

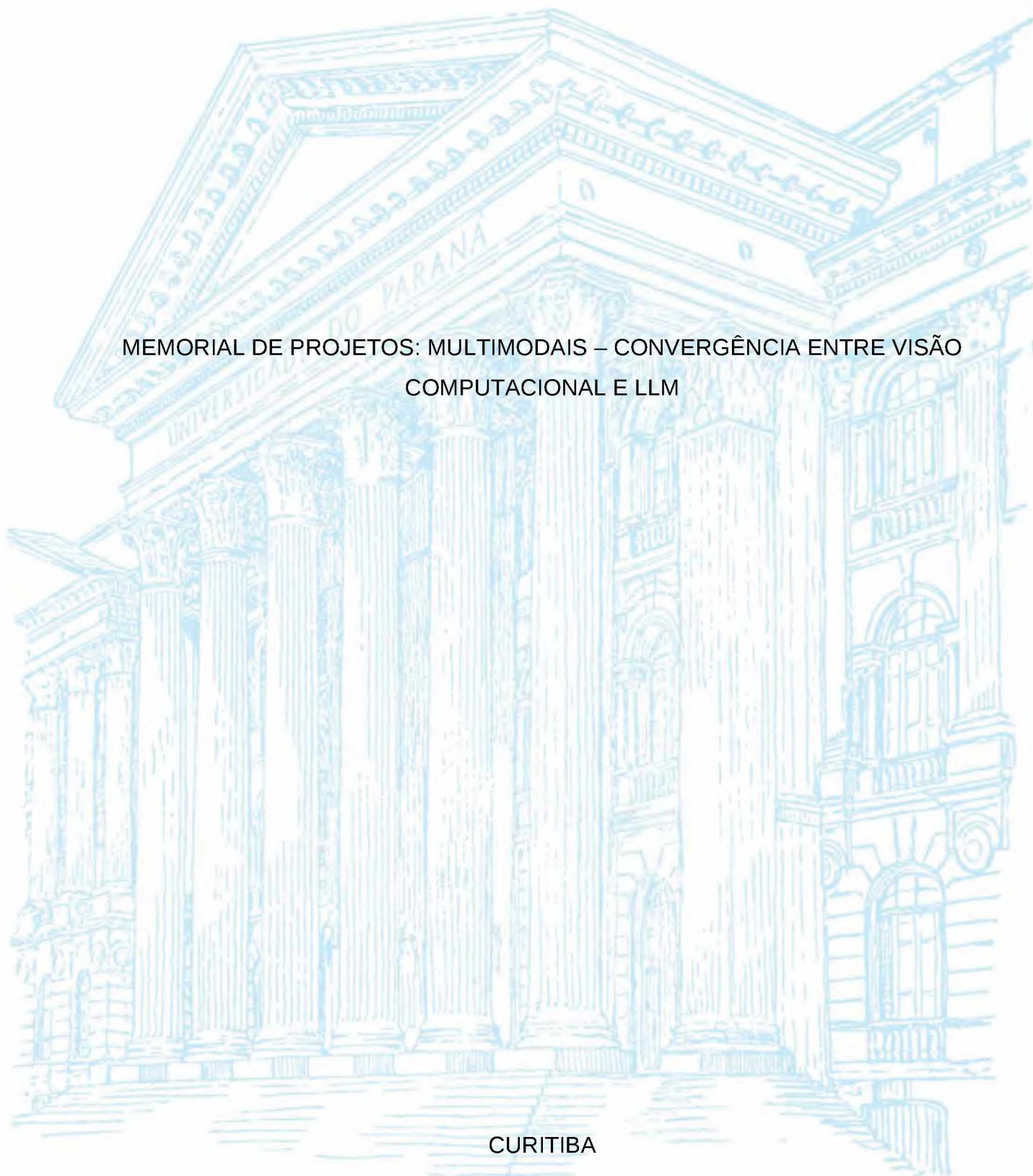
UNIVERSIDADE FEDERAL DO PARANÁ

MAGNUS HERMAN SOUZA SILVA

MEMORIAL DE PROJETOS: MULTIMODAIS – CONVERGÊNCIA ENTRE VISÃO
COMPUTACIONAL E LLM

CURITIBA

2025



MAGNUS HERMAN SOUZA SILVA

MEMORIAL DE PROJETOS: MULTIMODAIS – CONVERGÊNCIA ENTRE VISÃO
COMPUTACIONAL E LLM

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

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2025



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


TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **MAGNUS HERMAN SOUZA SILVA**, intitulada: **MEMORIAL DE PROJETOS: MULTIMODAIS CONVERGÊNCIA ENTRE VISÃO COMPUTACIONAL E LLM**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua **aprovação** no rito de defesa.

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

Curitiba, 04 de Outubro de 2025.


JAIWE WOJCIECHOWSKI
Presidente da Banca Examinadora


RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este trabalho percorre um panorama histórico-conceitual da evolução da Inteligência Artificial (IA), concentrando-se na convergência entre duas áreas inicialmente separadas: os Modelos de Linguagem de Grande Porte (LLMs – Large Language Model) e a Visão Computacional. A Inteligência Artificial (IA) consolidou-se como disciplina nas décadas de 1940-1950, passando por ciclos de avanços ("primaveras") e estagnação ("invernos"). A virada para a IA (Inteligência Artificial) moderna deu-se com as redes neurais, que permitem aprender padrões diretamente dos dados. Inicialmente, redes recorrentes (RNNs - Recurrent Neural Networks, LSTM - Long Short Term Memory) processavam sequências como texto, mas com limitações. A revolução veio em 2017 com a arquitetura Transformer e seu mecanismo de auto-atenção, base dos atuais Modelos de Linguagem de Grande Porte (LLMs – Large Language Model), como GPT (Generative Pre-trained Transformer) e BERT (Bidirectional Encoder Representations from Transformers), que compreendem e geram linguagem com profundidade sem precedentes. Paralelamente, a Visão Computacional evoluiu com Redes Neurais Convolucionais (CNNs - Convolutional Neural Network), capacitando máquinas a "enxergarem" e interpretar imagens. A fronteira atual reside na integração multimodal, onde modelos como o CLIP (Contrastive Language–Image Pre-Training) alinham representações de texto e imagem em um espaço vetorial comum. Isso permite que sistemas compreendam cenas visuais por meio da linguagem, classificando imagens sem treinamento prévio (zero-shot) e descrevendo-as detalhadamente. Essa convergência entre LLMs (Large Language Model) e Visão Computacional está criando assistentes inteligentes, robôs autônomos e ferramentas inovadoras, redefinindo a interação homem-máquina ao unir visão e linguagem em uma percepção artificial unificada e contextualizada.

Palavras-chave: inteligência artificial; LLM; transformador; visão computacional; multimodal; redes neurais convolucionais; mecanismo de auto-atenção.

ABSTRACT

This work traces a historical-conceptual panorama of the evolution of Artificial Intelligence (AI), focusing on the convergence between two initially separate areas: Large Language Models (LLMs - Large Language Models) and Computer Vision. Artificial Intelligence (AI) was established as a discipline in the 1940s-1950s, going through cycles of advances ("springs") and stagnation ("winters"). The turning point for modern AI (Artificial Intelligence) came with neural networks, which allow learning patterns directly from data. Initially, recurrent networks (RNNs - Recurrent Neural Networks, LSTM - Long Short-Term Memory) processed sequences like text, but with limitations. The revolution came in 2017 with the Transformer architecture and its self-attention mechanism, the basis of current Large Language Models (LLMs - Large Language Models), such as GPT (Generative Pre-trained Transformer) and BERT (Bidirectional Encoder Representations from Transformers), which understand and generate language with unprecedented depth. In parallel, Computer Vision evolved with Convolutional Neural Networks (CNNs - Convolutional Neural Networks), enabling machines to "see" and interpret images. The current frontier lies in multimodal integration, where models like CLIP (Contrastive Language-Image Pretraining) align text and image representations in a common vector space. This allows systems to understand visual scenes through language, classifying images without prior training (zero-shot) and describing them in detail. This convergence between LLMs (Large Language Models) and Computer Vision is creating intelligent assistants, autonomous robots, and innovative tools, redefining human-machine interaction by uniting vision and language into a unified and contextualized artificial perception.

Keywords: artificial intelligence; LLM; transformers; computer vision; multimodal; convolutional neural networks; self-attention mechanism.

