 secs
```

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

    def __call__(self, sentence, max_length=20):
        # input sentence is portuguese, hence adding the start and end
        token
        assert isinstance(sentence, tf.Tensor)
        if len(sentence.shape) == 0:
            sentence = sentence[tf.newaxis]
        sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
        encoder_input = sentence
        # as the target is english, the first token to the transformer
        should be the
        # english start token.
        start_end = self.tokenizers.en.tokenize([' '])[0]
        start = start_end[0][tf.newaxis]
        end = start_end[1][tf.newaxis]
        output_array = tf.TensorArray(dtype=tf.int64, size=0,
dynamic_size=True)
        output_array = output_array.write(0, start)
        for i in tf.range(max_length):
            output = tf.transpose(output_array.stack())
            predictions, _ = self.transformer([encoder_input, output],
training=False)
            predictions = predictions[:, -1:, :] # (batch_size, 1,
vocab_size)
            predicted_id = tf.argmax(predictions, axis=-1)
            output_array = output_array.write(i+1, predicted_id[0])
            if predicted_id == end:
                break
        output = tf.transpose(output_array.stack())
        # output.shape (1, tokens)
        text = tokenizers.en.detokenize(output)[0]
        tokens = tokenizers.en.lookup(output)[0]
        _, attention_weights = self.transformer([encoder_input,
output[:, :-1]],
training=False)
        return text, tokens, attention_weights
```


APÊNDICE 9 – BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

INTRODUÇÃO

A era do Big Data trouxe a necessidade de ferramentas e diversas técnicas capazes de lidar com volumes massivos de dados, de tal maneira estruturados quanto não estruturados, gerados cada vez mais em altas velocidades. Nesse cenário, bancos de dados NoSQL e NewSQL apareceram como alternativas poderosas aos tradicionais bancos de dados relacionais. O crescimento exponencial dos dados gerados por usuários e dispositivos têm impulsionado a adoção de tecnologias de Big Data em diversos setores, demandando de certa forma soluções que garantam a eficiência e a personalização.

Este projeto apresenta a implementação de um sistema de recomendação em tempo real para uma plataforma de e-commerce, utilizando ferramentas de Big Data, incluindo bancos de dados NoSQL e NewSQL. O objetivo é melhorar a experiência do usuário através de recomendações personalizadas, aumentando a taxa de conversão e a fidelização de clientes, aproveitando a escalabilidade e a alta performance dessas tecnologias para lidar com grandes volumes de dados e garantir respostas rápidas em tempo real.

CARACTERIZAÇÃO DOS DADOS E OS 5 Vs DO BIG DATA

Ao implementar uma solução de Big Data, é essencial compreender as características dos dados, guiadas pelos 5Vs do Big Data:

1. **Volume:** A plataforma de e-commerce gera terabytes de dados diariamente. São milhões de usuários ativos, centenas de milhares de produtos, e interações constantes, como visualizações, cliques e compras. Esse grande volume exige sistemas de armazenamento e processamento escaláveis, com capacidade de crescer horizontalmente.
2. **Velocidade:** Os dados são gerados em tempo real, especialmente durante picos como promoções e eventos sazonais. O sistema precisa ser capaz de processar esses dados em alta velocidade para fornecer recomendações atualizadas sem causar atrasos que prejudiquem a experiência do usuário.
3. **Variedade:** Os dados no e-commerce apresentam grande variedade. Há dados estruturados, como registros de transações, semiestruturados, como logs de navegação e eventos de cliques, e não estruturados, como comentários e avaliações de clientes em redes sociais. Cada tipo de dado requer um tratamento específico, o que torna a flexibilidade das tecnologias NoSQL e NewSQL fundamental.
4. **Veracidade:** A veracidade dos dados é essencial para garantir que as recomendações geradas sejam relevantes e precisas. Dados imprecisos ou inconsistentes podem resultar em recomendações incorretas, prejudicando a confiança do cliente e o desempenho do sistema.
5. **Valor:** O valor está na capacidade de transformar dados brutos em insights acionáveis. A extração de valor dos dados, gerando recomendações personalizadas, pode aumentar significativamente as vendas e a satisfação do cliente, convertendo dados em vantagens competitivas.

Ferramentas Utilizadas

NoSQL - Flexibilidade e Escalabilidade: no contexto do sistema de recomendação, utilizamos o **Apache Cassandra**, um banco de dados NoSQL orientado a colunas, para armazenar dados de eventos de usuário em tempo real. Cassandra foi escolhido por sua capacidade de escalabilidade horizontal e alta disponibilidade, o que o torna ideal para lidar com grandes volumes de dados com baixa latência. A flexibilidade dos bancos NoSQL permite que eles sejam otimizados para tarefas específicas, como consultas de leitura rápidas e eficientes, essenciais para garantir a atualização constante das recomendações com base nas interações mais recentes dos usuários.

Além disso, o Cassandra se adapta bem a ambientes distribuídos, oferecendo alta resiliência e tolerância a falhas, crucial para garantir que o sistema de recomendação continue operando mesmo em caso de falhas em servidores individuais. Sua modelagem orientada a consultas permite que os dados sejam organizados de forma a otimizar as leituras, o que é fundamental para sistemas de recomendação que precisam responder rapidamente a perguntas como “quais produtos este usuário visualizou recentemente?”.

NewSQL - Consistência e Transações ACID: para os dados transacionais mais críticos, utilizamos o **Google Cloud Spanner**, um banco de dados **NewSQL**. Ele combina a escalabilidade dos bancos NoSQL com as garantias de consistência e integridade dos bancos de dados relacionais, oferecendo suporte a transações **ACID**. O Cloud Spanner é ideal para armazenar dados sensíveis, como informações de pedidos e gerenciamento de inventário, onde a consistência e a integridade são essenciais. Por exemplo, quando um cliente faz um pedido, é crucial garantir que os dados sejam consistentes, de modo a evitar problemas como sobre vendas de produtos que já estão fora de estoque.

Para os dados transacionais mais críticos, utilizamos o **Google Cloud Spanner**, um banco de dados **NewSQL**. Ele combina a escalabilidade dos bancos NoSQL com as garantias de consistência e integridade dos bancos de dados relacionais, oferecendo suporte a transações **ACID**. O Cloud Spanner é ideal para armazenar dados sensíveis, como informações de pedidos e gerenciamento de inventário, onde a consistência e a integridade são essenciais. Por exemplo, quando

um cliente faz um pedido, é crucial garantir que os dados sejam consistentes, de modo a evitar problemas como sobre vendas de produtos que já estão fora de estoque.

Modelagem de Dados: NoSQL vs. NewSQL

A modelagem de dados varia conforme as características do sistema e os tipos de dados que ele processa.

Modelagem NoSQL (Cassandra) : no Cassandra, a modelagem de dados é orientada às consultas, ou seja, os dados são organizados de maneira que atendam às consultas frequentes com a maior eficiência possível. Criamos tabelas desnormalizadas, otimizadas para leitura rápida:

- **Tabela de Eventos de Usuário:** Armazena eventos como visualizações, cliques e adições ao carrinho, com chaves compostas por ID do usuário e carimbos de data/hora, facilitando a consulta rápida dos comportamentos mais recentes dos usuários.
- **Tabela de Interações de Produto:** Registra interações de usuários com produtos, ajudando na análise em tempo real de produtos mais populares ou mais visualizados em uma determinada categoria.

Modelagem NewSQL (Cloud Spanner) : já no Cloud Spanner, seguimos um modelo relacional mais tradicional, que aproveita as transações ACID e a consistência garantida. Este modelo é aplicado a dados que exigem alta integridade:

- **Tabela de Pedidos:** Armazena informações detalhadas sobre cada pedido, incluindo dados de clientes, produtos adquiridos e status do pedido, garantindo que as transações sejam consistentes.
- **Tabela de Inventário:** Gerencia o estoque em tempo real, prevenindo vendas de produtos esgotados e garantindo que as atualizações de estoque sejam refletidas imediatamente em todas as partes do sistema.

Integração e Processamento de Dados

Para integrar e processar os dados, utilizamos o **Apache Kafka** como sistema de mensajería, transmitindo dados de eventos de usuários para o Cassandra em tempo real. O processamento é realizado com o **Apache Spark Streaming**, que consome esses dados e atualiza continuamente os modelos de recomendação. O uso de Kafka e Spark permite que o sistema processe grandes volumes de dados em tempo real, gerando recomendações que são constantemente refinadas com base nas interações dos usuários.

Benefícios da Abordagem

- **Escalabilidade:** O uso de Cassandra e Cloud Spanner permite escalabilidade horizontal, suportando o aumento da demanda sem perda de performance.
- **Baixa Latência:** A modelagem desnormalizada no Cassandra otimiza leituras rápidas, essenciais para fornecer recomendações em tempo real.
- **Consistência e Integridade:** O uso de NewSQL garante que as transações críticas, como pedidos e gerenciamento de inventário, sejam sempre consistentes e seguras.
- **Flexibilidade:** A combinação de tecnologias permite incorporar novos tipos de dados, como dados de redes sociais, e ajustar os modelos de recomendação conforme necessário.

CONCLUSÃO

A implementação de um sistema de recomendação em tempo real, utilizando uma combinação de ferramentas de Big Data como NoSQL e NewSQL, é uma solução robusta para lidar com os desafios apresentados pelos 5Vs do Big Data. Ao alavancar a escalabilidade e flexibilidade do NoSQL com a consistência e integridade dos bancos NewSQL, é possível fornecer recomendações personalizadas e em tempo real, melhorando a experiência do usuário e gerando valor para o negócio.

APÊNDICE 10 – VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

- 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

- g) Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.
- h) Indique qual modelo dá o melhor resultado e a métrica utilizada

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

Apresentar a resolução (somente o resultado) das questões do trabalho. Como é para apresentar somente o resultado segue o mesmos:

Definição do melhor modelo

1 - Extração de características

O melhor modelo para classificar imagens de biópsia de câncer de mama utiliza a **CNN VGG16** para extração de características e uma **Rede Neural Artificial (RNA)** para a classificação final. A VGG16, uma rede neural convolucional pré-treinada, é utilizada para extrair características profundas das imagens, capturando padrões visuais complexos. Em seguida, a RNA, com uma arquitetura simples de 2 a 3 camadas, processa esses vetores de características para classificar as imagens nas categorias correspondentes (0, 1+, 2+ e 3+). Este modelo é eficiente porque a VGG16 consegue capturar detalhes intrincados das imagens, enquanto a RNA se ajusta bem a esses padrões, oferecendo um excelente equilíbrio entre precisão e sensibilidade.

Métricas de Avaliação:

As principais métricas utilizadas para avaliar o desempenho deste modelo incluem:

- Sensibilidade: mede a capacidade do modelo de identificar corretamente os casos positivos, crucial para evitar falsos negativos em diagnósticos de câncer.
- Especificidade: Avalia a capacidade de detectar corretamente os casos negativos, importante para minimizar falsos positivos.
- F1-Score: Proporciona um equilíbrio entre precisão e sensibilidade, sendo especialmente relevante em cenários com classes desbalanceadas.
- AUC-ROC: Indica a capacidade do modelo em distinguir entre as classes positivas e negativas, refletindo sua eficácia geral.

Com base nessas métricas, este modelo é considerado o mais robusto para detectar o câncer de mama, superando abordagens mais simples, como Random Forest e SVM.

2 - Redes Neurais

O melhor modelo Rede Neural Convolucional (CNN), ideal para tarefas de classificação de imagens. A CNN é eficaz na captura de padrões visuais complexos, essenciais para diagnósticos médicos.

As principais métricas de avaliação são:

- Sensibilidade: Avalia a capacidade de identificar corretamente os casos positivos (importante em diagnósticos de câncer para evitar falsos negativos).
- F1-Score: Mede o equilíbrio entre precisão e sensibilidade, essencial quando há classes desbalanceadas.
- AUC-ROC: Indica o quão bem o modelo separa as classes positivas e negativas.

Esse modelo é altamente recomendado para cenários onde a precisão no diagnóstico visual é crucial, como em estudos no Brasil que utilizam imagens médicas para melhorar o diagnóstico e tratamento do câncer.

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

Parâmetros para elaboração do Trabalho:

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

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

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

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

essas discussões, incentivando os alunos a aplicar conceitos éticos e legais aprendidos ao analisar um caso concreto.

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

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

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

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

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

B – RESOLUÇÃO

RESUMO

Este trabalho aborda os desafios éticos e legais associados ao uso da inteligência artificial (IA), com foco na tecnologia ChatGPT. A mineração de dados, essencial para o desenvolvimento de modelos de IA, levanta questões críticas sobre privacidade e direitos autorais, em conflito com o Marco Civil da Internet e a Lei Geral de Proteção de Dados (LGPD). A aplicação da IA em contextos sociais, como democracia, mercado de trabalho e educação, requer uma análise aprofundada dos limites éticos. O estudo discute a privacidade dos dados, a transparência dos algoritmos, a responsabilidade pelas decisões automatizadas e o combate aos vieses, propondo soluções para um uso ético e responsável da IA. As reflexões são

embasadas em referências teóricas e casos práticos, visando contribuir para uma regulamentação justa e equilibrada das tecnologias de IA.

Palavras-chave: inteligência artificial, ética, privacidade, transparência, responsabilidade.

ABSTRACT

This paper addresses the ethical and legal challenges associated with the use of artificial intelligence (AI), focusing on ChatGPT technology. Data mining, essential for developing AI models, raises critical issues regarding privacy and copyright, conflicting with the Brazilian Civil Rights Framework for the Internet and the General Data Protection Law (LGPD). The application of AI in social contexts such as democracy, the labor market, and education requires an in-depth analysis of ethical boundaries. The study discusses data privacy, algorithm transparency, accountability for automated decisions, and combating biases, proposing solutions for the ethical and responsible use of AI. The reflections are based on theoretical references and practical cases, aiming to contribute to a fair and balanced regulation of AI technologies.

Keywords: artificial intelligence, ethics, privacy, transparency, accountability.

INTRODUÇÃO

O avanço da inteligência artificial (IA) tem provocado transformações significativas em diversas áreas do conhecimento e setores da sociedade. Tecnologias como o ChatGPT, desenvolvidas a partir de redes neurais treinadas com vastas quantidades de dados, exemplificam o potencial disruptivo e as oportunidades proporcionadas pela IA. No entanto, esses avanços trazem consigo uma série de desafios éticos e legais que precisam ser cuidadosamente analisados e enfrentados.

A mineração de dados, prática essencial para o desenvolvimento de modelos de IA, levanta questões importantes sobre privacidade e direitos autorais. A utilização de informações sem o devido consentimento contraria princípios

estabelecidos pelo Marco Civil da Internet e pela Lei Geral de Proteção de Dados (LGPD). Além disso, o impacto da IA em contextos sociais, como a democracia, o mercado de trabalho e a educação, exige uma reflexão profunda sobre os limites éticos do seu uso.