SUMÁRIO

1 PARECER TÉCNICO.....	7
REFERÊNCIAS.....	10
APÊNDICE 1 - INTRODUÇÃO À INTELIGÊNCIA.....	11
APÊNDICE 2 - LINGUAGEM DE PROGRAMAÇÃO APLICADA.....	17
APÊNDICE 3 - LINGUAGEM R.....	28
APÊNDICE 4 - ESTATÍSTICA APLICADA I.....	36
APÊNDICE 5 - ESTATÍSTICA APLICADA II.....	43
APÊNDICE 6 - ARQUITETURA DE DADOS.....	49
APÊNDICE 7 - APRENDIZADO DE MÁQUINA.....	58
APÊNDICE 8 - DEEP LEARNING.....	67
APÊNDICE 9 - BIG DATA.....	91
APÊNDICE 10 - VISÃO COMPUTACIONAL.....	96
APÊNDICE 11 - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	118
APÊNDICE 12 - GESTÃO DE PROJETOS DE IA.....	124
APÊNDICE 13 - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	129
APÊNDICE 14 - VISUALIZAÇÃO DE DADOS E STORYTELLING.....	156
APÊNDICE 15 - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	160

1 PARECER TÉCNICO

A Inteligência Artificial (IA), tem suas raízes nas décadas de 1940 ~ 1950, com a proposta de Alan Turing sobre máquinas que poderiam "pensar" (Russell; Norvig, 2021). Ao longo de décadas, a área de IA passou por "invernos" e primaveras, ou seja, períodos de tempo onde o custo em conjunto com a ausência de capacidade computacional necessária, forçaram pesquisadores a não evoluírem chegando a um limite, não teriam condições técnicas de criarem inovações para propor novos paradigmas, ou impulsionada por avanços teóricos e, sobretudo, pelo aumento exponencial do poder computacional e da disponibilidade de grandes massas de dados (*big data*).

Inicialmente, os sistemas eram baseados em regras simbólicas, com conhecimento explicitamente programado. No entanto, a virada de chave para a IA moderna deu-se com o aprofundamento dos estudos sobre redes neurais artificiais – a visão conexionista. A capacidade de aprender representações diretamente dos dados, por meio de algoritmos, abriu caminho para soluções mais robustas e generalizáveis. Neste contexto, duas frentes se desenvolveram de forma paralela: a compreensão da linguagem natural e a interpretação de conteúdo visual.

Antes da era dos *Transformers*, a modelagem de sequências linguísticas era dominada por arquiteturas recorrentes:

- Redes Neurais Recorrentes (RNNs): Projetadas para processar sequências (como palavras em uma frase), as RNNs mantêm um "estado oculto" que age como uma memória sobre os elementos anteriores. No entanto, sofrem com o problema do desaparecimento do gradiente (*vanishing gradient*), o que limita severamente sua capacidade de aprender dependências de longo prazo em textos extensos. (Charniak, 2019)

- LSTM (*Long Short-Term Memory*) e GRU (*Gated Recurrent Unit*): Como evoluções das RNNs, as unidades LSTM e GRU introduziram mecanismos de "portas" (*gates*) que permitem ao modelo controlar com precisão qual informação reter, atualizar ou esquecer na memória de longo prazo. Isso mitigou significativamente o problema do gradiente e tornou-as a arquitetura padrão para tarefas de tradução automática e geração de texto por anos posteriores. (Charniak, 2019)

O momento importante na evolução da técnica de compreensão e geração de textos ocorreu em 2017 com a publicação do artigo "*Attention Is All You Need*" (Alammar; Grootendorst, 2024). A arquitetura *Transformer* abandonou completamente o conceito sequencial, introduzindo o Mecanismo de Auto-Atenção (*Self-Attention*) (Alammar; Grootendorst, 2024). Este mecanismo permite que o modelo pese a importância de todas as palavras em uma sequência, independentemente da sua distância.

É sobre esta arquitetura que as Large Language Models (LLMs) são construídas. Modelos como GPT (*Generative Pre-trained Transformer*) e BERT (*Bidirectional Encoder Representations from Transformers*) (Alammar, 2019) são pré-treinados em grandes bases de textos em uma tarefa de modelagem de linguagem para "aprender" a próxima palavra. Esse pré-treinamento permite que eles capturem profundamente a sintaxe, semântica e até mesmo algum conhecimento presente nos dados. Paralelamente, a Visão Computacional buscou dotar as máquinas da capacidade de "ver" e interpretar imagens. A revolução nesta área foi impulsionada pelas Redes Neurais Convolucionais (CNNs) (Goodfellow; Bengio; Courville, 2016):

- **Conceito Básico das CNNs:** O operador convolucional aplica filtros (ou *kernels*) que "deslizem" sobre a imagem, detectando padrões locais como bordas, fronteiras, texturas e formas. As camadas mais profundas combinam essas características para identificar padrões complexos, como olhos, nariz ou até mesmo objetos inteiros.

- **Avanços com Modelos Pré-treinados:** A rede *AlexNet* demonstrou a superioridade das CNNs profundas, vencendo uma competição de reconhecimento de imagem por uma margem significativa. Este fato inaugurou a era do *deep learning* na área de visão computacional. Modelos como ResNet, VGG e, mais recentemente, os *Vision Transformers* (ViTs), que adaptam a arquitetura *Transformer* para *patches* de imagem.

Hoje, a Visão Computacional é usada desde sistemas de segurança com reconhecimento facial e inspeção de qualidade em linhas de produção até diagnósticos por imagem na medicina e a calibragem de sensores em veículos autônomos.

Durante anos, LLMs e Visão Computacional evoluíram como campos separados, tratando modalidades distintas de dados. A fronteira mais promissora da

IA contemporânea reside justamente na sua integração, criando sistemas multimodais. A chave para essa convergência é a tradução de diferentes modalidades para um espaço vetorial comum. Assim como uma LLM representa palavras como vetores densos (*embeddings*) em um espaço semântico vetorial, uma rede de visão computacional (como uma CNN ou um ViT) pode representar uma imagem como um vetor em um espaço de características visuais. O ponto fundamental e marcante neste processo ocorre quando se treina um modelo para alinhar esses espaços (Idrees, 2023).

Modelos como CLIP (*Contrastive Language–Image Pre-training*) da OpenAI são exemplos desta quebra de paradigma. O CLIP é treinado com milhões de pares (imagem, legenda). O objetivo é maximizar a similaridade entre os *embeddings* da imagem e da legenda correta, e minimizar a similaridade com legendas incorretas. Após o treinamento, o modelo adquire uma capacidade notável: compreender o conteúdo visual em termos linguísticos. Isso permite tarefas de classificação *zero-shot* - "zero exemplos" ou "zero disparos" de treinamento para aquela classe específica), simplesmente comparando o *embedding* da imagem com *embeddings* de descrições textuais das categorias (Data Science Academy).

Avanços recentes, como o Multimodal LLMs, levam este conceito adiante. Eles utilizam um "projetor" (*projector*) neural que transforma os *embeddings* visuais de um modelo de visão em um formato que pode ser interpretado por uma LLM pré-treinada. Dessa forma, a LLM atua como um cérebro central que processa tanto a entrada textual quanto a entrada visual projetada, permitindo diálogos complexos sobre imagens, geração de descrições detalhadas (*alt text*) e até mesmo resposta a perguntas que requerem raciocínio sobre o conteúdo visual (Data Science Academy).

A jornada da IA, desde suas origens simbólicas até os atuais modelos multimodais, reflete uma busca por uma percepção mais integrada e contextualizada. As LLMs, baseadas na arquitetura *Transformer*, conquistaram uma compreensão profunda da linguagem. A Visão Computacional, alavancada por *CNNs* e *Vision Transformers* (ViTs), conferiu às máquinas a possibilidade de "visão" aguçada. Esta convergência está pavimentando o caminho para assistentes verdadeiramente inteligentes, robôs mais autônomos e ferramentas de criatividade aumentada, redefinindo os limites do possível na interação homem-máquina.

REFERÊNCIAS

ALAMMAR, J. A Visual Guide to Using BERT for the First Time. *Blog de Jay Alammr*, 26 nov. 2019. Disponível em: <https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/>. Acesso em: 21 set. 2025.

ALAMMAR, J.; GROOTENDORST, M. Hands-On Large Language Models. Sebastopol: O'Reilly Media, 2024.

CHARNIAK, E. Introduction to Deep Learning. Cambridge: MIT Press, 2019.

DATA SCIENCE ACADEMY. Deep Learning Book, 2025. Disponível em: <https://www.deeplearningbook.com.br>. Acesso em: 10 Set. de 2025

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

IDREES, H. Vision Transformer vs CNN: a comparison of two image processing giants. *Medium*, 2023. Disponível em: <https://medium.com/@hassaanidrees7/vision-transformer-vs-cnn-a-comparison-of-two-image-processing-giants-d6c85296f34f>. Acesso em: 22 set. 2025.

RUSSELL, S.; NORVIG, P. Artificial Intelligence: A Modern Approach. 4th ed. Harlow: Pearson, 2021.

APÊNDICE 1 - INTRODUÇÃO À INTELIGÊNCIA

A – ENUNCIADO

1 ChatGPT

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

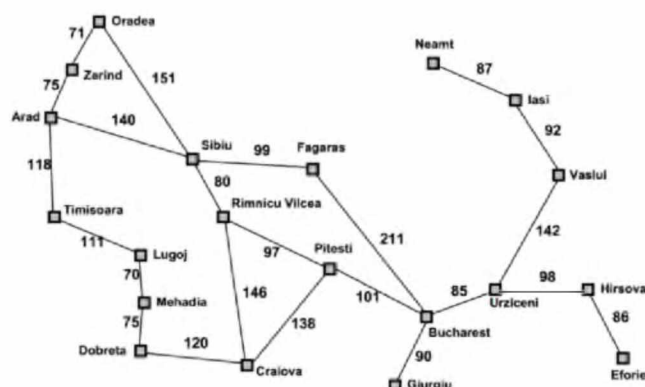
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

Se as uvas caem, então a raposa as come

Se a raposa as come, então estão maduras

As uvas estão verdes ou caem

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem

q: a raposa come as uvas

r: as uvas estão maduras

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

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

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

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

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

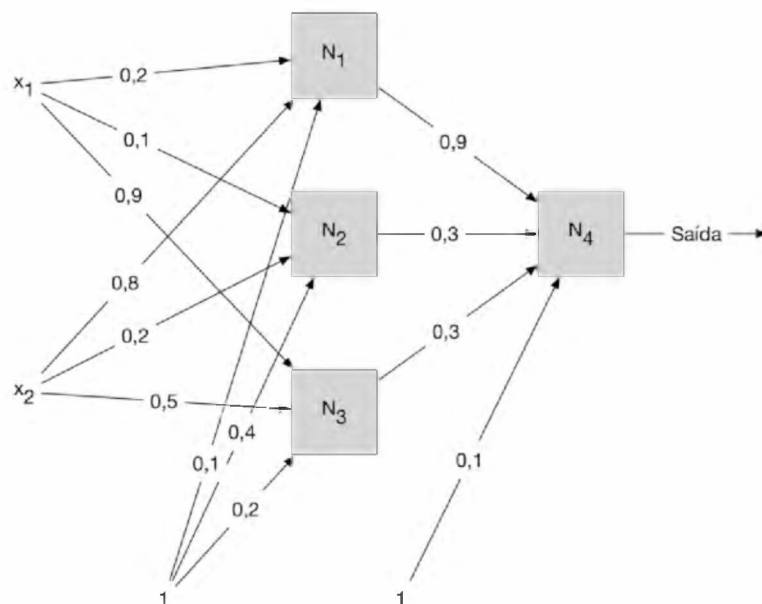
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



Os neurônios 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 - O que é ChatGPT:

Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e tecnologias capazes de realizar tarefas que, quando realizadas por humanos, geralmente exigem inteligência. Estas tarefas incluem, mas não se limitam a, aprendizado, raciocínio, resolução de problemas, reconhecimento de padrões, compreensão da linguagem natural e percepção sensorial.

A IA busca criar sistemas que possam simular ou replicar a capacidade humana de pensar, aprender, perceber, tomar decisões e resolver problemas. Isso é frequentemente alcançado através do uso de algoritmos, modelos matemáticos e técnicas de aprendizado de máquina, onde os sistemas são treinados com grandes volumes de dados para reconhecer padrões e fazer previsões ou tomar decisões.

Existem várias abordagens e subcampos dentro da IA, incluindo aprendizado de máquina, visão computacional, processamento de linguagem natural, robótica e sistemas especialistas. A IA tem aplicações em uma ampla gama de setores, incluindo medicina, finanças, manufatura, automotivo, entretenimento e muitos outros.

B – Classificação:

Abordagem: PENSAR COMO HUMANOS

A resposta é dada de acordo com as palavras que fornecemos como entrada para elaboração da resposta dada como inteligente. Esta resposta pode ter sido gerada de acordo com a base de dados do ChatGPT, relacionando justamente com o que estamos perguntando como entrada, não de maneira fruto de raciocínio e reflexões do Chat, mas uma maneira de gerar informações inteligentes como nós humanos fazemos.

C – Como funciona ChatGPT

O funcionamento do Chat GPT tem como base a técnica de modelagem de linguagem, isto é, obter conhecimento a partir de informações formatadas e padronizadas, tendo como origem uma base gigantesca de dados, no qual existe a entrada de dados seja por nós usuários e outras informações que existem na internet.

Porém devemos relevar muitas das respostas fornecidas pela IA, pois como sabemos pode haver respostas erradas ou que não tenha uma certa clareza, e isso se dá justamente por essas fontes de informações serem coletadas de diversos lugares.

Referências:

<https://br.hubspot.com/blog/marketing/chatgpt>

https://www.dca.fee.unicamp.br/cursos/EA871/references/complementos_ea871/

[linguagens_modelagem.pdf](#)

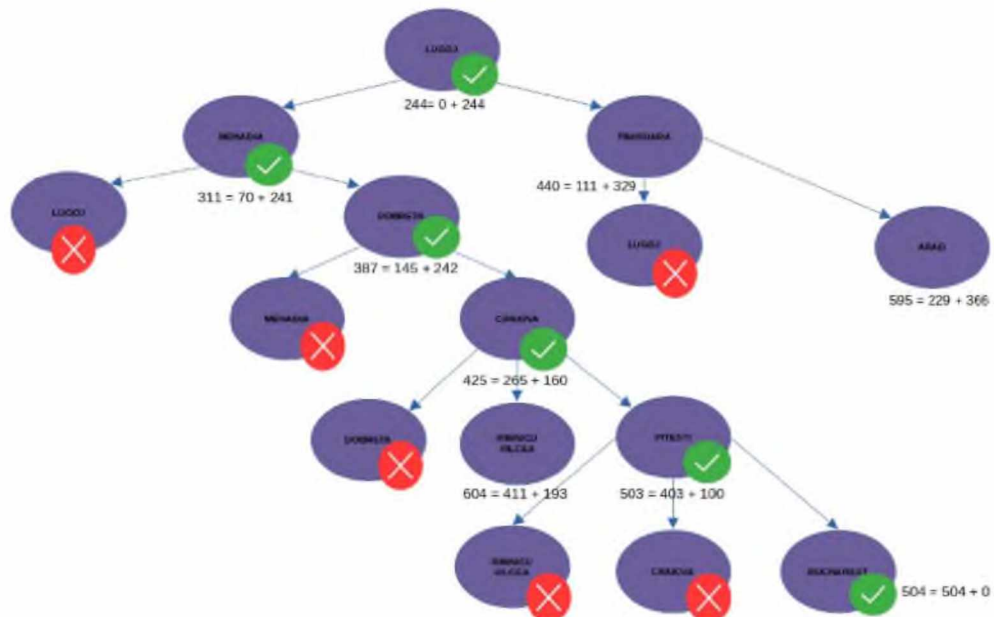
D – Agir racionalmente

Poderíamos tentar relacionar com o "Agir como humano" pois como ele possui uma base gigantesca, ele estaria apenas encontrando as melhores palavras/frases que foram fornecidas por um humano, porém isto não faria sentido, visto que se olharmos seu funcionamento, percebemos que existe um "treinamento" por parte da IA para que possa aprender novos conhecimentos e caminhos para ser seguido, fazendo assim a representação do raciocínio.

O Chat GPT tem limitações para entender contextos e agir diante de informações em conflito entre si ou opiniões. E seu propósito principal é responder perguntas em modo chat de acordo com sua base de dados, portanto pode gerar resposta sem sentido dependendo da pergunta. Através de experimentos, ele tem sim capacidade de aprendizado com as perguntas de usuários, sem paixões e emoções ele se corrigi o tempo todo quando insistimos e mudamos os contextos de perguntas, sendo mais lógico que nós mesmos que colocamos a emoção na frente da lógica em várias circunstâncias, principalmente quando envolvem vaidades.