Este trabalho busca explorar os desafios e as implicações éticas do uso da IA, com foco particular no ChatGPT. Serão abordadas questões como a privacidade dos dados, a transparência dos algoritmos, a responsabilidade pelas decisões automatizadas e o combate aos vieses. O objetivo é propor soluções que promovam um uso ético e responsável da IA, equilibrando a inovação tecnológica com a proteção dos direitos individuais e coletivos

Ao longo deste estudo, serão discutidos os impactos da IA na sociedade contemporânea, baseando-se em referências teóricas e casos práticos. Espera-se que as reflexões apresentadas contribuam para um entendimento mais abrangente das responsabilidades éticas associadas ao desenvolvimento e aplicação de tecnologias de IA, apontando caminhos para uma regulamentação mais justa e equilibrada.

Limites éticos e legais do uso da IA na mineração de dados.

Aplicação do Marco Civil da Internet e da Lei Geral de Proteção de Dados

O ChatGPT funciona utilizando redes neurais que são treinadas com um volume enorme de informações e os modelos que geram os melhores resultados são mantidos enquanto aqueles com resultados ruins são descartados. O problema gerado por essa mecânica é que esse grande volume de informação, que é usado para treinar os modelos, muitas vezes são obras autorais, pertencem a alguém, ou até são informações que foram adquiridas através da mineração de dados, que é uma técnica de coletar informações de diversas fontes, sem necessariamente solicitar a autorização para o uso da informação. Por exemplo, para treinar um modelo é possível que seja feita uma mineração de dados na Web (Web Scrapping) que coletou informações diversas pessoas em um site de rede social, mas as informações que estavam disponíveis na rede, não foram publicadas para o uso da IA, elas haviam sido publicadas como forma de compartilhar remotamente com outras pessoas. Esse entre outros exemplos é um problema que pode levantar

questões sobre a ética e a legalidade do uso da informação, sendo que esse tipo de aplicação pode vir a ignorar preceitos defendidos pelo MCI e pela LGPD.

Desafios éticos da aplicação da IA em contextos sociais.

Uma ferramenta como o ChatGPT pode apresentar grandes desafios éticos no contexto social. Por exemplo, podemos discutir a maneira como ela pode influenciar a democracia de países, pois ela pode ser utilizada como uma ferramenta de lobby afetando assim o contexto político de nações. Outro exemplo é o contexto de seu efeito no mercado de trabalho, isso porque a maneira como o ChatGPT processa a linguagem natural, permite que ele realize muitas tarefas que costumavam ser executadas exclusivamente por humanos, alguns exemplos: ele pode resumir longos textos, traduzir textos de uma língua para outra, gerar conteúdo para páginas na internet e redes sociais, automatizar serviços de relacionamento com o cliente; sendo assim, atividades de trabalho como as citadas anteriormente podem vir a deixar de existir. A IA também pode impactar a grande divisão digital, isso decorre que apesar de existir pessoas com acesso à tecnologia e conhecimento, esse acesso não é uniforme, sendo que existe um número considerável de pessoas que não tem acesso a este tipo de tecnologia.

A maneira como o ChatGPT exerce sua influência na educação é algo que deve ser levado em consideração também pois ele pode tanto ajudar como criar problemas para educadores. Ele pode ajudar pois o ChatGPT possui uma ótima capacidade de sintetizar e resumir assuntos da maneira como ele processa a linguagem natural, ele pode ser uma ferramenta auxiliar aos professores, esclarecendo dúvidas que não foram abordadas durante as aulas, fornecendo exemplo e criando cenários que expliquem assuntos, introduzir os alunos a novos tópicos e hobbies e fomentar a curiosidade de maneira geral, pode ajudar professores na elaboração de tarefas e questões. No entanto, seu uso pode causar impactos negativos. Por exemplo, a redução na qualidade do ensino, ou seja, o uso constante de ferramentas como ChatGPT gerar uma estagnação do desenvolvimento do pensamento crítico e do entendimento de assuntos, as respostas geradas pelos modelos também nem sempre são confiáveis; outra questão é a diminuição da interação com professores e outros alunos, o uso frequente de uma ferramenta como ChatGPT pode fazer com que o professor

assuma um papel secundário na educação e isso pode gerar problemas de interação social e comunicação interpessoal; uma questão um tanto ética é sobre o uso de ChatGPT na elaboração de trabalhos e na questão de plágio, que também surge quando a ferramenta é usada no contexto acadêmico.

O isolamento social é outro problema que surge do uso de uma ferramenta como ChatGPT, isso porque o uso constante pode ser considerado viciante, como no caso de uma pessoa usa apenas o ChatGPT como fonte de informação e entretenimento. A maneira como o ChatGPT funciona permite a criação de aplicativos que emitem conversas com pessoas, é possível treinar a ferramenta para se comportar como um personagem ou uma celebridade e desse tipo de interação relacionamentos virtuais entre uma pessoa e a IA, outro exemplo de como o ChatGPT pode contribuir para o isolamento social.

De forma contrária, a eterna vigilância passa a ser um problema com o uso de uma tecnologia como o ChatGPT. A anos o uso de câmeras e tem se tornado mais comum, mas a maneira como estas eram usadas estava limitado a uma forma de registra acontecimentos, pois para usar vídeos de outra maneira, por exemplo para prevenir situações, para cada câmera seria necessária uma pessoa para interpretar o que estava acontecendo em tempo real e tomar uma decisão sobre o que estava acontecendo. Com o uso de ferramentas como ChatGPT, é possível utilizar modelos treinados para interpretar imagens e dessa forma identificar pessoas, gestos e ações, possibilitando assim a automatização de tomada de decisão no monitoramento de imagens. Mas isso pode levantar a questão: quem é responsável e o que acontece quando a IA errar?

O que fazer para evitar os impactos negativos?

Com todos esses desafios, o que pode ser feito para diminuir os impactos negativos da IA? O primeiro passo é trabalhar na educação e transparência, algo que acontece com as respostas do ChatGPT são as chamadas "alucinações", que são respostas que são geradas sobre um assunto que a IA não domina e dessa forma gera uma resposta incorreta. É importante educar quem faz uso da ferramenta que embora ela tenha várias qualidades, ela também pode ser imprecisa e por vezes suas respostas podem trazer algum viés, por isso ela deve ser utilizada com bastante cuidado. É algo que deve ser trabalhado também a questão de moderação

de conteúdo gerado, esse tipo de trabalho de moderação é importante para evitar problemas de viés e desinformação. O uso de "feedback" dos usuários também é uma forma de identificar momentos e assuntos que a ferramenta pode apresentar problemas e ajudar na calibração dos modelos. O estabelecimento de diretrizes e políticas éticas também é uma forma de buscar garantir que a ferramenta seja usada da maneira correta evitando a disseminação de desinformação.

Multimodalidade do uso da inteligência artificial

Desde recentes eleições em todo o globo, a inteligência artificial tem desafiado a forma como olhamos, analisamos e atuamos dentro da democracia moderna. Uma quantidade assustadora de notícias falsas endossadas por bots, robôs de softwares criados para interagir com posts humanos, tem assustado a todos, bem como tem plantado uma dúvida sobre a real veracidade das informações. Adicionar a isso tecnologias como as deep fakes, tecnologias extremamente poderosas para mimetizar rostos e vozes de pessoas chaves em nosso cenário global, fazem com que a gente não precise mais apenas duvidar do que estamos lendo, mas também do que estamos vendo e ouvindo. O livro “21 Lições para o Século 21” do historiador Yuval Noah Harari aborda questões relevantes sobre a inteligência artificial e seus impactos na sociedade. No segundo capítulo, Harari explora o impacto das redes sociais e da vida digital em nossa capacidade de formar verdadeiras comunidades. Ele discute como a tecnologia nos conecta, mas também pode nos fazer sentir mais solitários e desconectados. Além disso, o livro examina o desafio tecnológico e como os engenheiros do Facebook, por exemplo, poderiam usar a inteligência artificial para criar uma comunidade global que salvaguardasse a liberdade e a igualdade humanas.

Ao unir tais questões com provocações como as levantadas pelo livro “O Mal Estar da Pós-Modernidade” de Zygmunt Bauman, onde aborda questões relevantes sobre a sociedade contemporânea e suas inquietações. Bauman explora a liberdade individual como uma demanda pós-moderna. A busca pela liberdade alimenta as inquietações da era atual, diferentemente das preocupações anteriores que surgiam do desejo excessivo de controle e ordem. A pós-modernidade é marcada por um estado de insegurança e medo generalizado. Na sociedade pós-moderna, o mal-estar surge da desregulamentação do mundo. Bauman destaca que, ao

contrário da modernidade, onde a ordem era exaltada como uma realização desejável, hoje vivemos em um contexto de incertezas e ameaças constantes. Isso nos leva a atenção de como essas novas tecnologias podem nos trazer a um estado de apreensão, medo e desconfiança, sobretudo pelo desconhecido. Tecnologia como o ChatGPT podem também influenciar em como o mercado de trabalho se forma atualmente, transformando milhares de pessoas em inaptos para produzir em nossa sociedade de consumo dando lugar para trabalhadores ChatGPT, isso traz à tona discussões como a necessidade de uma renda universal. Harari, em seu livro, também discute o impacto da automação e da IA no mercado de trabalho. Ele alerta para a possibilidade de milhões de empregos serem substituídos por máquinas e como isso afetará a sociedade.

Direito à informação e liberdade de expressão.

A interseção entre direito de informação, liberdade de expressão e o avanço da Inteligência Artificial, apresenta desafios cruciais que demandam uma análise aprofundada e responsável para garantir o equilíbrio entre inovação tecnológica e proteção dos direitos individuais, pois esses quesitos são pilares democráticos fundamentais, essenciais para a garantia da participação cidadã e o exercício pleno da autonomia individual. No contexto da IA, esses direitos ganham novas dimensões e desafios, especialmente relacionados à disseminação de informações precisas e à diversidade de opiniões.

Embora a IA ofereça oportunidades sem precedentes para acessar e processar grandes volumes de dados, seu uso levanta questões éticas significativas. A privacidade dos dados pessoais, a segurança da informação e a responsabilidade legal pela tomada de decisões automáticas são temas críticos que exigem regulamentações claras e mecanismos de supervisão robustos.

No contexto do livro “Infocracia” Byung-chul Han, informa sobre a interseção entre inteligência artificial, direito à informação e liberdade de expressão, é imperativo não apenas vislumbrar os potenciais benefícios, mas também enfrentar os riscos e desafios emergentes. A falta de transparência nos algoritmos, a manipulação do conteúdo digital e a crescente polarização das opiniões representam preocupações fundamentais que exigem abordagens colaborativas e multidisciplinares.

Para promover um uso ético e responsável da IA, é essencial implementar políticas que garantam a transparência e a prestação de contas das plataformas digitais. Mecanismos de auditoria algorítmica, educação digital e engajamento público são pilares essenciais para fortalecer a confiança na tecnologia e proteger os direitos individuais e coletivos dos cidadãos.

Em síntese, enquanto a evolução do ChatGPT e da inteligência artificial oferecem oportunidades transformadoras para o direito à informação e a liberdade de expressão, ela também impõe responsabilidades éticas e legais que não podem ser subestimadas. A busca por soluções que equilibrem inovação tecnológica com valores democráticos fundamentais é crucial para garantir um futuro digital inclusivo e equitativo.

Relevância Ética

Com todos os aspectos já abordados sobre as aplicações de inteligência artificial (IA), em especial o ChatGPT, levanta inúmeras questões éticas importantes. As relevâncias éticas se manifestam na forma como essas tecnologias instigam e interagem com diferentes usuários, ou seja, parte interessadas, como desenvolvedores e a sociedade em geral.

Dessa forma é relevante trazer alguns dilemas éticos como por exemplo.

- **Privacidade dos usuários** - a coleta e utilização de dados pessoais pelo sistema desenvolvimento através de utilização de IA, sem qualquer limitação, podem invadir a privacidade dos usuários. Sendo possível coletar dados sensíveis, e ser usado para produtos (perfis, contas) detalhados, o que pode levar a uma violação da privacidade.
- **Transparência** - os algoritmos de IA, como por exemplo o ChatGPT, de certa forma são como caixas-pretas, ao qual suas decisões e processos internos não são transparentes. Isso levanta questões sobre a necessidade de aplicabilidade, em que os usuários devem entender como e por que tal decisão foi tomada.
- **Responsabilidade e Prestação de Contas** - com a crescente autonomia das IA identificar quem é responsável pelas ações e decisões da IA de certa forma se torna um dilema. Se alguma IA comete um erro ou causa danos, a

responsabilidade incide sobre os desenvolvedores, os operadores ou até mesmo o próprio sistema. Eis o viés que ainda de certa forma não tem decisão explícita, tudo depende do contexto a qual a mesma está sendo utilizada.

- **Viés e Discriminação** - os algoritmos de IA podem perpetuar ou até exacerbar vieses existentes na sociedade, resultando em discriminação contra certos grupos. Isso é especialmente preocupante quando a IA é usada em áreas críticas como recrutamento, policiamento ou até mesmo em serviços financeiros.

Contudo o ChatGPT pode ampliar a disseminação de informações falsas ou enganosas. Devido à sua capacidade de gerar texto coerente e convincente, sendo possível ser utilizado para espalhar desinformação intencionalmente ou imprudentemente. A ausência de mecanismos de verificação pode levar a uma maior propagação de notícias falsas, afetando a percepção pública e o processo democrático.

Ao qual levar o dilema da privacidade dos usuários, ao qual é uma preocupação central, especialmente quando o ChatGPT é treinado em grandes conjuntos de dados que podem incluir informações pessoais. O uso indevido desses dados pode resultar em violações de privacidade. Além disso, a falta de clareza sobre como os dados são armazenados e utilizados pode aumentar a desconfiança entre os usuários. Tornando assim um dilema de como esses dados são utilizados para tomadas de decisões de certa forma, decisões éticas. Sendo que a falta de um A falta de um framework ético robusto pode resultar em decisões que não consideram adequadamente os valores humanos fundamentais. Podendo assim levar a discriminações, sendo que esses dados de treinamento podem levar a resultados discriminatórios, afetando negativamente grupos minoritários. Estudos de caso mostraram que algoritmos de IA podem diferenciar injustamente com base em raça, gênero ou outras características demográficas.

Esses assuntos erguem uma situação de como tomar soluções responsáveis, para lidar com esses dilemas éticos, ao qual é fundamental propor soluções que provem o uso ético da IA. Um desses assuntos, podendo ser o principal, é aplicar políticas e regulamentações. Governo e organismos reguladores devem desenvolver políticas que garantam a transparência, a responsabilização e a equidade no uso de

IA. Isso inclui a criação de diretrizes claras sobre a coleta, armazenamento e uso de dados. Trazendo assim certas diretrizes de privacidade, em que as empresas devem seguir rigorosamente leis e proteção de dados, como a LGPD encontrada aqui no Brasil, para garantir que a privacidade dos usuários seja respeitada.

Essas políticas fazem com que as empresas responsáveis pelo desenvolvimento de aplicações de IA tomem algumas práticas de Design ético, como a transparência algorítmica, sendo que os desenvolvedores devem trabalhar para tornar os processos decisórios dos algoritmos mais transparentes. Isso pode incluir a criação de interfaces que expliquem claramente como e por que certas decisões foram tomadas. Outra prática também que pode ser encontrada é a implementação de métodos para identificar e mitigar vieses nos algoritmos. Sendo que isso pode envolver o uso de técnicas de auditoria e treinamento de IA em conjuntos de dados diversificados e equilibrados. Essas políticas bem aplicadas podem levar a garantir que os usuários sejam plenamente informados sobre como seus dados serão utilizados e obter seu consentimento explícito antes de qualquer coleta ou processamento de dados.