2 Busca Heurística

Heurística: **Lugoj → Bucharest**



Distância Lugoj até Bucharest é de 504

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: $\neg R \vee P$

R4: Equivalência Implicação R3

$R \rightarrow P$

R5: Silogismo Hipotético R2 e R4

$Q \rightarrow P$

R6: Conjunção R1 e R5

$(P \rightarrow Q) \wedge (Q \rightarrow P)$

R7: Equivalência Bicondicional R6

$P \leftrightarrow Q$

4 – Redes Neurais Artificiais

Os neurônios N1, N2 e N3 possuem função de ativação linear. Já N4 possui função de ativação tangente hiperbólica.

Sendo:

$x_1 = -3$

$x_2 = 1$

$N1 = (-3 \cdot 0,2) + (1 \cdot 0,8) + (0,1) = 0,3 = fa(u) = u$

$N2 = (-3 \cdot 0,1) + (1 \cdot 0,2) + (0,4) = 0,3 = fa(u) = u$

$N3 = (-3 \cdot 0,9) + (1 \cdot 0,5) + (0,2) = -2 = fa(u) = u$

$N4 = (0,3 \cdot 0,9) + (0,3 \cdot 0,3) + (-2 \cdot 0,3) + (0,1) = -0,14$

$fa(u) = (EXP(-0,14) - EXP(0,14)) / (EXP(-0,14) + EXP(0,14))$

$fa(u) = -0,1390$

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

2 Visualização dos dados

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

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

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

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

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

- 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 R²
- h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

B - RESOLUÇÃO

a) Importação bibliotecas e carga da base de dados - DATASET

```
<class 'pandas.core.frame.DataFrame'>
```

RangeIndex: 202297 entries, 0 to 202296

Data columns (total 11 columns):

#	Column	Non-Null	Count	Dtype
0	year_of_reference	202297	non-null	int64
1	month_of_reference	202297	non-null	object
2	fipe_code	202297	non-null	object
3	authentication	202297	non-null	object
4	brand	202297	non-null	object
5	model	202297	non-null	object
6	fuel	202297	non-null	object
7	gear	202297	non-null	object
8	engine_size	202297	non-null	object
9	year_model	202297	non-null	int64
10	avg_price_brl	202297	non-null	int64

dtypes: int64(3), object(8)

memory usage: 17.0+ MB

	YEAR_OF_REFERENCE	MONTH_OF_REFERENCE	FIPE_CODE	AUTHENTICATION \
0	2021	JANUARY	004001-0	CFZLCTZFWRCP
1	2021	JANUARY	004001-0	CDQWXWPW3Y2P
2	2021	JANUARY	004001-0	CB1T3XWWJ1XP
3	2021	JANUARY	004001-0	CB9GCT6J65R0
4	2021	JANUARY	004003-7	G15WG0GBZ1FX

	BRAND	MODEL	FUEL	GEAR \
0	GM - CHEVROLET	CORSA WIND 1.0 MPFI / EFI 2P	GASOLINE	MANUAL
1	GM - CHEVROLET	CORSA WIND 1.0 MPFI / EFI 2P	GASOLINE	MANUAL
2	GM - CHEVROLET	CORSA WIND 1.0 MPFI / EFI 2P	GASOLINE	MANUAL
3	GM - CHEVROLET	CORSA WIND 1.0 MPFI / EFI 2P	ALCOHOL	MANUAL
4	GM - CHEVROLET	CORSA PICK-UP GL/ CHAMP 1.6 MPFI / EFI	GASOLINE	MANUAL

	ENGINE_SIZE	YEAR_MODEL	AVG_PRICE_BRL
0	1	2002	9162
1	1	2001	8832
2	1	2000	8388
3	1	2000	8453
4	1,6	2001	12525

B) VERIFICAÇÃO DE VALORES FALTANTES:

YEAR_OF_REFERENCE	FALSE
MONTH_OF_REFERENCE	FALSE
FIPE_CODE	FALSE
AUTHENTICATION	FALSE
BRAND	FALSE
MODEL	FALSE
FUEL	FALSE
GEAR	FALSE

```
ENGINE_SIZE      FALSE
YEAR_MODEL       FALSE
AVG_PRICE_BRL    FALSE
DTYPE: BOOL
```

QUANTIDADE FALTANTE:0

C) CHECAGEM SE HÁ DADOS DUPLICADOS
DUPLICADOS: 2
DUPLICADOS REMOVIDOS

D) CRIE DUAS CATEGORIAS, PARA SEPARAR COLUNAS NUMÉRICAS E CATEGÓRICAS.

	YEAR_OF_REFERENCE	YEAR_MODEL	AVG_PRICE_BRL
COUNT	202295.00	202295.00	202295.00
MEAN	2021.56	2011.27	52756.77
STD	0.57	6.38	51628.91
MIN	2021.00	2000.00	6647.00
25%	2021.00	2006.00	22855.00
50%	2022.00	2012.00	38027.00
75%	2022.00	2016.00	64064.00
MAX	2023.00	2023.00	979358.00

	MONTH_OF_REFERENCE	FIPE_CODE	AUTHENTICATION	BRAND \
COUNT	202295	202295	202295	202295
UNIQUE	12	2091	202295	6
TOP	JANUARY	003281-6	CFZLCTZFWRCP	FIAT
FREQ	24260	425	1 44962	

	MODEL	FUEL	GEAR	ENGINE_SIZE
COUNT	202295	202295	202295	202295
UNIQUE	2112	3	2	29
TOP	PALIO WEEK. ADV/ADV TRYON 1.8 MPI FLEX	GASOLINE	MANUAL	1,6
FREQ	425	168684	161883	47420

E) IMPRIMA A CONTAGEM DE VALORES POR MODELO (MODEL) E MARCA DO CARRO (BRAND)

BRAND	MODEL	
FIAT	PALIO WEEK. ADV/ADV TRYON 1.8 MPI FLEX	425
	DOBLO ADV/ADV TRYON/LOCKER 1.8 FLEX	375
	DOBLO CARGO 1.8 MPI FIRE FLEX 8V/16V 4P	318
	PALIO WEEKEND ADVENTURE LOCKER 1.8 FLEX	300
	SIENA 1.0/ EX 1.0 MPI FIRE/ FIRE FLEX 8V	300

...

VW - VOLKSWAGEN	POLO HIGHLINE TSI 1.0 FLEX 12V AUT.	8
	GOL LAST EDITION 1.0 FLEX 12V 5P	2
	POLO TRACK 1.0 FLEX 12V 5P	2
	SAVEIRO ROBUST 1.6 TOTAL FLEX 16V	2
	SAVEIRO ROBUST 1.6 TOTAL FLEX 16V CD	2

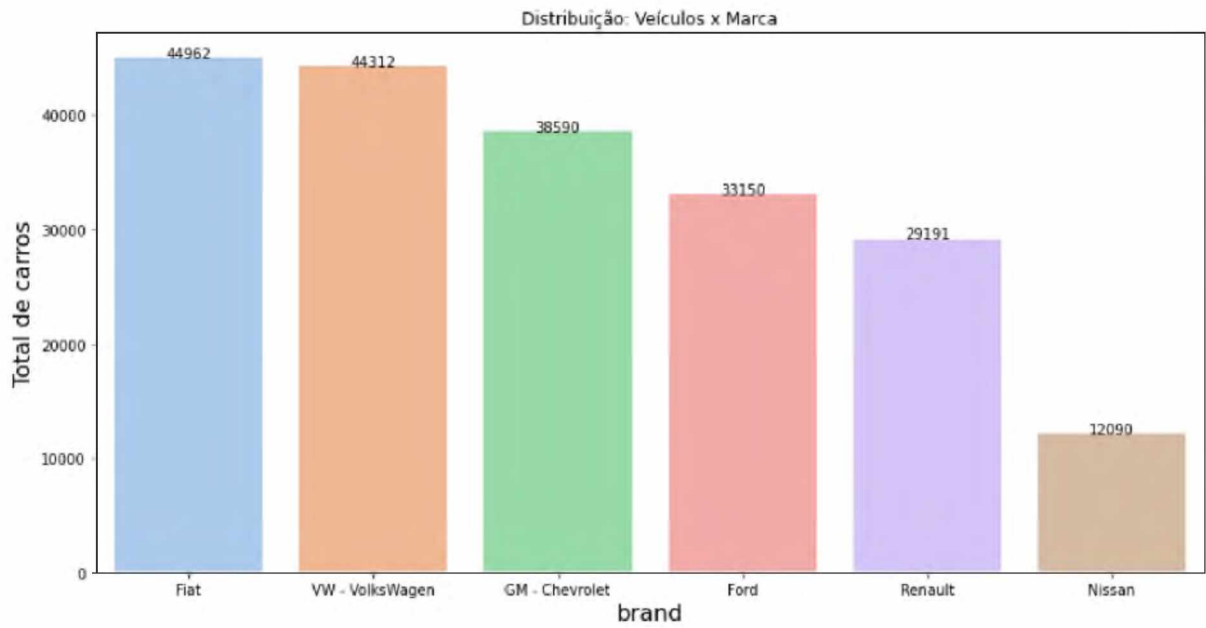
NAME: MODEL, LENGTH: 2112, DTYPE: INT64

F) DÊ UM BREVE EXPLICAÇÃO (MÁXIMO DE QUATRO LINHAS) SOBRE OS PRINCIPAIS RESULTADOS ENCONTRADOS NA ANÁLISE EXPLORATÓRIA DOS DADOS:

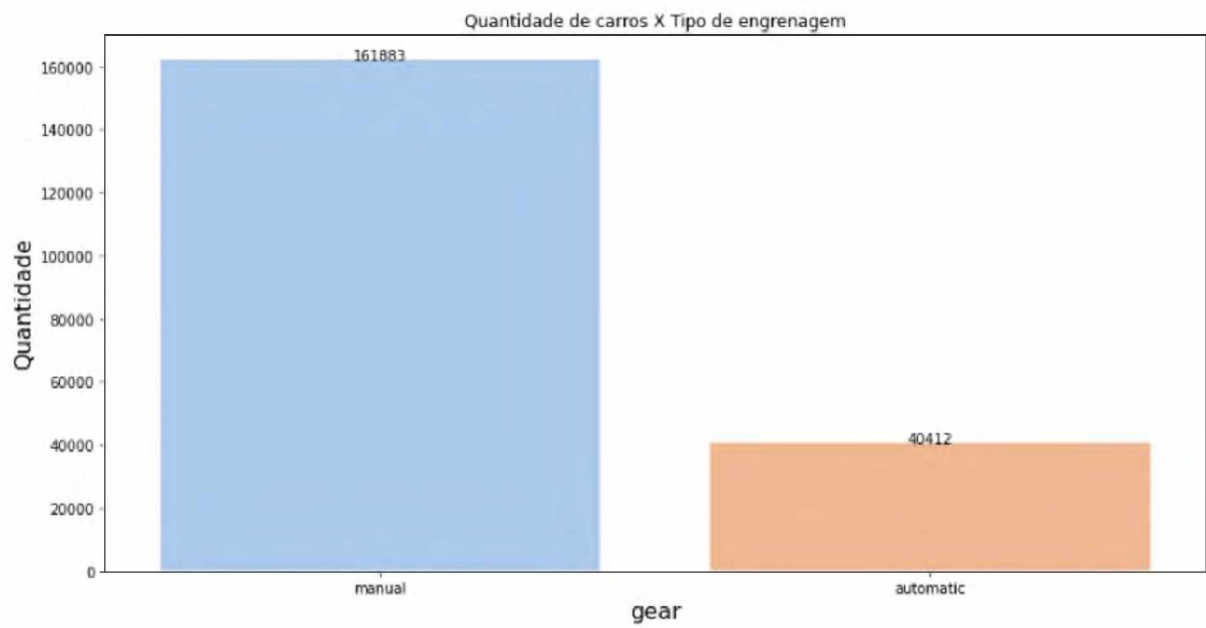
COM O DATASET TRABALHADO DESDE O INÍCIO DA ANÁLISE EXPLORATÓRIA, OBSERVAMOS QUE O DATASET POSSUI 202297 LINHAS E 11 COLUNAS, SENDO QUE DESTAS LINHAS APENAS DUAS SÃO DUPLICADAS, E FORAM EXCLUÍDAS. NÃO FOI ENCONTRADO INFORMAÇÕES FALTANTES NO DATASET. 8 COLUNAS SÃO DO TIPO CATEGÓRICAS, E APENAS 3 SÃO NUMERICAS. AS COLUNAS CATEGORICAS BRAND, FUEL, GEAR, MONTH_OF_REFERENCE SERÃO CONVERTIDAS PARA NUMERICAS MAIS ADIANTE.