Contudo o uso de IA, em diversos contextos sociais e empresariais, traz à tona uma série de dilemas éticos que precisam ser cuidadosamente analisados e abordados. Por meio de políticas robustas, adoção de práticas de design ético, educação contínua e avaliação regular, é possível mitigar os riscos éticos associados à IA e promover um uso mais responsável e apropriado dessa tecnologia.

Dentro desse contexto ético as bolhas informacionais, criadas por algoritmos de personalização em plataformas online, levantam questões éticas significativas relacionadas à diversidade de informações e ao impacto social desses algoritmos. Ao mostrar apenas conteúdos que reforçam as crenças dos usuários, esses algoritmos podem prejudicar a diversidade de opiniões e impactar negativamente a sociedade. A sociedade informacional, caracterizada pela predominância das tecnologias de informação e comunicação, transforma profundamente a privacidade dos indivíduos, a equidade no acesso à informação e o poder de influência de grandes corporações tecnológicas. A neutralidade da rede, princípio que defende o tratamento igualitário de todos os dados na internet, é crucial para assegurar um ambiente digital justo, onde todos os usuários têm acesso igual às informações e serviços.

A existência dessas bolhas pode influenciar a disseminação de informações, muitas vezes espalhando desinformação. A privacidade dos usuários também é uma preocupação, pois os algoritmos coletam e utilizam dados pessoais para personalizar conteúdos. Esses algoritmos podem apresentar vieses, favorecendo certas informações e discriminando outras, levantando questões de responsabilidade sobre quem deve ser culpabilizado por possíveis danos causados.

Na sociedade informacional, a coleta e o uso massivo de dados pessoais levantam questões sobre privacidade e vigilância, enquanto a desigualdade no acesso às tecnologias pode exacerbar disparidades sociais. Vieses algorítmicos podem levar a discriminação e injustiças, e a responsabilidade por essas questões é complexa, envolvendo empresas, governos e indivíduos.

A ausência de neutralidade da rede pode levar à discriminação digital, onde certos conteúdos são favorecidos em detrimento de outros, afetando a liberdade de expressão, a inovação e a competição justa no mercado digital. A responsabilidade pela manutenção ou quebra da neutralidade recai sobre reguladores governamentais e provedores de serviços de internet, que devem ser monitorados para evitar abusos de poder.

Soluções Responsáveis

Para enfrentar esses desafios éticos, é necessário propor soluções que incluam políticas e regulamentações que promovam a transparência nos algoritmos de personalização e na proteção de dados. Práticas de design que favoreçam a diversidade de conteúdos e a inclusão de perspectivas variadas são fundamentais para mitigar os efeitos das bolhas informacionais.

Na sociedade informacional, a criação de políticas de proteção de dados rigorosas, regulamentações que garantam a transparência nos algoritmos e práticas de design que promovam a inclusão e a equidade no acesso à tecnologia são essenciais. Essas soluções devem equilibrar os interesses de todas as partes interessadas, promovendo valores éticos fundamentais como transparência, justiça e privacidade.

Para assegurar a neutralidade da rede, é essencial implementar regulamentações que proíbam a priorização paga de conteúdos e garantam a transparência nas práticas dos provedores de internet. Promover a educação dos

consumidores sobre seus direitos e incentivar a inovação justa e competitiva são passos fundamentais. Tais soluções devem equilibrar os interesses de todas as partes envolvidas, promovendo valores éticos como igualdade, transparência e liberdade de acesso à informação.

Conclusão:

A discussão sobre os limites éticos e legais do uso da Inteligência Artificial (IA), especialmente na mineração de dados, revela a necessidade urgente de regulamentação clara e políticas éticas robustas. Tecnologias como o ChatGPT, que utilizam grandes volumes de dados para treinamento, enfrentam críticas por possíveis violações de direitos autorais e privacidade.

A mineração de dados, sem a devida autorização, desafia princípios defendidos pelo Marco Civil da Internet e pela Lei Geral de Proteção de Dados (LGPD). Além das questões legais, a aplicação da IA em contextos sociais apresenta desafios éticos significativos. A influência da IA na democracia, no mercado de trabalho e na educação destaca a necessidade de um equilíbrio entre inovação tecnológica e proteção de direitos individuais.

Ferramentas como o ChatGPT têm o potencial de melhorar processos educacionais e empresariais, mas também podem perpetuar desigualdades e fomentar o isolamento social. Para mitigar os impactos negativos, é crucial investir em educação e transparência. Usuários devem ser informados sobre as limitações e potenciais vieses das ferramentas de IA.

A moderação de conteúdo e a utilização de feedback dos usuários são essenciais para melhorar a precisão e a responsabilidade das respostas geradas. Políticas éticas devem ser estabelecidas para garantir o uso adequado e evitar a disseminação de desinformação.

A relevância ética da IA reside na necessidade de transparência, responsabilidade e combate ao viés. A coleta de dados deve ser realizada com consentimento e utilizada de forma a proteger a privacidade dos usuários. A transparência nos algoritmos e a aplicabilidade das decisões são fundamentais para manter a confiança do público.

Por fim, a interseção entre direito à informação, liberdade de expressão e IA exige um enfoque multidisciplinar e colaborativo. A implementação de políticas que

assegurem a transparência e a prestação de contas é essencial para um futuro digital inclusivo e equitativo. A IA, quando utilizada de forma ética e responsável, tem o potencial de transformar positivamente a sociedade, mas requer vigilância contínua para evitar abusos e proteger os direitos fundamentais.

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Qual o objetivo do estudo descrito pelo artigo?	Qual problema/oportunidade/situação que levou à necessidade de realização desse estudo?	Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?	Quais os principais resultados obtidos pelo estudo?
O artigo explora a engenharia de requisitos (RE) em sistemas de inteligência artificial (IA), abordando as práticas atuais utilizadas e identificando diretrizes	A motivação para este estudo revela a falta de práticas consolidadas e de uma escassez de pesquisas sobre como a engenharia de requisitos (RE) para sistemas de IA deve	A metodologia utilizada pelos autores envolveu diversas etapas para obter uma compreensão abrangente das práticas de engenharia de requisitos (RE) em sistemas de IA.	Os principais resultados do estudo indicam que, embora a maioria dos profissionais concorde sobre a importância de abordar aspectos centrados no ser humano durante a

centradas no ser humano que podem melhorar o desenvolvimento desses sistemas. A pesquisa busca compreender como a RE está sendo aplicada na prática em IA, considerando os desafios específicos dessa área, como a complexidade e a imprevisibilidade dos algoritmos de aprendizado de máquina, e o impacto dessas tecnologias na vida dos usuários. Outro objetivo do estudo é identificar vácuos entre as práticas industriais e a literatura acadêmica, revelando aspectos da RE para IA (RE4AI) que ainda não foram suficientemente abordados ou que necessitam de diretrizes mais claras e acessíveis para os profissionais do setor. Isso inclui temas como ética, transparência, responsabilidade e usabilidade, que são fundamentais para um desenvolvimento de IA que respeite valores e necessidades humanas. Por fim, o artigo sugere que a redução desses vácuos pode contribuir para um alinhamento mais efetivo entre as práticas de engenharia de requisitos e os princípios centrados no ser humano, ajudando a construir sistemas de IA que sejam não apenas funcionais, mas também confiáveis, éticos e socialmente responsáveis.	incorporar aspectos centrados no ser humano. Embora grandes empresas, como Google, Microsoft e Apple, já possuam diretrizes voltadas para a criação de sistemas de IA mais humanizados, ainda existe um vácuo considerável quanto à aplicação dessas diretrizes na etapa de RE. Esse contexto revela uma oportunidade para desenvolver práticas e orientações mais detalhadas que ajudem a guiar o desenvolvimento de IA com foco nas necessidades, valores e expectativas dos usuários, promovendo soluções mais éticas, transparentes e confiáveis.	Primeiro, eles realizaram um levantamento com 29 profissionais da indústria para coletar informações sobre as ferramentas, práticas e desafios enfrentados na RE para IA. Em seguida, analisaram diretrizes de empresas líderes no setor, como Google e Microsoft, para entender os padrões já existentes voltados para o desenvolvimento human-centered. Além disso, realizaram um mapeamento dessas diretrizes em comparação com os estudos encontrados em uma revisão sistemática da literatura (SLR), permitindo identificar vácuos e convergências entre a teoria e a prática. Essa metodologia combinou dados empíricos com uma análise teórica, proporcionando uma visão detalhada das práticas e necessidades do mercado frente ao desenvolvimento de sistemas de IA centrados no ser humano.	engenharia de requisitos (RE) para IA, as ferramentas e métodos atualmente utilizados, como UML e pacotes do Office, mostram-se insuficientes para capturar os requisitos específicos da IA. O estudo revelou que muitos aspectos human-centered, essenciais para sistemas de IA, como a explicabilidade, a transparência e o gerenciamento de erros, não são adequadamente tratados na fase de RE. Essa limitação reflete um vácuo entre as necessidades práticas dos profissionais e as capacidades das ferramentas e métodos existentes, destacando a urgência de desenvolver abordagens específicas para atender a esses requisitos únicos da IA e promover o alinhamento com valores humanos.
--	--	--	---

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

Informações:

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

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

2 Regressão (RNA)

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

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

Informações:

- **Base de dados:** Wine Quality
- **Descrição:** O objetivo deste dataset prever a qualidade dos vinhos com base em suas características químicas. A variável target (y) neste exemplo será o

score de qualidade do vinho, que varia de 0 (pior qualidade) a 10 (melhor qualidade)

- **Tamanho:** 1599 amostras, 12 features.
- **Importação:** Copiar código abaixo.

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

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

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

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

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

3 Sistemas de Recomendação

Implementar o exemplo de Sistemas de Recomendação usando a base de dados Base_livos.csv e a arquitetura vista na aula **FRA - Aula 22 - 4.3 Resolução**

do Exercício de Sistemas de Recomendação. Além disso, fazer uma breve explicação dos seguintes resultados:

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

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

B – RESOLUÇÃO

1. Classificação (RNA)

```
# Importações necessárias
import tensorflow as tf
import numpy as np
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten, Dense
import matplotlib.pyplot as plt

# Carregar a base de dados Fashion MNIST
fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# Normalizar os dados para o intervalo [0, 1]
x_train = x_train / 255.0
x_test = x_test / 255.0

# Definir a arquitetura da rede neural
model = Sequential([
    Flatten(input_shape=(28, 28)), # Camada de entrada (achatar a
    imagem 28x28)
    Dense(128, activation='relu'), # Camada oculta com 128 neurônios e
    ReLU
    Dense(10, activation='softmax') # Camada de saída (10 classes)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Treinar o modelo
history = model.fit(x_train, y_train, epochs=10, validation_split=0.2)

Epoch 1/10
1500/1500 ————— 7s 4ms/step - accuracy:
0.7715 - loss: 0.6599 - val_accuracy: 0.8493 - val_loss: 0.4171
Epoch 2/10
1500/1500 ————— 10s 4ms/step - accuracy:
0.8575 - loss: 0.4002 - val_accuracy: 0.8605 - val_loss: 0.3866
Epoch 3/10
1500/1500 ————— 8s 5ms/step - accuracy:
0.8726 - loss: 0.3513 - val_accuracy: 0.8712 - val_loss: 0.3554
Epoch 4/10
1500/1500 ————— 6s 4ms/step - accuracy:
0.8856 - loss: 0.3180 - val_accuracy: 0.8789 - val_loss: 0.3319
```

```

Epoch 5/10
1500/1500 _____ 7s 5ms/step - accuracy:
0.8902 - loss: 0.3016 - val_accuracy: 0.8800 - val_loss: 0.3389
Epoch 6/10
1500/1500 _____ 6s 4ms/step - accuracy:
0.8958 - loss: 0.2805 - val_accuracy: 0.8783 - val_loss: 0.3454
Epoch 7/10
1500/1500 _____ 8s 5ms/step - accuracy:
0.8965 - loss: 0.2752 - val_accuracy: 0.8773 - val_loss: 0.3342
Epoch 8/10
1500/1500 _____ 6s 4ms/step - accuracy:
0.9074 - loss: 0.2573 - val_accuracy: 0.8878 - val_loss: 0.3177
Epoch 9/10
1500/1500 _____ 10s 4ms/step - accuracy:
0.9083 - loss: 0.2482 - val_accuracy: 0.8813 - val_loss: 0.3423
Epoch 10/10
1500/1500 _____ 7s 5ms/step - accuracy:
0.9086 - loss: 0.2428 - val_accuracy: 0.8839 - val_loss: 0.3388
# Avaliar o modelo
test_loss, test_acc = model.evaluate(x_test, y_test)
print(f"Acurácia no conjunto de teste: {test_acc:.2f}")

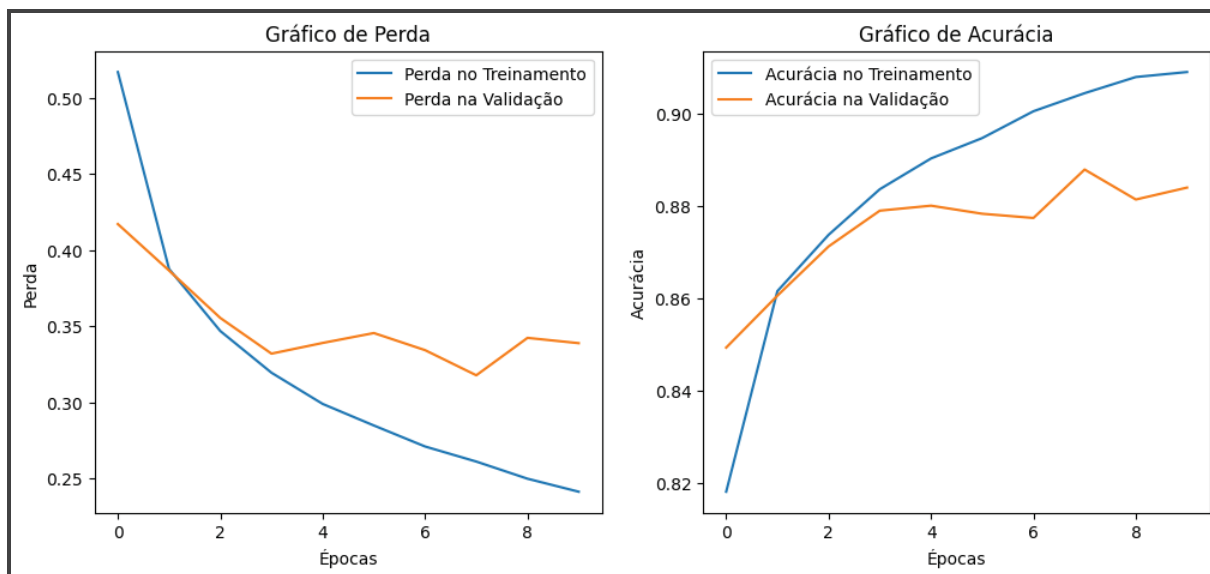
# Plotar gráficos de perda e acurácia
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Perda no Treinamento')
plt.plot(history.history['val_loss'], label='Perda na Validação')
plt.xlabel('Épocas')
plt.ylabel('Perda')
plt.legend()
plt.title('Gráfico de Perda')

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Acurácia no Treinamento')
plt.plot(history.history['val_accuracy'], label='Acurácia na
Validação')
plt.xlabel('Épocas')
plt.ylabel('Acurácia')
plt.legend()
plt.title('Gráfico de Acurácia')

plt.show()