2 - VISUALIZAÇÃO DOS DADOS

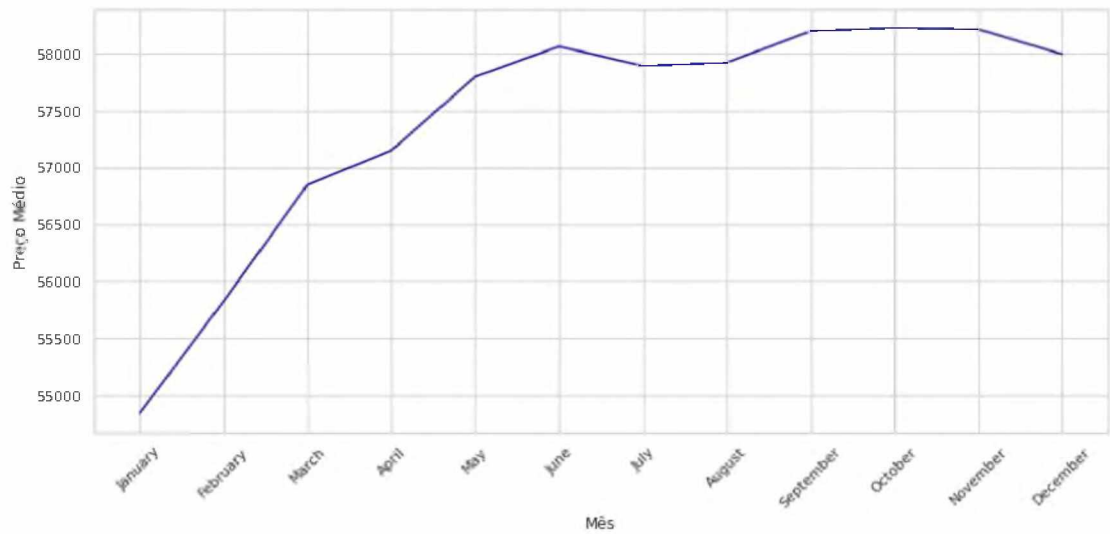
A) GRÁFICO DA DISTRIBUIÇÃO DA QUANTIDADE DE CARROS X MARCA



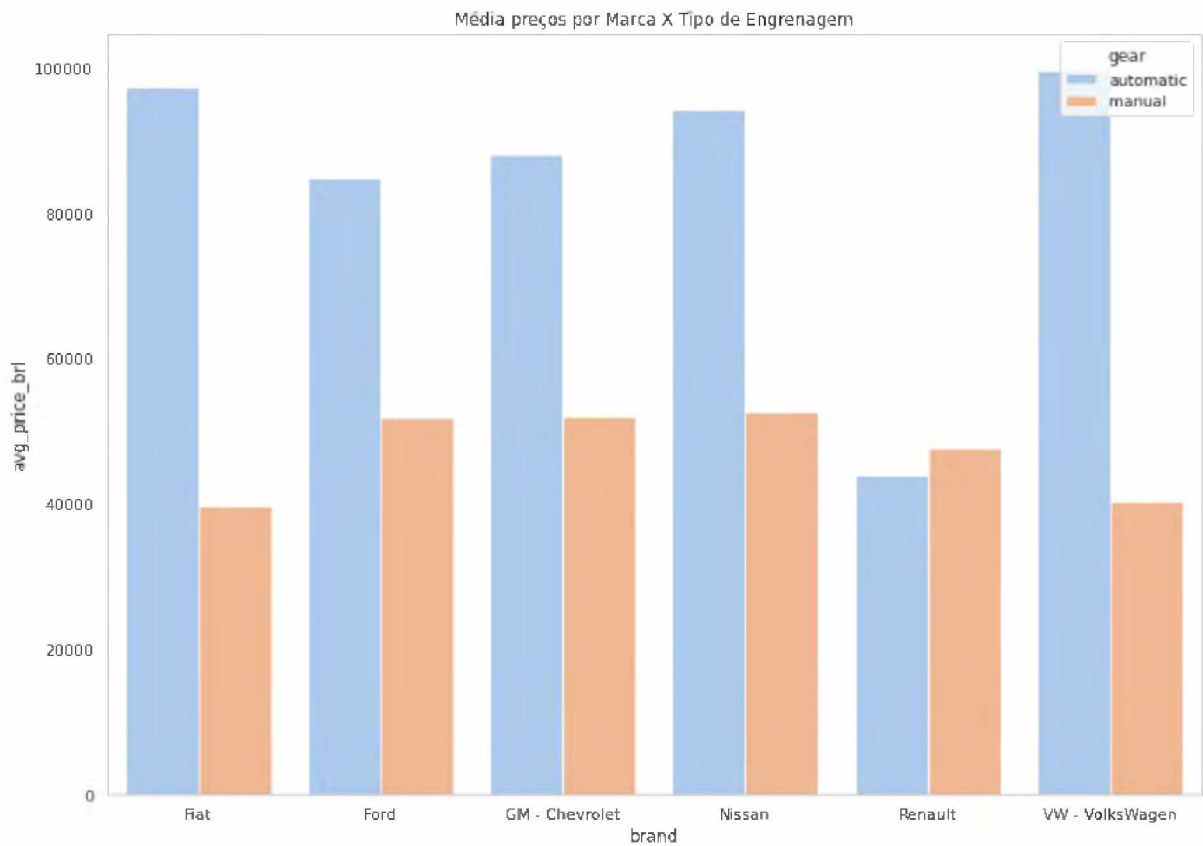
B) GRÁFICO DA DISTRIBUIÇÃO DA QUANTIDADE DE CARROS X TIPO DE ENGRENAGEM



C) GRÁFICO DA EVOLUÇÃO DA MÉDIA DE PREÇO DOS CARROS AO LONGO DOS MESES DE 2022

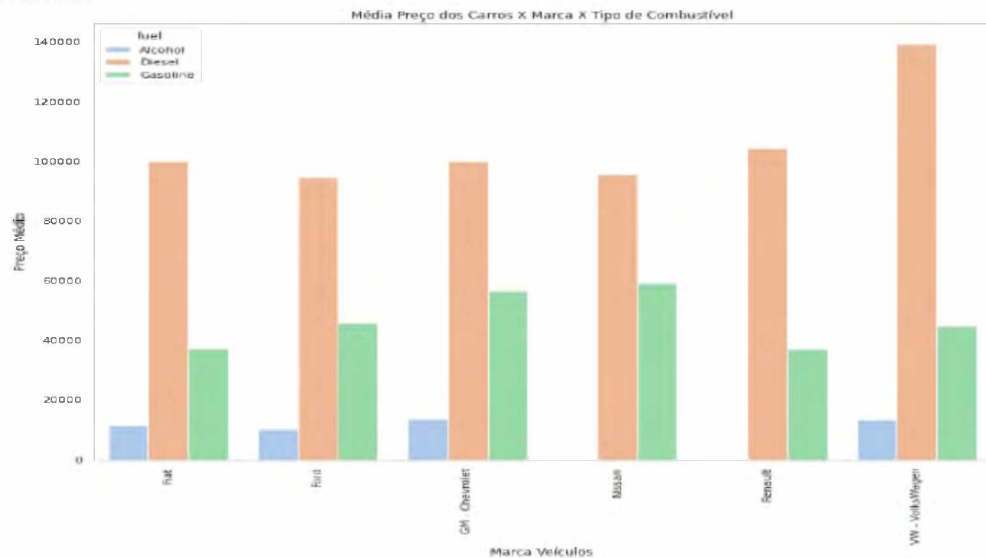


D) GRÁFICO DA DISTRIBUIÇÃO DA MÉDIA DE PREÇO DOS CARROS X MARCA E TIPO DE ENGRENAGEM



E) DÊ UMA BREVE EXPLICAÇÃO SOBRE OS RESULTADOS GERADOS NO ITEM D.
O GRÁFICO MOSTRA QUE AS MARCAS COM PREÇOS MAIS ALTOS NO BRASIL SÃO FIAT E VOLKSWAGEN COM CÂMBIO AUTOMÁTICO E A COM PREÇO MÉDIO MAIS BAIXO É A FIAT COM CÂMBIO MANUAL.
OS CARROS COM CÂMBIO AUTOMÁTICO SÃO OS MAIS CAROS ENTRE TODAS AS MARCAS, EXCETO RENAULT. SOMENTE A MARCA RENAULT POSSUI CARROS COM CÂMBIO MANUAL MAIS CAROS QUE AS VERSÕES COM CÂMBIO AUTOMÁTICO DENTRO DA MESMA MARCA.

F) GRÁFICO DA DISTRIBUIÇÃO DA MÉDIA PREÇO DOS CARROS X MARCA E TIPO DE COMBUSTÍVEL



G) DÊ UMA BREVE EXPLICAÇÃO SOBRE OS RESULTADOS GERADOS NO ITEM F

O GRÁFICO ACIMA INDICA QUE NO BRASIL, DE MANEIRA GERAL, OS VEÍCULOS MOVIDOS À DIESEL SÃO OS MAIS CAROS. A MARCA VOLKSWAGEN - DIESEL - POSSUÍ A MÉDIA DE PREÇO MAIS ALTA PARA ESSA CATEGORIA.

OS VEÍCULOS VERSÕES A GASOLINA SÃO PRODUZIDOS POR TODAS AS MARCAS E SEU PREÇO MÉDIO É BEM MAIOR DE MANEIRA GERAL DAS VERSÕES A ALCOOL.

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

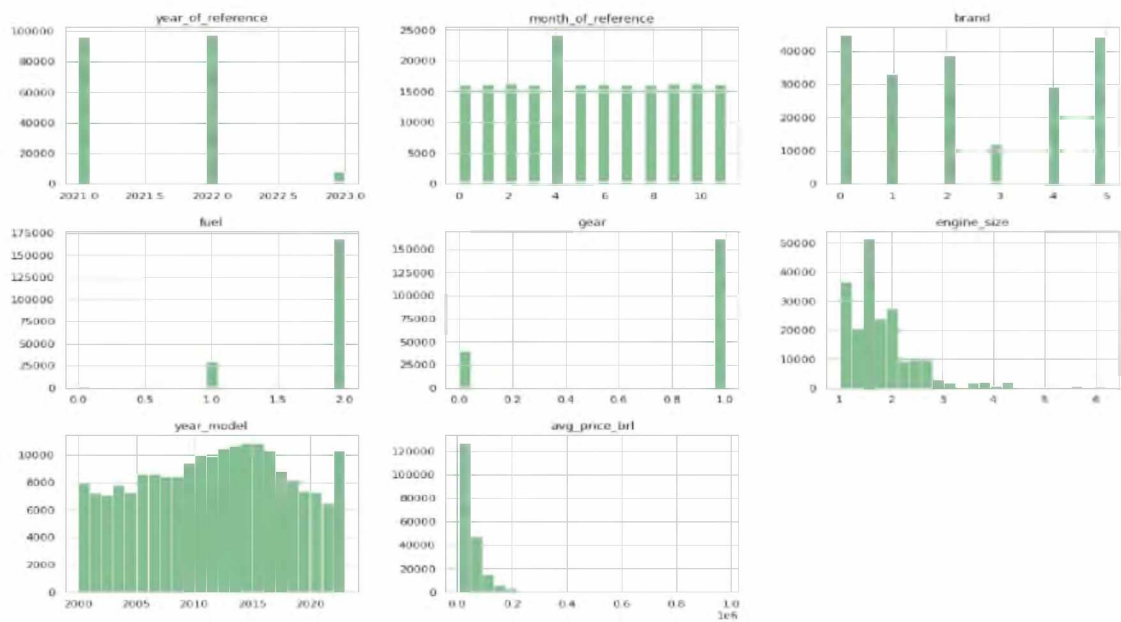
YEAR_OF_REFERENCE	INT64
MONTH_OF_REFERENCE	OBJECT
FIPE_CODE	OBJECT
AUTHENTICATION	OBJECT
BRAND	OBJECT
MODEL	OBJECT
FUEL	OBJECT
GEAR	OBJECT
ENGINE_SIZE	OBJECT
YEAR_MODEL	INT64
AVG_PRICE_BRL	INT64
DTYPE: OBJECT	

	YEAR_OF_REFERENCE	MONTH_OF_REFERENCE	PIPE_CODE	AUTHENTICATION	BRAND \
0	2021	4	004001-0	CFZLCTZFWRCP	2
1	2021	4	004001-0	CDQWXWPW3Y2P	2
2	2021	4	004001-0	CB1T3XWWJ1XP	2
3	2021	4	004001-0	CB9GCT6J65R0	2
4	2021	4	004003-7	G15WG0GBZ1FX	2