```



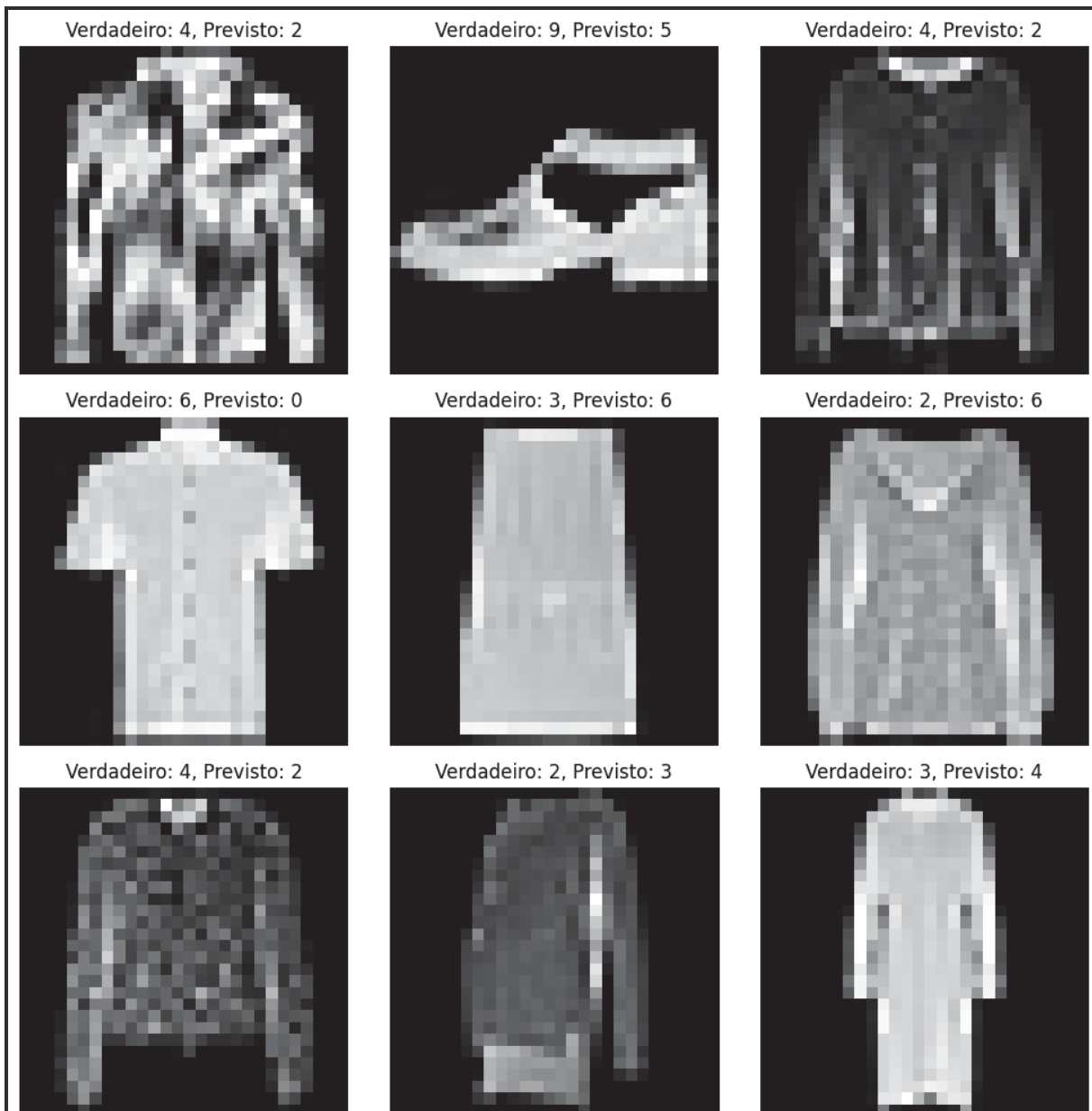
```

predictions = np.argmax(model.predict(x_test), axis=1)
incorrect = np.where(predictions != y_test)[0]

plt.figure(figsize=(10, 10))
for i, idx in enumerate(incorrect[:9]): # Mostrar as primeiras 9
    classificações erradas
    plt.subplot(3, 3, i + 1)
    plt.imshow(x_test[idx], cmap='gray')
    plt.title(f"Verdadeiro: {y_test[idx]}, Previsto:
{predictions[idx]}")
    plt.axis('off')
plt.tight_layout()
plt.show()

]z

```



Explicação dos resultados:

Gráficos de perda e acurácia:

O gráfico de perda mostra como o erro do modelo diminui ao longo do treinamento e da validação.

O gráfico de acurácia ilustra a melhoria do modelo em classificar corretamente as imagens ao longo das épocas.

Resultados Gráfico Perda e Acurácia: Gráfico de perda: Mostra como o erro do modelo diminui ao longo do treinamento e da validação. Gráfico de acurácia: Ilustra a melhoria do modelo em classificar corretamente as imagens ao longo das épocas.

2. Regressão (RNA)

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt

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

# Renomear as colunas para português
data.columns = [
    'acidez_fixa', 'acidez_volatil', 'acido_citrico',
    'acucar_residual',
    'cloretos', 'dioxido_de_enxofre_livre', 'dioxido_de_enxofre_total',
    'densidade', 'pH', 'sulfatos', 'alcool', 'score_qualidade_vinho']

# Separação dos dados (x e y)
X = data.iloc[:, :-1] # Todas as colunas exceto a última
y = data.iloc[:, -1]  # Última coluna (target)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Construção do modelo RNA
model = Sequential([
    Dense(64, input_dim=X_train_scaled.shape[1], activation='relu'),
```

```

        Dense(32, activation='relu'),
        Dense(1) # Saída para regressão
    ])

# Compilação do modelo
model.compile(optimizer='adam',                loss='mean_squared_error',
metrics=['mae'])

# Treinar o modelo
history = model.fit(X_train_scaled, y_train, epochs=50,
validation_split=0.2, batch_size=32, verbose=0)

# Avaliação do modelo
y_pred = model.predict(X_test_scaled)

# Métricas de avaliação
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Resultados
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
print(f"R2 Score: {r2:.2f}")

# Gráficos de avaliação
plt.figure(figsize=(18, 5))

# Gráfico de Loss
plt.subplot(1, 3, 1)
plt.plot(history.history['loss'], label='Loss Treino')
plt.plot(history.history['val_loss'], label='Loss Validação')
plt.title('Gráfico de Loss')
plt.xlabel('Épocas')
plt.ylabel('Loss')
plt.legend()

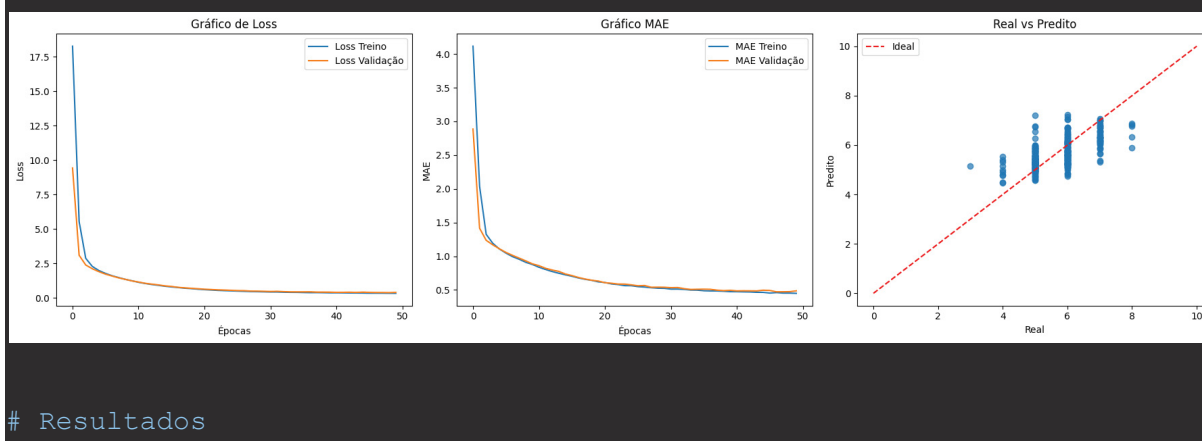
# Gráfico de MAE
plt.subplot(1, 3, 2)
plt.plot(history.history['mae'], label='MAE Treino')
plt.plot(history.history['val_mae'], label='MAE Validação')
plt.title('Gráfico MAE')
plt.xlabel('Épocas')
plt.ylabel('MAE')

```

```
plt.legend()

# Gráfico de Predição vs Real
plt.subplot(1, 3, 3)
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([0, 10], [0, 10], 'r--', label='Ideal')
plt.title('Real vs Predito')
plt.xlabel('Real')
plt.ylabel('Predito')
plt.legend()

plt.tight_layout()
plt.show()
```



```
# Resultados
```

Gráfico de Loss:

- O gráfico de loss mostra a redução do erro durante as épocas de treinamento para os conjuntos de treino e validação. Uma boa curva deve apresentar redução consistente do erro, sem grandes diferenças entre as curvas de treino e validação (o que indicaria overfitting).

Métricas de Avaliação:

- MSE (Mean Squared Error): Mede o erro quadrático médio, sendo sensível a grandes diferenças entre previsões e valores reais. MAE (Mean Absolute Error): Indica o erro médio absoluto, mais robusto contra outliers
- Gráfico de MAE: Adicionado para monitorar o erro absoluto médio durante o treinamento e validação. Este gráfico ajuda a entender como o modelo melhora ao longo das épocas em termos de erro absoluto médio. R^2 Score:

Avalia a proporção da variância nos dados explicada pelo modelo. Valores próximos a 1 indicam bom desempenho.

Gráfico Real vs Predito:

- Este gráfico avalia visualmente a qualidade das previsões. Quanto mais alinhados os pontos estiverem na linha ideal (reta vermelha), melhor o desempenho do modelo.

3. Sistemas de Recomendação

```
# Importações necessárias
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Flatten, Dot, Dense
import matplotlib.pyplot as plt
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

# Carregar a base de dados
#data = pd.read_csv('drive/MyDrive/code/ufpr/base_livros.csv')
data = pd.read_csv('drive/MyDrive/code/ufpr/heineken.csv',
delimiter=",")

# Pré-visualizar os dados
print(data.head())
```

	ISBN	Titulo \			
0	2005018	Clara Callan			
1	60973129	Decision in Normandy			
2	374157065	Flu: The Story of the Great Influenza Pandemic...			
3	393045218	The Mummies of Urumchi			
4	399135782	The Kitchen God's Wife			
	Autor	Ano	Editora	ID_usuario	Notas
0	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0
1	Carlo D'Este	1991	HarperPerennial	276726	2

2	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6
3	E. J. W. Barber	1999	W. W. Norton & Company	276729	1
4	Amy Tan	1991	Putnam Pub Group	276729	9

```

# Filtrar as colunas relevantes
data = data[['ID_usuario', 'ISBN', 'Notas']] # ID_usuario, ISBN e
Notas são os campos necessários

# Criar índices únicos para usuários e livros
user_ids = data['ID_usuario'].unique()
book_ids = data['ISBN'].unique()

user_to_index = {id_: index for index, id_ in enumerate(user_ids)}
book_to_index = {id_: index for index, id_ in enumerate(book_ids)}

data['user_idx'] = data['ID_usuario'].map(user_to_index)
data['book_idx'] = data['ISBN'].map(book_to_index)

# Converter 'Notas' para numérico, tratando erros
data['Notas'] = pd.to_numeric(data['Notas'], errors='coerce')

# Substituir NaN por 0
data['Notas'] = data['Notas'].fillna(0)

# Converter para int64
data['Notas'] = data['Notas'].astype('int64')

print(data.dtypes)
print(data[['user_idx', 'book_idx', 'Notas']].head())

<ipython-input-6-e02f513060e1>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
data['user_idx'] = data['ID_usuario'].map(user_to_index)
ID_usuario    object
ISBN           object
Notas          int64
user_idx       int64

```

```

book_idx      int64
dtype: object
   user_idx  book_idx  Notas
0         0         0      0
1         1         1      2
2         2         2      6
3         3         3      1
4         3         4      9

# Dividir os dados em conjuntos de treinamento e teste
train, test = train_test_split(data, test_size=0.2, random_state=42)

# Número de usuários e livros únicos
num_users = len(user_ids)
num_books = len(book_ids)

# Construir o modelo de recomendação
embedding_dim = 50 # Dimensão dos embeddings

# Inputs
user_input = Input(shape=(1,), name='user_input')
book_input = Input(shape=(1,), name='book_input')

# Embeddings
user_embedding = Embedding(input_dim=num_users,
output_dim=embedding_dim, name='user_embedding')(user_input)
book_embedding = Embedding(input_dim=num_books,
output_dim=embedding_dim, name='book_embedding')(book_input)

# Flatten embeddings
user_vec = Flatten()(user_embedding)
book_vec = Flatten()(book_embedding)

# Similaridade (dot product)
dot_product = Dot(axes=1)([user_vec, book_vec])

# Rede adicional para melhorar o aprendizado
dense = Dense(128, activation='relu')(dot_product)
output = Dense(1)(dense)

# Modelo
model = Model(inputs=[user_input, book_input], outputs=output)
model.compile(optimizer='adam', loss='mse', metrics=['mse'])

```

```
# Treinar o modelo
history = model.fit([train['user_idx'], train['book_idx']],
                    train['Notas'],
                    validation_data=([test['user_idx'],
test['book_idx']], test['Notas']),
                    epochs=20, batch_size=64)
```

Epoch 1/20

```
1612/1612 _____ 127s 77ms/step - loss:
614236288.0000 - mse: 614236288.0000 - val_loss: 703816320.0000 -
val_mse: 703816320.0000
```

Epoch 2/20

```
1612/1612 _____ 130s 81ms/step - loss:
638635392.0000 - mse: 638635392.0000 - val_loss: 699345408.0000 -
val_mse: 699345408.0000
```

Epoch 3/20

```
1612/1612 _____ 119s 67ms/step - loss:
621538944.0000 - mse: 621538944.0000 - val_loss: 693781312.0000 -
val_mse: 693781312.0000
```

Epoch 4/20

```
1612/1612 _____ 151s 73ms/step - loss:
534436480.0000 - mse: 534436480.0000 - val_loss: 690248128.0000 -
val_mse: 690248128.0000
```

Epoch 5/20

```
1612/1612 _____ 100s 62ms/step - loss:
553144832.0000 - mse: 553144832.0000 - val_loss: 688933376.0000 -
val_mse: 688933376.0000
```

Epoch 6/20

```
1612/1612 _____ 155s 70ms/step - loss:
493273792.0000 - mse: 493273792.0000 - val_loss: 689249536.0000 -
val_mse: 689249536.0000
```

Epoch 7/20

```
1612/1612 _____ 130s 63ms/step - loss:
425677728.0000 - mse: 425677728.0000 - val_loss: 691812096.0000 -
val_mse: 691812096.0000
```

Epoch 8/20

```
1612/1612 _____ 138s 61ms/step - loss:
374548608.0000 - mse: 374548608.0000 - val_loss: 696677184.0000 -
val_mse: 696677184.0000
```

Epoch 9/20

```

1612/1612 _____ 123s 76ms/step - loss:
278115680.0000 - mse: 278115680.0000 - val_loss: 704711680.0000 -
val_mse: 704711680.0000
Epoch 10/20
1612/1612 _____ 117s 72ms/step - loss:
208520688.0000 - mse: 208520688.0000 - val_loss: 712691712.0000 -
val_mse: 712691712.0000
Epoch 11/20
1612/1612 _____ 124s 62ms/step - loss:
176405520.0000 - mse: 176405520.0000 - val_loss: 721099328.0000 -
val_mse: 721099328.0000
Epoch 12/20
1612/1612 _____ 159s 72ms/step - loss:
117752312.0000 - mse: 117752312.0000 - val_loss: 731522496.0000 -
val_mse: 731522496.0000
Epoch 13/20
1612/1612 _____ 142s 73ms/step - loss:
103215312.0000 - mse: 103215312.0000 - val_loss: 719590528.0000 -
val_mse: 719590528.0000
Epoch 14/20
1612/1612 _____ 124s 62ms/step - loss:
70244712.0000 - mse: 70244712.0000 - val_loss: 726283840.0000 -
val_mse: 726283840.0000
Epoch 15/20
1612/1612 _____ 142s 61ms/step - loss:
47454980.0000 - mse: 47454980.0000 - val_loss: 732251200.0000 -
val_mse: 732251200.0000
Epoch 16/20
1612/1612 _____ 159s 72ms/step - loss:
38411040.0000 - mse: 38411040.0000 - val_loss: 720774592.0000 -
val_mse: 720774592.0000
Epoch 17/20
1612/1612 _____ 144s 73ms/step - loss:
25618386.0000 - mse: 25618386.0000 - val_loss: 722471616.0000 -
val_mse: 722471616.0000
Epoch 18/20
1612/1612 _____ 124s 62ms/step - loss:
21757464.0000 - mse: 21757464.0000 - val_loss: 713205440.0000 -
val_mse: 713205440.0000
Epoch 19/20
1612/1612 _____ 117s 73ms/step - loss:
16243529.0000 - mse: 16243529.0000 - val_loss: 706298240.0000 -
val_mse: 706298240.0000

```

```

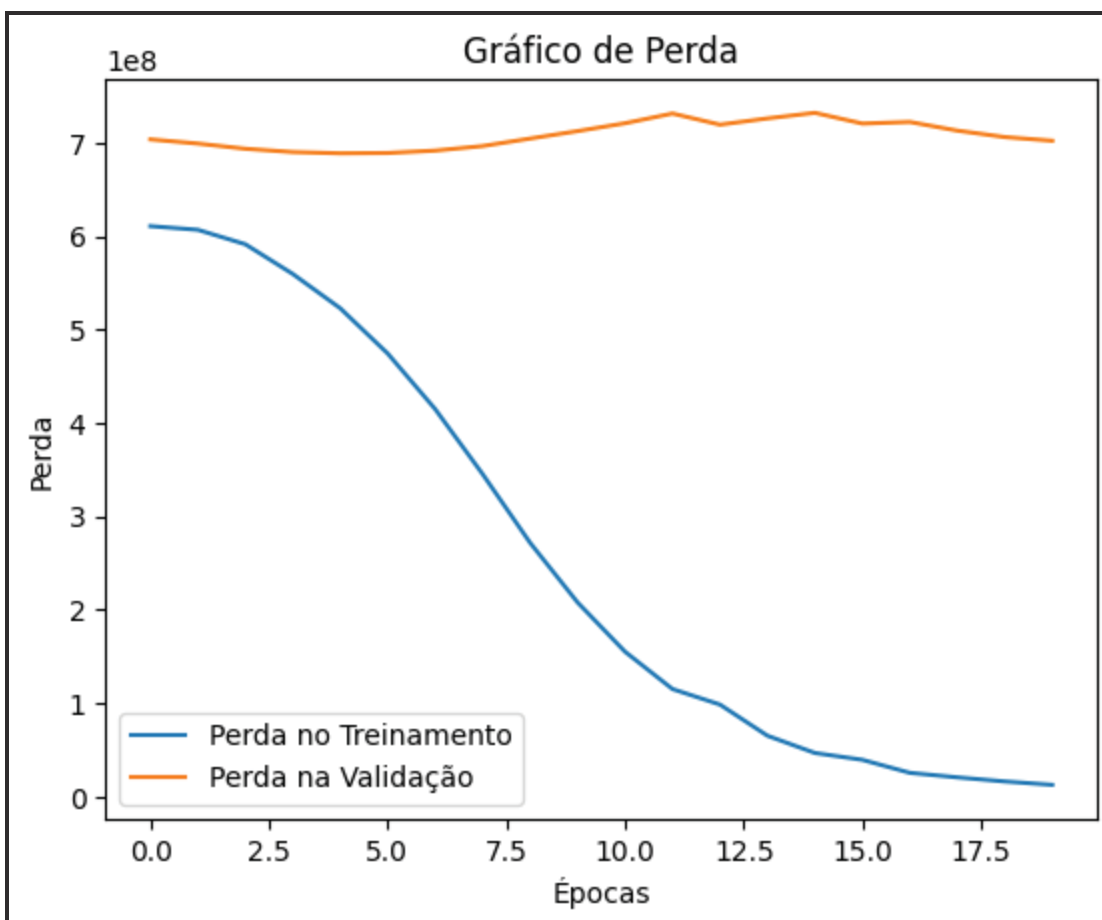
Epoch 20/20
1612/1612 _____ 140s 72ms/step - loss:
12756409.0000 - mse: 12756409.0000 - val_loss: 702384512.0000 -
val_mse: 702384512.0000

# Avaliar o modelo
mse = model.evaluate([test['user_idx'], test['book_idx']],
test['Notas'])
print(f"MSE no conjunto de teste: {mse[0]:.2f}")

806/806 _____ 1s 1ms/step - loss:
698012416.0000 - mse: 698012416.0000
MSE no conjunto de teste: 702384256.00

# Plotar o gráfico de perda
plt.plot(history.history['loss'], label='Perda no Treinamento')
plt.plot(history.history['val_loss'], label='Perda na Validação')
plt.xlabel('Épocas')
plt.ylabel('Perda')
plt.legend()
plt.title('Gráfico de Perda')
plt.show()

```



```
# Recomendação de livros para um usuário específico
user_id = user_to_index['2']
all_books = np.arange(num_books)
predictions = model.predict([np.array([user_id] * num_books),
all_books])
top_books = predictions.flatten().argsort()[-5:][::-1] # Top 5 livros

print("Recomendações de livros:")
for book_idx in top_books:
    isbn = [key for key, val in book_to_index.items() if val ==
book_idx][0]
    print(f"ISBN do livro recomendado: {isbn}")
```

4028/4028 ————— 11s 3ms/step

Recomendações de livros:

```
ISBN do livro recomendado: 3150000335
ISBN do livro recomendado: 8440616252
ISBN do livro recomendado: 765341840
ISBN do livro recomendado: 1583940677
ISBN do livro recomendado: 879517980
```

Eixo X (Épocas):

Representa o número de vezes que o modelo passa por todo o conjunto de treinamento. Cada época corresponde a um ciclo completo de aprendizado.

Eixo Y (Perda):

Mede o valor da função de perda (loss) usada para avaliar o desempenho do modelo. No caso, a perda é calculada com a métrica Mean Squared Error (MSE). Quanto menor o valor, melhor o modelo está se ajustando aos dados.

Linhas no Gráfico:

Perda no Treinamento: Representa o erro do modelo ao prever os dados do conjunto de treinamento. Geralmente, essa linha começa alta e diminui ao longo das épocas, indicando que o modelo está aprendendo.

Perda na Validação: Representa o erro do modelo ao prever os dados do conjunto de validação (dados que o modelo não viu durante o treinamento). Essa linha ajuda a verificar se o modelo está se generalizando bem.

Exemplo de Recomendação:

Ilustra como os embeddings capturam preferências, sugerindo livros com base nos padrões de avaliação do usuário.

```
Recomendações de livros:
ISBN do livro recomendado: 3150000335
ISBN do livro recomendado: 8440616252
ISBN do livro recomendado: 765341840
ISBN do livro recomendado: 1583940677
ISBN do livro recomendado: 879517980
```

4. DeepDream

```
import tensorflow as tf
import numpy as np

import matplotlib as mpl

import IPython.display as display
import PIL.Image
```

```

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

# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
    name = url.split('/')[-1]
    image_path = tf.keras.utils.get_file(name, origin=url)
    img = PIL.Image.open(image_path)
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)

# Normalização da imagem
def deprocess(img):
    img = 255*(img + 1.0)/2.0
    return tf.cast(img, tf.uint8)

# Mostra a imagem
def show(img):
    display.display(PIL.Image.fromarray(np.array(img)))

# Redução do tamanho da imagem para facilitar o trabalho da RNN
original_img = download(url, max_dim=500)
show(original_img)
display.display(display.HTML('Image cc-by: Von.grzanka'))

```



Aqui faremos o download um modelo de classificação de imagem pré-treinado. Usaremos o [InceptionV3](#) que é semelhante ao modelo originalmente usado no DeepDream. Observe que qualquer [modelo pré-treinado](#) funcionará, embora você precise ajustar os nomes das camadas abaixo, caso deseje alterá-lo.

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

A ideia no DeepDream é escolher uma camada (ou camadas) e maximizar a "perda" de forma que a imagem cada vez mais treine as camadas.

```
# Maximizando as ativações das camadas
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]

# Criação do modelo
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)
```

Cálculo da perda (loss) - A perda é a soma das ativações nas camadas escolhidas.

```
def calc_loss(img, model):
```

```

# Passe a imagem pelo modelo para recuperar as ativações.
# Converte a imagem em um batch de tamanho 1.
img_batch = tf.expand_dims(img, axis=0)
layer_activations = model(img_batch)
if len(layer_activations) == 1:
    layer_activations = [layer_activations]

losses = []
for act in layer_activations:
    loss = tf.math.reduce_mean(act)
    losses.append(loss)

return tf.reduce_sum(losses)

```

Depois de ter calculado a perda para as camadas escolhidas, tudo o que resta é calcular os gradientes em relação à imagem e adicioná-los à imagem original.

```

class DeepDream(tf.Module):
    def __init__(self, model):
        self.model = model

    @tf.function(
        input_signature=(
            tf.TensorSpec(shape=[None, None, 3], dtype=tf.float32),
            tf.TensorSpec(shape=[], dtype=tf.int32),
            tf.TensorSpec(shape=[], dtype=tf.float32),
        )
    )
    def __call__(self, img, steps, step_size):
        print("Tracing")
        loss = tf.constant(0.0)

        for n in tf.range(steps):
            with tf.GradientTape() as tape:
                # Gradientes relativos a img
                tape.watch(img)
                loss = calc_loss(img, self.model)

            # Calculo do gradiente da perda em relação aos pixels da imagem
            # de entrada.
            gradients = tape.gradient(loss, img)