	MODEL	FUEL	GEAR	ENGINE_SIZE	YEAR_MODEL \	
0	CORSA WIND 1.0 MPFI / EFI 2P		2	1	1	2002
1	CORSA WIND 1.0 MPFI / EFI 2P		2	1	1	2001
2	CORSA WIND 1.0 MPFI / EFI 2P		2	1	1	2000
3	CORSA WIND 1.0 MPFI / EFI 2P		0	1	1	2000
4	CORSA PICK-UP GL/ CHAMP 1.6 MPFI / EFI		2	1	1,6	2001

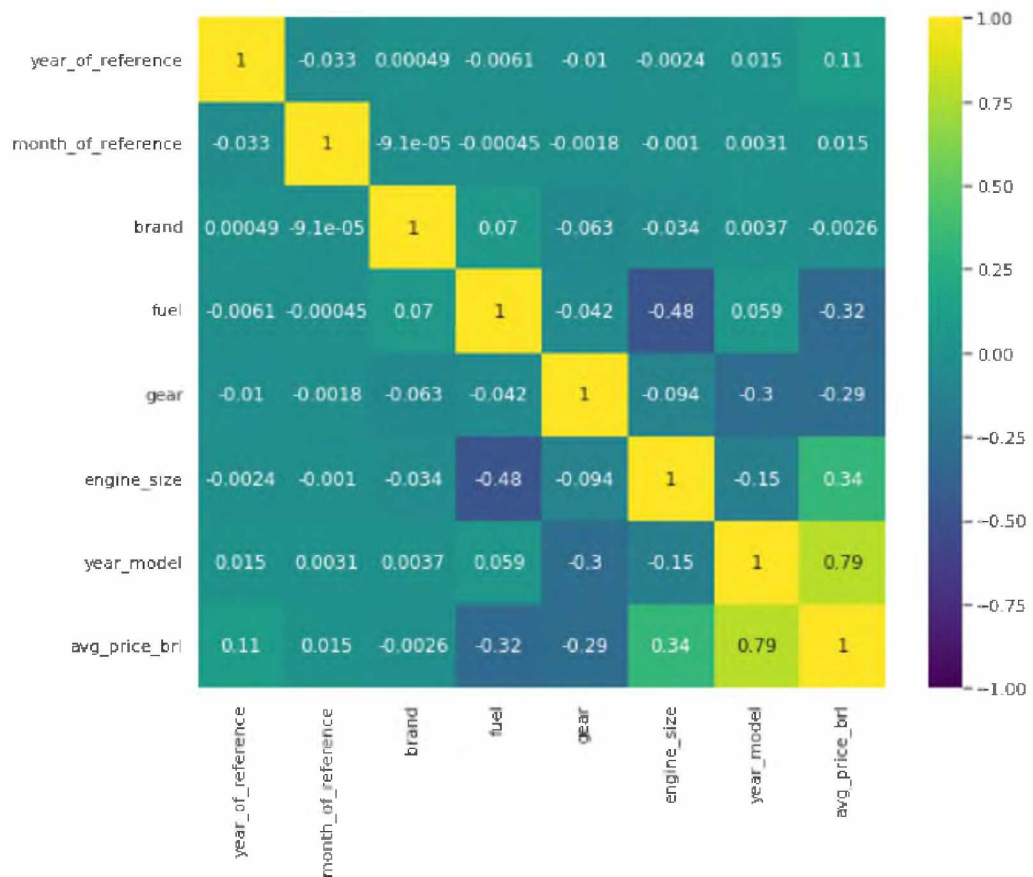
	AVG_PRICE_BRL
0	9162
1	8832
2	8388
3	8453
4	12525

HISTOGRAMA: AVALIAÇÃO DA NORMALIDADE DOS DADOS - DADOS NUMÉRICO (NÃO CATEGORICAS)

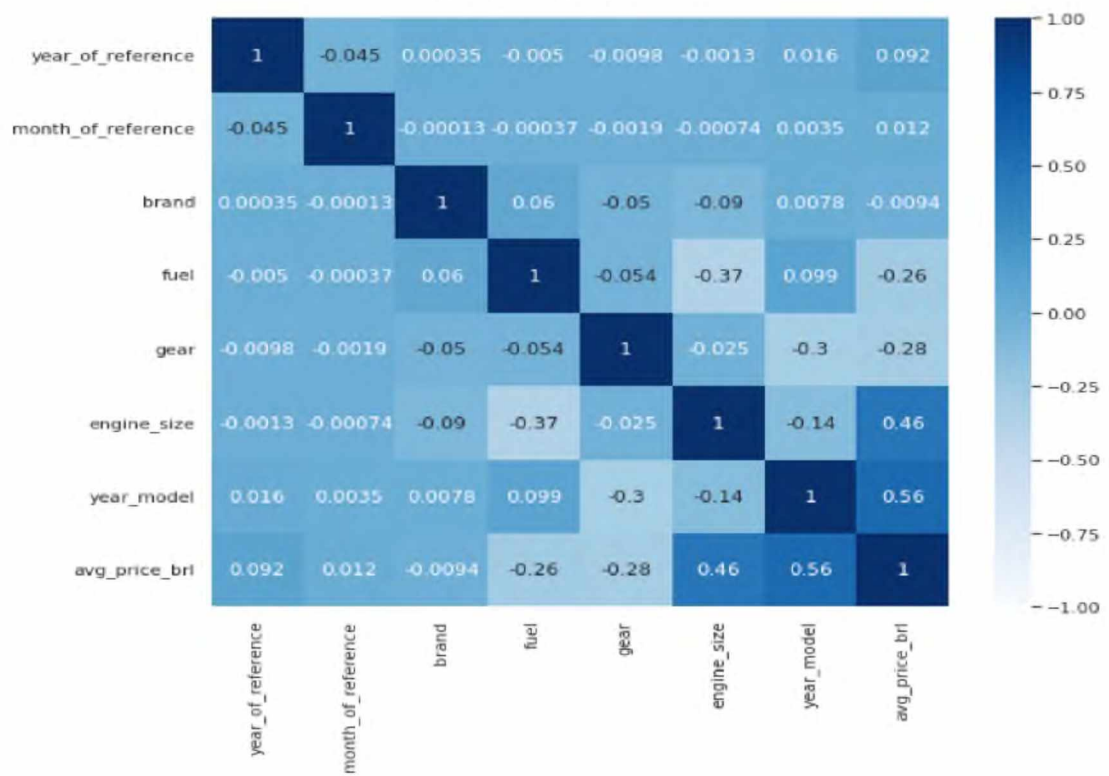


GRÁFICOS DE CORRELAÇÃO SPEARMAN E PEARSON

Mapa de Correlação Spearman



Mapa de Correlação Pearson



B) CRIE PARTIÇÕES CONTENDO 75% DOS DADOS PARA TREINO E 25% PARA TESTE

```
151722 16072
151723 14692
151724 20068
151725 18692
151726 19197
151727 15332
151728 14191
151729 13390
151730 15387
151731 13589
```

NAME: AVG_PRICE_BRL, DTYPE: INT64

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.

RANDOMFOREST

```
RANDOMFORESTREGRESSOR(N_ESTIMATORS=100, *, CRITERION='SQUARED_ERROR', MAX_DEPTH=None,
MIN_SAMPLES_SPLIT=2, MIN_SAMPLES_LEAF=1, MIN_WEIGHT_FRACTION_LEAF=0.0, MAX_FEATURES=1.0,
MAX_LEAF_NODES=None, MIN_IMPURITY_DECREASE=0.0, BOOTSTRAP=True, OOB_SCORE=False,
N_JOBS=None, RANDOM_STATE=None, VERBOSE=0, WARM_START=False, CCP_ALPHA=0.0,
MAX_SAMPLES=None, MONOTONIC_CST=None)
```

IMPORTANCE

ENGINE_SIZE	0.492073
YEAR_MODEL	0.426128
FUEL	0.033441
GEAR	0.024642
BRAND	0.013679
YEAR_OF_REFERENCE	0.008901
MONTH_OF_REFERENCE	0.001136

XGBOOST

```
XGBREGRESSOR(BASE_SCORE=0.5, BOOSTER='GBTREE', COLSAMPLE_BYLEVEL=1, COLSAMPLE_BYNODE=1,
COLSAMPLE_BYTREE=1, ENABLE_CATEGORICAL=False, GAMMA=0, GPU_ID=-1, IMPORTANCE_TYPE=None,
INTERACTION_CONSTRAINTS='', LEARNING_RATE=0.300000012, MAX_DELTA_STEP=0, MAX_DEPTH=32,
MIN_CHILD_WEIGHT=2, MONOTONE_CONSTRAINTS='()', N_ESTIMATORS=100, N_JOBS=8,
NUM_PARALLEL_TREE=1, PREDICTOR='AUTO', RANDOM_STATE=43, REG_ALPHA=0, REG_LAMBDA=1,
CALE_POS_WEIGHT=1, SUBSAMPLE=1, TREE_METHOD='EXACT', VALIDATE_PARAMETERS=1,
VERBOSITY=None)
```