            # Normalizacao dos gradintes
            gradients /= tf.math.reduce_std(gradients) + 1e-8

```




Conseguimos gerar uma imagem. Porém, há alguns problemas com esta primeira tentativa:

1. A saída é ruidosa (isso pode ser resolvido com uma perda `tf.image.total_variation`).
2. A imagem é de baixa resolução.
3. Os padrões parecem estar acontecendo na mesma granularidade.

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

OCTAVE_SCALE = 1.30

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[: -1]
float_base_shape = tf.cast(base_shape, tf.float32)

for n in range(-2, 3):
    new_shape = tf.cast(float_base_shape * (OCTAVE_SCALE ** n), tf.int32)

    img = tf.image.resize(img, new_shape).numpy()

    img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
```

```
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```



13.460156440734863

Resultados:

Imagem Onírica obtida pelo Main Loop:

Apresenta padrões refinados e detalhados, amplificando as características de interesse (como bordas e texturas).

A abordagem é direta e se concentra na amplificação global dos padrões da imagem original.

Imagem Onírica obtida ao levar o modelo até uma oitava:

Com a técnica de oitavas, a imagem é processada em diferentes escalas, permitindo que padrões sejam amplificados em diferentes níveis de detalhe. Isso resulta em uma imagem mais "onírica" e complexa, com uma sobreposição de padrões em várias escalas.

Diferenças:

Main Loop: Foca nos detalhes da resolução atual, gerando padrões mais uniformes.

Oitavas: Amplia padrões em várias escalas, criando uma composição mais rica e diversificada.

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

B – RESOLUÇÃO

Explicação do Contexto e Público-Alvo

A visualização de dados e storytelling desenvolvidos neste trabalho têm como foco a **análise da distribuição da frota de veículos no Brasil**, destacando as diferenças regionais e os impactos da mobilidade em diversas áreas, como infraestrutura, agronegócio e segurança viária.

O alvo público dessa análise inclui:

- **Tomadores de decisão** : Governos e órgãos de transporte que precisam criar políticas públicas eficazes.
- **Empresas do setor de mobilidade** : Montadoras, seguradoras e aplicativos de transporte que buscam entender padrões de consumo.
- **Pesquisadores e acadêmicos** : Estudos sobre infraestrutura, mobilidade e urbanização.
- **O público geral** : Interessados em compreender a dinâmica dos transportes no Brasil.

Escolha do Tipo de Visualização e Ferramenta Utilizada

A escolha da planilha do Google como ferramenta se justifica por sua acessibilidade, facilidade de uso e recursos que permitem a criação de tabelas e gráficos dinâmicos. Esse formato facilita a análise e interpretação dos dados de forma clara e intuitiva.

Os tipos de visualização foram escolhidos de acordo com os padrões de distribuição e comparação dos dados:

- **Gráficos de barras** para comparar a quantidade de veículos por tipo e por região.

Explicação da Escolha da Visualização de Dados

Para representar os dados sobre a distribuição da frota de veículos no Brasil, foram escolhidos gráficos comparativos e distribuições percentuais. A escolha foi guiada pela necessidade de evidenciar as diferenças regionais e destacar as principais tendências de mobilidade no país.

Por que esses tipos de gráficos foram escolhidos?

- **Gráficos de Barras**: Os gráficos de barras foram utilizados para comparar o volume total de veículos por região. Esse tipo de gráfico é ideal para

representar detalhes e permite uma fácil identificação das diferenças entre regiões, como o domínio do Sudeste na frota nacional ou a alta presença de motocicletas no Norte e Nordeste.

- Gráficos de Colunas: As colunas foram utilizadas para destacar ocorrências dentro de cada região, como a participação de automóveis, motocicletas, caminhões e caminhonetes. Esse tipo de visualização facilita a compreensão do perfil da frota em diferentes contextos.
- Tabelas Resumidas: Além dos gráficos, as tabelas foram usadas para sintetizar informações e permitir comparações numéricas diretas. Isso ajuda na análise detalhada dos dados para insights mais específicos.

Por que não foram usados mapas de calor ou gráficos de pizza?

- Mapas de Calor: Apesar de serem eficazes para mostrar concentrações geográficas, mapas de calor apresentam um detalhamento geoespacial mais avançado e podem não ser a melhor escolha para representar distribuições absolutas de veículos sem um componente espacial detalhado.
- Gráficos de Pizza: Embora úteis para mostrar proporções, gráficos de pizza podem dificultar a comparação precisa entre categorias. Como o foco é evidenciado em variações **regionais e tendências**, gráficos de barras e colunas são oferecidos mais claros na comparação entre os dados.

Conclusão

A escolha da visualização foi baseada na clareza e facilidade de interpretação dos dados. Os gráficos utilizados permitem uma leitura objetiva, destacando as principais diferenças regionais na frota brasileira e facilitando a geração de insights estratégicos para mobilidade e infraestrutura.

A seguir, apresento a descrição da narrativa (storytelling) desta visualização de dados, conforme os conceitos discutidos em aula, seguindo padrões de cores e gráficos para garantir clareza e impacto na comunicação das informações.

Distribuição da Frota de Veículos no Brasil - Um Panorama Regional

Análise da Distribuição de Veículos por Região

Visão Geral: O Brasil possui uma frota total de 123,9 milhões de veículos, sendo que 51% são automóveis e 22,8% são motocicletas. Esses dados foram obtidos em dezembro de 2024, a partir de levantamentos realizados pelo Governo Federal. A análise revela variações significativas entre as regiões do país, influenciadas por fatores como urbanização, necessidades de transporte e políticas públicas.

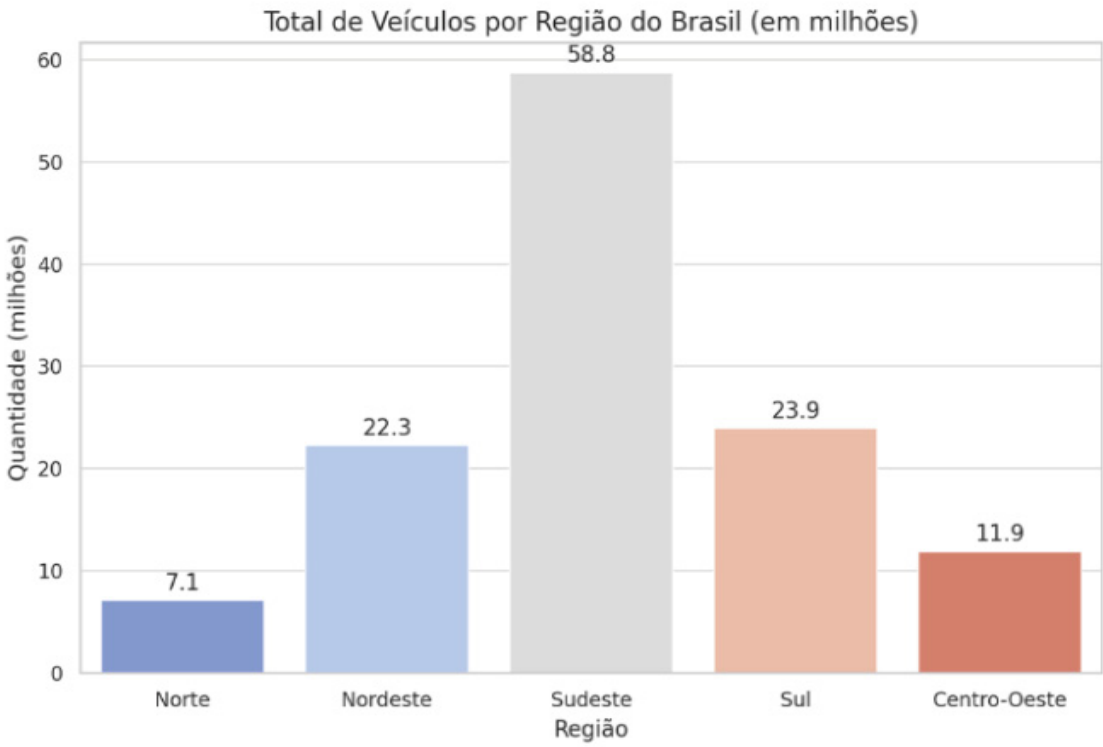
- **Sudeste:** Com 58,7 milhões de veículos, concentra quase metade da frota nacional. São Paulo lidera com 34,3 milhões de veículos, sendo 20,4 milhões de automóveis.
- **Nordeste:** Apresenta um alto número de motocicletas, especialmente em estados como Ceará e Maranhão, onde são essenciais para deslocamentos urbanos e rurais.
- **Sul:** Possui uma frota equilibrada, destacando-se pelo uso de caminhonetes e utilitários, refletindo a força do agronegócio na região.
- **Centro-Oeste:** Tem uma alta proporção de caminhonetes e caminhões, essenciais para o escoamento da produção agrícola
- **Norte:** Apesar da menor frota total, apresenta crescimento significativo no uso de motocicletas para acesso a áreas remotas.

Tendências da Frota Brasileira: O Que os Dados Revelam e Como Agir

Insights e Ações Possíveis

Problema	Impacto
Mobilidade urbana e infraestrutura.	O alto número de automóveis no Sudeste exige investimentos contínuos em transporte público e infraestrutura viária para evitar congestionamento.
Crescimento das motocicletas no Nordeste e Norte.	Indica a necessidade de políticas de segurança viária e programas de financiamento para a modernização da frota.

Demanda por veículos utilitários e caminhões no Sul e Centro-Oeste.	Reforça a importância do setor agropecuário e a necessidade de melhorias nas rodovias para transporte de cargas.
Sustentabilidade e renovação da frota.	Com base nesses dados, políticas de incentivo à eletrificação e redução de emissões podem ser direcionadas para regiões com maior frota de veículos pesados.

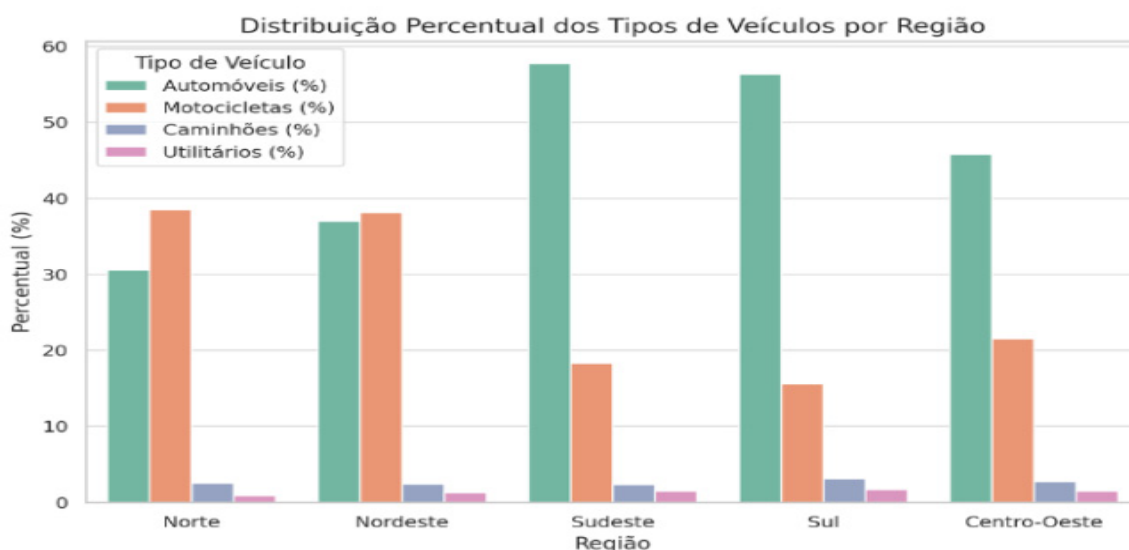


Esse gráfico destaca a concentração da frota no Sudeste, seguida pelo Nordeste e Sul.

Distribuição dos veículos por Região

O gráfico abaixo apresenta a distribuição percentual dos tipos de veículos por região do Brasil. Essa visualização permite comparar como diferentes categorias de

veículos estão distribuídas pelo território nacional, destacando tendências regionais e padrões de mobilidade.



- **Sudeste: O Domínio dos Automóveis**

- O Sudeste possui a maior concentração de automóveis, representando quase 60% da frota regional. Esse cenário é influenciado pela forte urbanização e pela infraestrutura rodoviária desenvolvida, onde o carro particular ainda é o meio de transporte predominante. No entanto, essa alta dependência do automóvel gera desafios como congestionamentos e impactos ambientais, reforçando a necessidade de alternativas como transporte público e mobilidade elétrica.

- **Norte e Nordeste: O Papel das Motocicletas**

- O gráfico revela que o Norte e o Nordeste possuem as maiores proporções de motocicletas no país, superando 38% da frota regional. Esse dado reflete a acessibilidade das motos, que são uma opção econômica para deslocamentos curtos, tanto em cidades quanto em áreas rurais. No entanto, essa tendência também alerta para a importância de políticas de segurança viária, visto que motociclistas estão mais vulneráveis a acidentes.

- **Sul e Centro-Oeste: A Força dos Caminhões e Utilitários**

- As regiões Sul e Centro-Oeste apresentam uma participação maior de caminhonetes e caminhões, essenciais para o transporte de cargas e o escoamento da produção agrícola. No Centro-Oeste, por exemplo,

esse tipo de veículo representa mais de 20% da frota em algumas localidades, evidenciando a forte relação entre o agronegócio e a mobilidade regional.

- **Ônibus e Utilitários: Impacto do Transporte Coletivo**

- Embora com menor participação percentual, os ônibus e utilitários são fundamentais em todas as regiões, especialmente onde o transporte público tem um papel relevante. A presença dessas categorias no Sudeste e Centro-Oeste sugere que, apesar do domínio dos automóveis, ainda há demanda por transporte coletivo em algumas áreas metropolitanas.

A Supremacia dos Automóveis

Mais da metade da frota nacional é composta por automóveis (51,06%). Isso significa que, a cada dois veículos que encontramos, um é um carro de passeio. Essa predominância reflete a cultura do transporte individual no Brasil, onde a infraestrutura urbana e rodoviária incentiva o uso do carro particular.

O Papel Fundamental das Duas Rodas

No entanto, se olharmos com atenção, veremos outro fenômeno importante: as motocicletas representam 22,82% da frota nacional. Nos últimos anos, o crescimento dos serviços de entrega, como aplicativos de delivery e transporte rápido, impulsionou o uso desse veículo, tornando-o indispensável principalmente em centros urbanos.

O Transporte de Cargas e Passageiros

Caminhões (2,55%) e caminhonetes (8,10%) mostram a importância do transporte rodoviário na economia do Brasil, país onde a maior parte das mercadorias ainda é transportada por estradas. Ônibus (0,59%) e micro-ônibus (0,37%) representam uma pequena fatia da frota total, refletindo desafios no transporte público, que ainda precisa de melhorias para atrair mais usuários.

Conclusão

A distribuição dos veículos no Brasil conta uma história sobre o estilo de vida da população, os desafios do transporte público e a dependência do modal rodoviário. A predominância dos automóveis e motocicletas nos leva a refletir sobre

a necessidade de investimentos em infraestrutura, sustentabilidade e alternativas de mobilidade para um futuro mais equilibrado.

APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java.

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida).

Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels.

O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

- (1) considere o cromossomo formado pelas cidades, onde a cidade de início (escolhida aleatoriamente) deverá estar na posição 0 e 100 e a ordem das

idades visitadas nas posições de 1 a 99 deverão ser definidas pelo algoritmo genético.

- (2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
- (3) Utilize no mínimo uma população com 100 indivíduos;
- (4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;
- (5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);
- (6) Preserve sempre a melhor solução de uma geração para outra.

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

Introdução

Este documento apresenta um estudo detalhado sobre dois tópicos fundamentais de Inteligência Artificial: a resolução do Problema do Caixeiro Viajante utilizando Algoritmos Genéticos e a Visualização de Representação Vetorial de Textos com PCA. O primeiro tópico aborda a implementação de um algoritmo genético para encontrar rotas eficientes entre diversas cidades, minimizando a

distância percorrida. Para isso, são definidos parâmetros como número de cidades, tamanho da população, gerações, taxa de mutação e crossover. As cidades são geradas aleatoriamente dentro de um espaço definido e a distância entre elas é calculada usando a métrica euclidiana. A população inicial de rotas é criada aleatoriamente, e a evolução ocorre por meio de seleção por torneio, crossover ordenado e mutação. A cada geração, a melhor solução é selecionada e sua rota é visualizada graficamente para acompanhamento da otimização. Após várias iterações, o algoritmo apresenta a melhor rota encontrada, evidenciando a eficiência da abordagem evolutiva.

O segundo tópico do documento explora a visualização de representações vetoriais de textos por meio do PCA. Inicialmente, um conjunto de seis textos curtos é definido, contendo algumas similaridades entre si. A vetorização desses textos é realizada com o método TF-IDF, que transforma cada texto em um vetor numérico. Esses vetores são então reduzidos a duas dimensões utilizando a técnica de Análise de Componentes Principais (PCA), permitindo uma representação visual da proximidade semântica entre os textos. A projeção resultante é plotada em um gráfico bidimensional, com cada ponto representando um dos textos. O objetivo desse procedimento é demonstrar como textos semelhantes tendem a se agrupar em um espaço vetorial, facilitando a interpretação das relações semânticas entre eles.

Ambos os tópicos apresentados evidenciam a aplicabilidade da Inteligência Artificial para resolver problemas complexos, seja na otimização de rotas ou na análise de dados textuais. As técnicas utilizadas neste estudo são amplamente empregadas em diversas áreas, desde logística e transporte até a análise de linguagem natural, destacando a importância da IA na resolução de desafios do mundo real.

Questão 1

Importação de Bibliotecas

Importa numpy, matplotlib.pyplot, random e itertools.permutations, essenciais para manipulação de dados, cálculos e visualização.

```
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import permutations
```

Definição de Parâmetros do Algoritmo

Define variáveis globais como o número de cidades, tamanho da população, número de gerações, taxa de mutação e taxa de crossover.

```
NUM_CITIES = 100
POP_SIZE = 100
GENERATIONS = 1000
MUTATION_RATE = 0.01
CROSSOVER_RATE = 0.9
```

Geração de Coordenadas das Cidades

Função generate_cities() cria cidades aleatórias dentro de um espaço definido.

```
def generate_cities(num_cities, size=100):
    return np.random.rand(num_cities, 2) * size
```

Cálculo da Distância entre Duas Cidades

Implementa a função distance() para calcular a distância euclidiana entre duas cidades.

```
def distance(city1, city2):
    return np.linalg.norm(city1 - city2)
```

Cálculo da Distância Total de uma Rota

A função total_distance() calcula o comprimento total de uma rota específica.

```
def total_distance(route, cities):
    dist = sum(distance(cities[route[i]], cities[route[i+1]]) for i in range(len(route) - 1))
```

```
return dist + distance(cities[route[-1]], cities[route[0]]) # Retorno à cidade inicial
```

Geração da População Inicial

A função `initial_population()` cria uma população inicial de rotas aleatórias.

```
def initial_population(pop_size, num_cities):
    return [np.random.permutation(num_cities) for _ in range(pop_size)]
```

Seleção por Torneio

A função `tournament_selection()` escolhe uma rota com base na menor distância total dentro de um subconjunto aleatório da população.

```
def tournament_selection(population, cities, k=5):
    selected = random.sample(population, k)
    return min(selected, key=lambda route: total_distance(route, cities))
```

Crossover Ordenado (OX)

Implementa `ordered_crossover()`, que combina duas rotas de maneira ordenada para criar um indivíduo.

```
def ordered_crossover(parent1, parent2):
    size = len(parent1)
    start, end = sorted(random.sample(range(size), 2))
    child = [-1] * size
    child[start:end+1] = parent1[start:end+1]

    ptr = 0
    for city in parent2:
        if city not in child:
            while child[ptr] != -1:
                ptr += 1
            child[ptr] = city
    return np.array(child)
```

Mutação de uma Rota

A função `mutate()` troca aleatoriamente duas cidades dentro da rota com uma certa probabilidade.

```
def mutate(route, mutation_rate):
    if random.random() < mutation_rate:
        i, j = random.sample(range(len(route)), 2)
        route[i], route[j] = route[j], route[i]
    return route
```

Criação da próxima geração

A função `next_generation()` aplica elitismo, seleção, crossover e mutação para gerar a nova população.

```
def next_generation(population, cities):
    new_population = [min(population, key=lambda r: total_distance(r, cities))] #
    Elitismo
    while len(new_population) < POP_SIZE:
        parent1 = tournament_selection(population, cities)
        parent2 = tournament_selection(population, cities)
        if random.random() < CROSSOVER_RATE:
            child = ordered_crossover(parent1, parent2)
        else:
            child = parent1.copy()
        new_population.append(mutate(child, MUTATION_RATE))
    return new_population
```

Visualização da Rota

Plota graficamente a melhor rota encontrada até o momento, permitindo visualizar a eficiência do algoritmo.

```
def plot_route(route, cities, generation):
    plt.figure(figsize=(8, 8))
    ordered_cities = np.array([cities[i] for i in route] + [cities[route[0]]])
    plt.plot(ordered_cities[:, 0], ordered_cities[:, 1], 'bo-')
```

```
plt.scatter(cities[:, 0], cities[:, 1], color='red')
plt.title(f'Best Route - Generation {generation}')
plt.show()
```

Configuração Inicial do Algoritmo

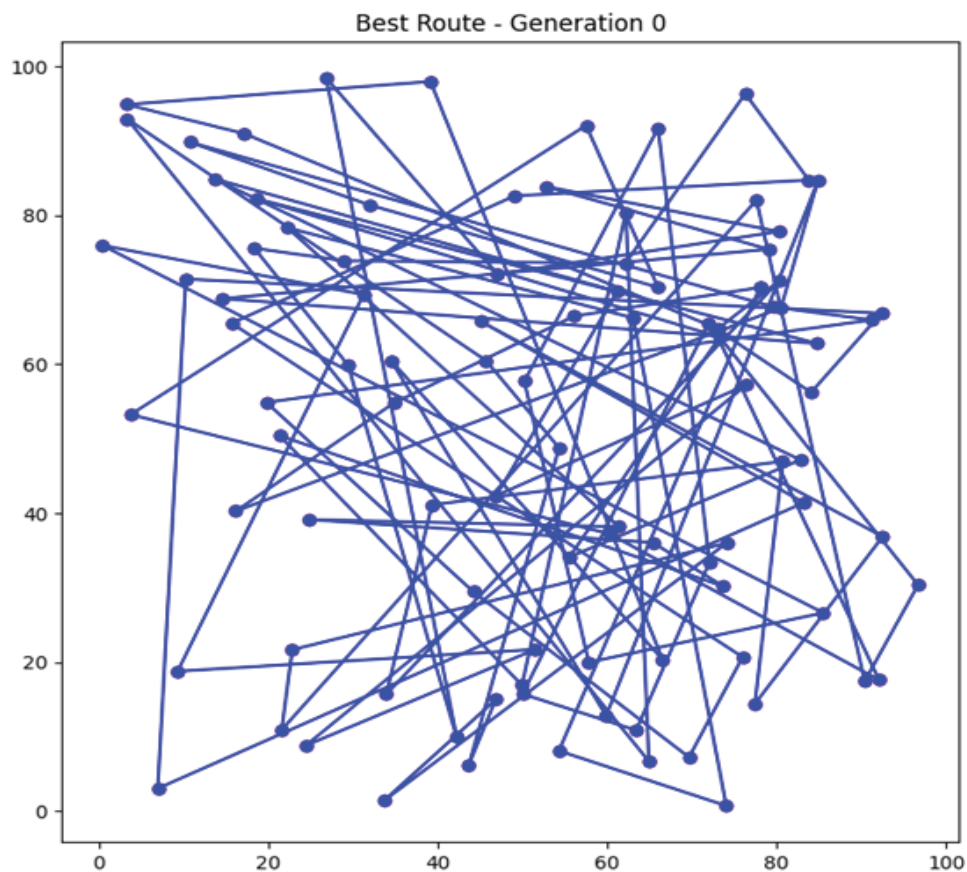
Define os parâmetros globais e gera as cidades aleatórias para a simulação do problema do Caixeiro Viajante.

```
data_cities = generate_cities(NUM_CITIES)
population = initial_population(POP_SIZE, NUM_CITIES)
best_route = min(population, key=lambda r: total_distance(r, data_cities))
```

Solução Inicial Aleatória

Gera uma solução inicial aleatória e calcula seu custo como ponto de partida para a otimização.

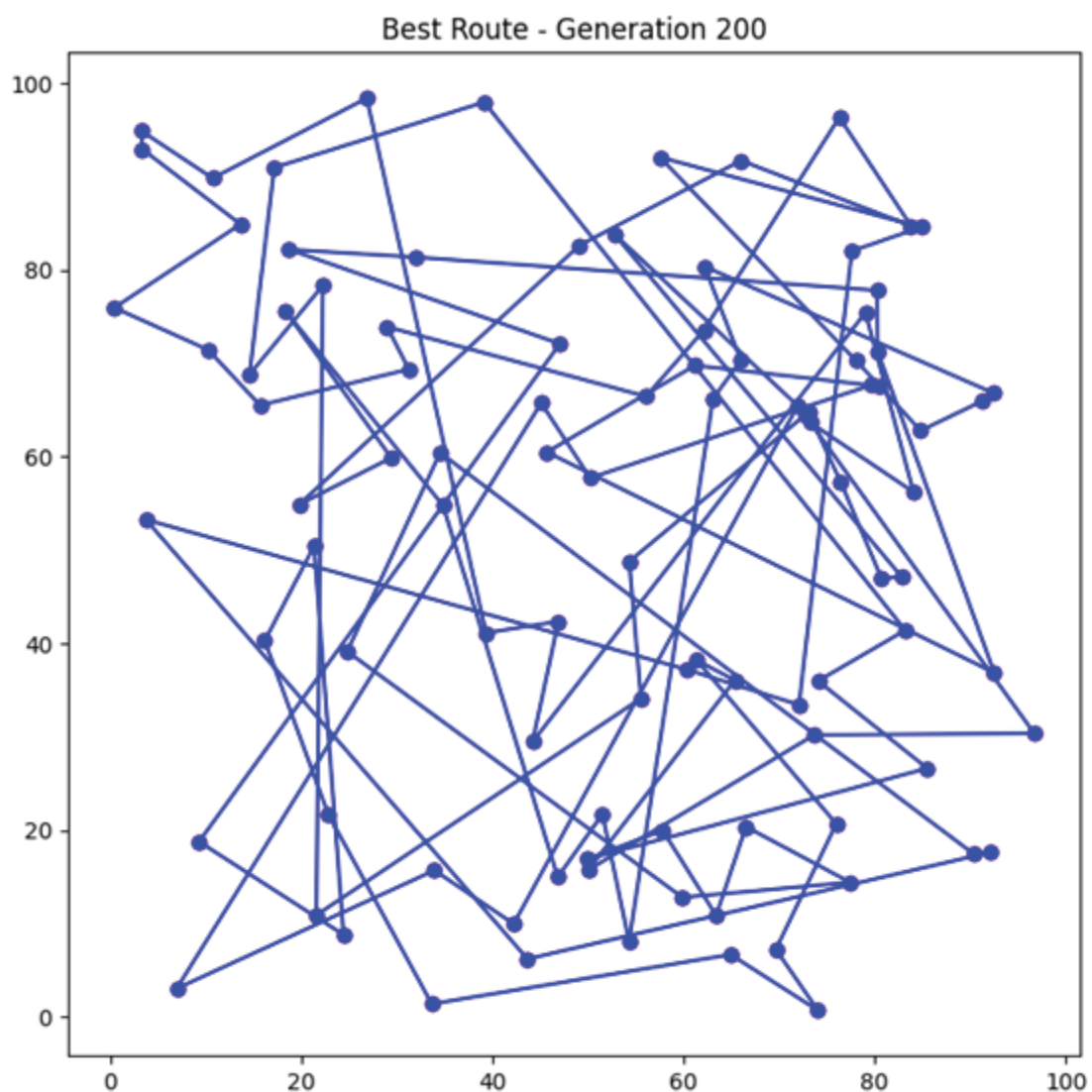
```
plot_route(best_route, data_cities, 0)
```

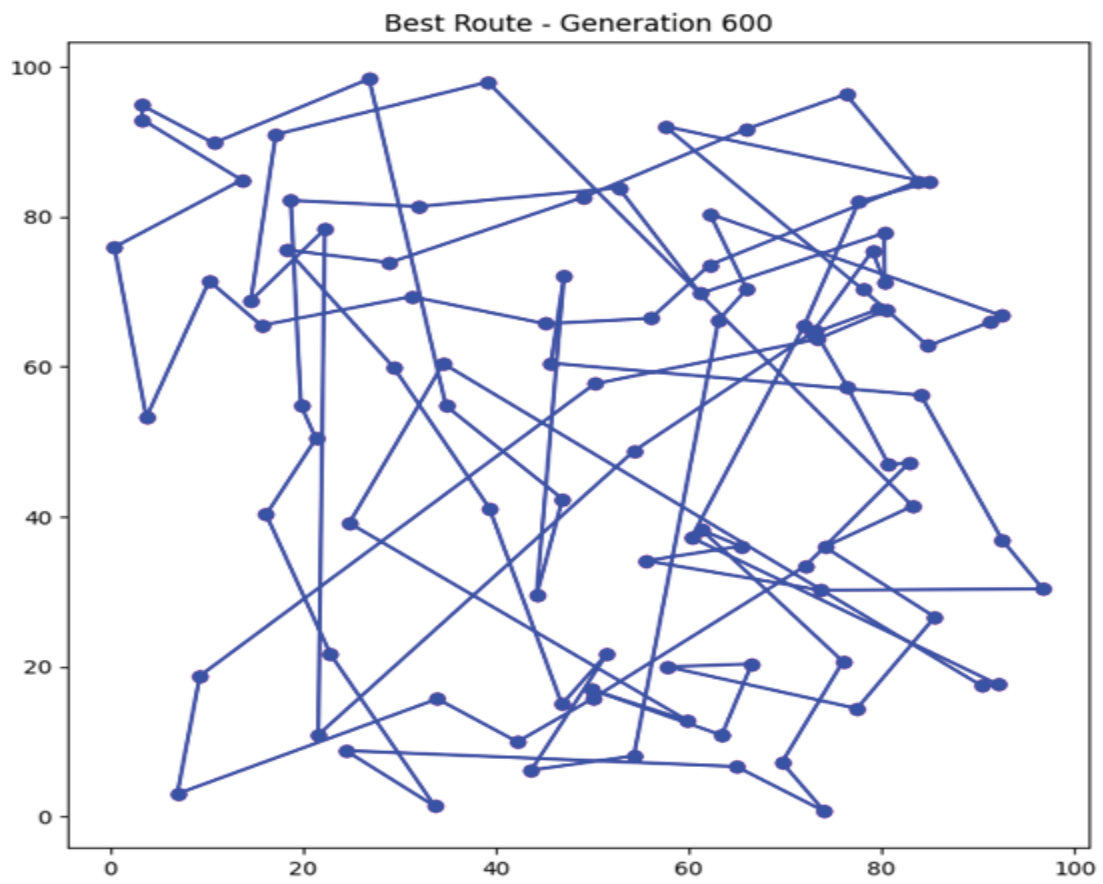
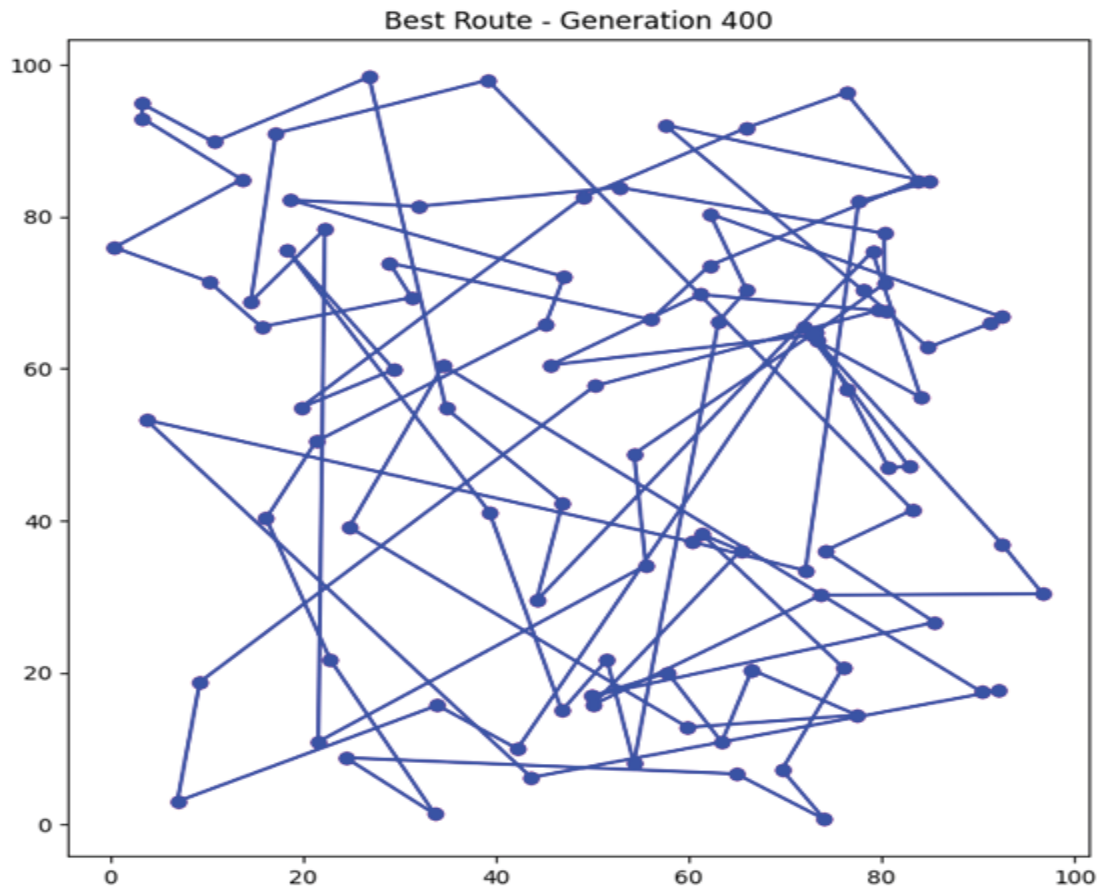


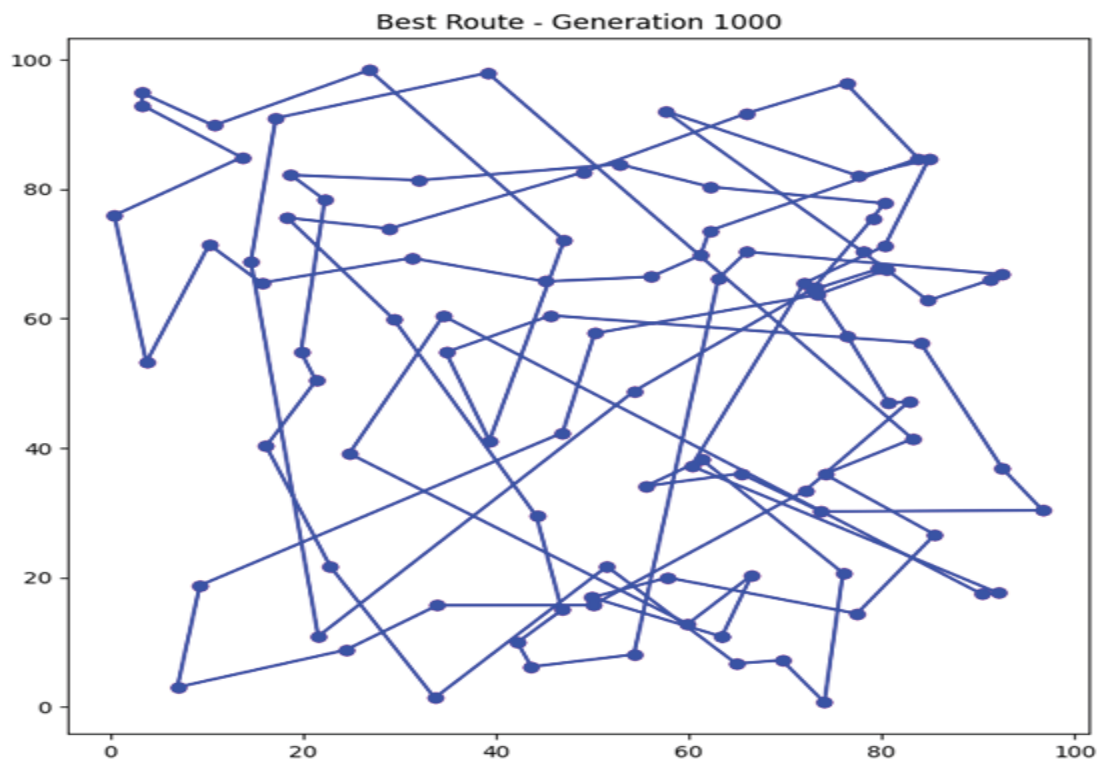
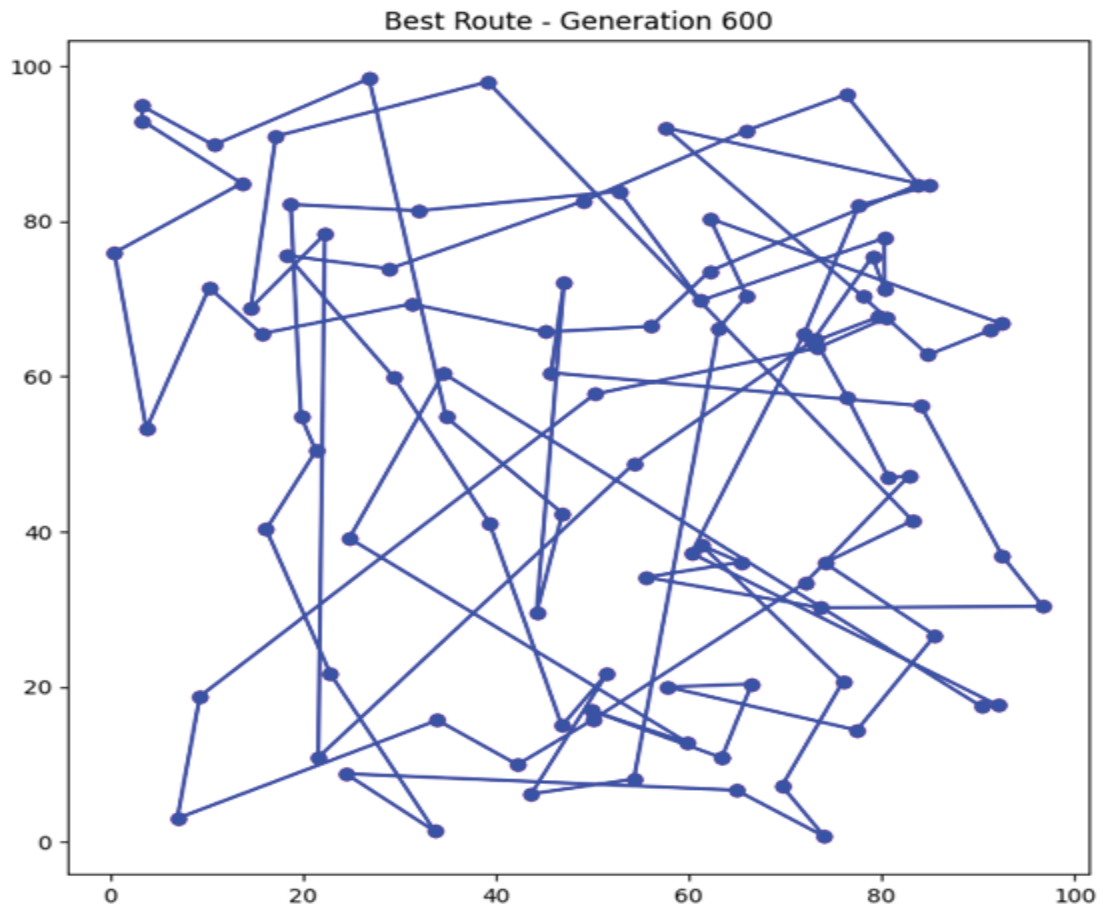
Progresso da Otimização

Executa várias gerações do algoritmo genético, aprimorando progressivamente as soluções.

```
for gen in range(1, GENERATIONS + 1):
    population = next_generation(population, data_cities)
    best_route = min(population, key=lambda r: total_distance(r, data_cities))
    if gen % 200 == 0:
        plot_route(best_route, data_cities, gen)
```



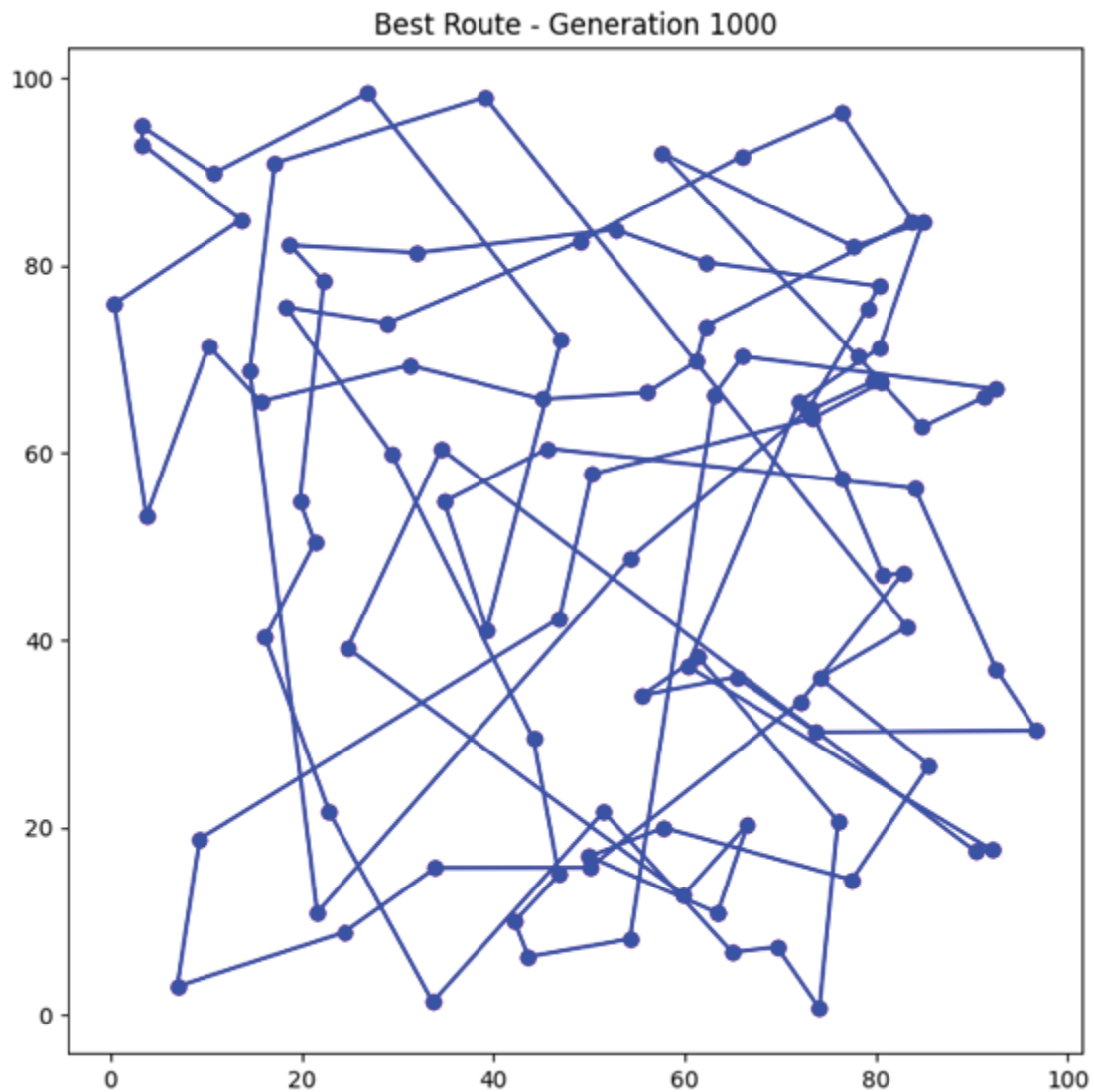




Resultado Final do Algoritmo

Exibe a melhor rota encontrada após todas as iterações e plota o resultado final.

```
plot_route(best_route, data_cities, GENERATIONS)
```



Questão 2

Visualização de Representação Vetorial de Textos com PCA

Neste notebook, iremos:

1. Definir um conjunto de 6 textos pequenos (com palavras/sentenças similares);
2. Vetorizá-los usando TfidfVectorizer (vetor por texto);

3. Reduzir a dimensionalidade dos vetores para 2 dimensões com PCA;
4. Plotar a projeção dos vetores com anotações.

Esta abordagem permite visualizar as similaridades entre os textos.

Importar as bibliotecas necessárias

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer
```

Para exibir os gráficos online no notebook

```
%matplotlib inline
```

Definir uma lista de 6 textos (com pelo menos duas similares em conteúdo)

```
texts = [
    "O sol nasce no oriente e se põe no ocidente.",
    "A lua brilha intensamente durante a noite.",
    "Meu Pokémon favorito é o Slowpoke.",
    "Amo viajar pelo mundo.",
    "Eu jogo Nintendo Switch.",
    "Estudo Inteligência Artificial Aplicada."
]
```

Exibe os textos para conferência

```
for i, text in enumerate(texts, 1):
    print(f"Texto {i}: {text}")
```

Texto 1: O sol nasce no oriente e se põe no ocidente.

Texto 2: A lua brilha intensamente durante a noite.

Texto 3: Meu Pokémon favorito é o Slowpoke.

Texto 4: Amo viajar pelo mundo.

Texto 5: Eu jogo Nintendo Switch.

Texto 6: Estudo Inteligência Artificial Aplicada.

Vetorização dos textos utilizando TfidfVectorizer

```
vectorizer = TfidfVectorizer()
```

```
X = vectorizer.fit_transform(texts) # X terá dimensão (6, tamanho_vocab)
```

Converter a matriz esparsa para densidade para visualização e manipulação

```
X_dense = X.toarray()
```

Exibir a representação vetorial dos textos

```
print("Representação vetorial (TF-IDF) de cada texto:")
```

```
print(X_dense)
```

Representação vetorial (TF-IDF) de cada texto:

```
[[0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.
  0.  0.  0.31622777 0.  0.63245553 0.
  0.31622777 0.31622777 0.  0.  0.31622777 0.31622777
  0.  0.31622777 0.  0.  ]
 [0.  0.  0.  0.4472136 0.4472136 0.
  0.  0.  0.  0.4472136 0.  0.4472136
  0.  0.  0.  0.  0.  0.4472136
  0.  0.  0.  0.  0.  0.
  0.  0.  0.  0.  ]
 [0.  0.  0.  0.  0.  0.
  0.  0.5 0.  0.  0.  0.
  0.5  0.  0.  0.  0.  0.
  0.  0.  0.  0.5 0.  0.
  0.5  0.  0.  0.  ]
 [0.5  0.  0.  0.  0.  0.
  0.  0.  0.  0.  0.  0.
  0.  0.5 0.  0.  0.  0.
  0.  0.  0.5 0.  0.  0.
  0.  0.  0.  0.5 ]
 [0.  0.  0.  0.  0.  0.
  0.5  0.  0.  0.  0.5  0.
  0.  0.  0.  0.5 0.  0.]
```

```

0.  0.  0.  0.  0.  0.
0.  0.  0.5  0.  ]
[0.  0.5  0.5  0.  0.  0.5
0.  0.  0.5  0.  0.  0.
0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  0.  0.
0.  0.  0.  0.  ]]

```

Reduzir a dimensionalidade dos vetores para 2 dimensões usando PCA

```

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_dense)

```

Exibir os vetores projetados no espaço 2D

```

print("Vetores dos textos após PCA:")
print(X_pca)

```

Vetores dos textos após PCA:

```

[[-4.64562295e-17  2.52190960e-16]
 [ 2.90372461e-01 -5.22507760e-01]
 [-3.63193435e-01 -3.80117144e-02]
 [-2.20788521e-01  7.43318993e-01]
 [-4.41380395e-01 -3.70961732e-01]
 [ 7.34989890e-01  1.88162213e-01]]

```

Plotar a projeção PCA

```

plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c='blue')

```

Adicionar anotações para cada ponto

```

for i, (x, y) in enumerate(X_pca):
    plt.annotate(f"Texto {i+1}", (x, y), textcoords="offset points", xytext=(5,5),
ha='center')

```

```
plt.title('Projeção dos Vetores de Texto usando PCA')  
plt.xlabel('Componente Principal 1')  
plt.ylabel('Componente Principal 2')  
plt.grid(True)  
plt.show()
```