IMPORTANCE

ENGINE_SIZE	0.387461
YEAR_MODEL	0.278951
GEAR	0.132319
FUEL	0.127812
BRAND	0.038601
YEAR_OF_REFERENCE	0.029185
MONTH_OF_REFERENCE	0.005671

F) DÊ UMA BREVE EXPLICAÇÃO SOBRE OS RESULTADOS ENCONTRADOS NA ANÁLISE DE IMPORTÂNCIA DE VARIÁVEIS

PARA RANDOMFOREST, A VARIÁVEL MAIS IMPORTANTE É O ENGINE_SIZE, SEGUIDO DE YEAR_MODEL. AS VARIÁVEIS DE TIPO DE COMBUSTÍVEL (FUEL) E TIPO DE ENGRENAGEM (GEAR) TIVERAM IMPORTÂNCIA BASTANTE TÍMIDAS.

NO MODELO XGBOOST OCORRERAM SEMELHANÇAS E FOI MAIS EQUILIBRADO, COM O TAMANHO DO MOTOR (ENGINE_SIZE) E O ANO DO MODELO (YEAR_MODEL) SENDO AS CARACTERÍSTICAS MAIS IMPORTANTES. VALE CONSIDERAR QUE A IMPORTÂNCIA RELATIVA DAS VARIÁVEIS DE TIPO DE COMBUSTÍVEL (FUEL) E TIPO DE ENGRENAGEM (GEAR) SÃO UM POUCO MELHORES EM COMPARAÇÃO COM O MODELO RANDOMFOREST.

G) ESCOLHA O MELHOR MODELO COM BASE NAS MÉTRICAS DE AVALIAÇÃO MSE, MAE E R2;

RANDOMFOREST

MSE: 306618435.0

MAE: 7440.53

R2: 0.9

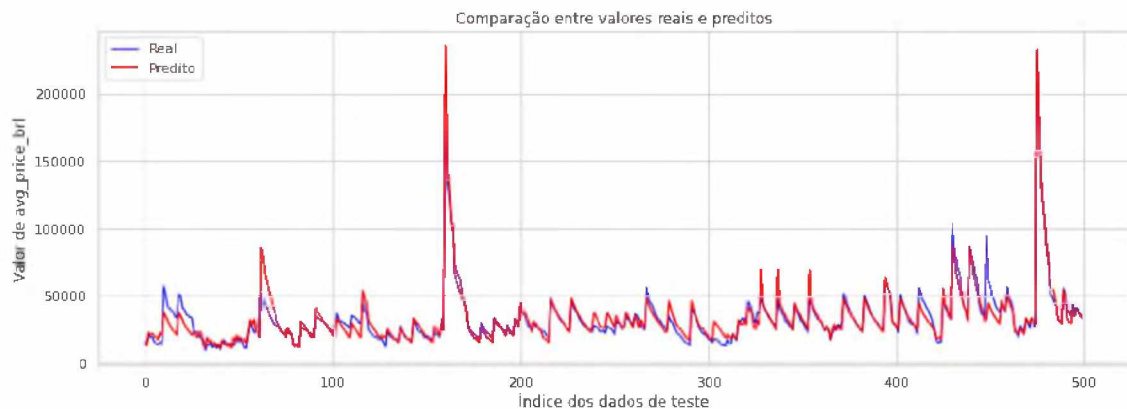
MELHOR MODELO COM MAIS EQUILÍBRIO ENTRE AS VARIÁVEIS SERIA XGBOOST

H) DÊ UMA BREVE EXPLICAÇÃO (MÁXIMO DE QUATRO LINHAS) SOBRE QUAL MODELO GEROU O MELHOR RESULTADO E A MÉTRICA DE AVALIAÇÃO UTILIZADA.

CONCLUSÃO

COM BASE NAS MÉTRICAS DEMONSTRADA POR AMBOS MODELOS, VEMOS QUE A SEMELHANÇA É MUITO PRÓXIMA, PORÉM O MODELO XGB OBTVEU UM VALOR MAIOR DE R2 E MSE BEM MENOR EM COMPARAÇÃO COM O RANDOMFOREST, LOGO, CONSIDERANDO A MÉTRICA R2, O MODELO XGB É O ESCOLHIDO COM 96% DE ACURÁCIA.

VISUALIZAÇÃO DA RELAÇÃO DA PREDIÇÃO COM VALOR REAL – XGBOOST



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

01) Carregando a base de dados de Satellite 'data.frame': 6435 obs. of 5 variables:

Head:

x.17 x.18 x.19 x.20 classes

1 92 112 118 85 grey soil

2 84 103 104 81 grey soil

3 84 99 104 78 grey soil

4 84 99 104 81 grey soil

5 76 99 104 81 grey soil

6 76 99 108 85 grey soil

Sumário:

x.17	x.18	x.19	x.20	classes
Min. : 40.00	Min. : 27.00	Min. : 50.00	Min. : 29.0	red soil :1533
1st Qu.: 60.00	1st Qu.: 71.00	1st Qu.: 85.00	1st Qu.: 69.0	cotton crop : 703
Median : 68.00	Median : 85.00	Median :100.00	Median : 81.0	grey soil :1358
Mean : 69.05	Mean : 83.17	Mean : 99.15	Mean : 82.6	damp grey soil : 626
3rd Qu.: 79.00	3rd Qu.:103.00	3rd Qu.:113.00	3rd Qu.: 92.0	vegetation_stubble: 707
Max. :104.00	Max. :130.00	Max. :145.00	Max. :157.0	very damp grey soil:1508

02) Separando a base para dados de treino(80%) e de teste (20%)

Treino:

'data.frame': 5151 obs. of 5 variables:

Head:

x.17 x.18 x.19 x.20 classes

1 92 112 118 85 grey soil

2 84 103 104 81 grey soil

4 84 99 104 81 grey soil

6 76 99 108 85 grey soil

7 80 112 118 88 grey soil

8 80 107 113 85 grey soil

Teste:

'data.frame': 1284 obs. of 5 variables:

Head:

x.17 x.18 x.19 x.20 classes

3 84 99 104 78 grey soil

5 76 99 104 81 grey soil

12 76 87 91 67 damp grey soil

22 88 99 104 85 grey soil

35 88 99 104 85 grey soil

36 84 103 108 85 grey soil

03) Treinando os dados com os modelos RandomForest, SVM e RNA para
predição.

> # RNA

```
> model_nnet<-train(classes ~.,data=satelite_treino,method="nnet",trace=FALSE,
linout=T)
```

```
> predictions_nnet <- predict(model_nnet, satelite_teste)
```

> # Random Forest

```
> model_rf <- train(classes ~., data=satelite_treino, method="rf")
```

```
> predictions_rf <- predict(model_rf, satelite_teste)
```

> # SVM

```
> model_svm <- train(classes ~., data=satelite_treino, method="svmRadial")
```

```
> predictions_svm <- predict(model_svm, satelite_teste)
```

04) Escolhendo o melhor modelo com base em suas matrizes de confusão

Confusion Matrix and Statistics (NNET)

Reference

Prediction	red_soil	cotton_crop	grey_soil	damp_grey_soil	vegetation_stubble	very_damp_grey_soil
red_soil	288	3	5	4	12	0
cotton crop	2	121	0	0	7	0
grey soil	8	0	260	62	2	32
damp grey soil	0	0	0	0	0	0
vegetation stubble	8	13	0	3	106	15
very damp grey soil	0	3	6	56	14	254

Overall Statistics

Accuracy : 0.8014 95% CI : (0.7785, 0.8229)

No Information Rate : 0.2383

P-Value [Acc > NIR] : < 0.00000000000000022

Kappa : 0.7511 McNemar's Test P-Value : NA

Confusion Matrix and Statistics (Random Forest)

Reference

Prediction	red_soil	cotton_crop	grey_soil	damp_grey_soil	vegetation_stubble	very_damp_grey_soil
red soil	296	1	5	2	9	0
cotton crop	0	123	0	0	1	1
grey soil	3	0	239	29	1	10
damp grey soil	1	0	21	64	2	42
vegetation stubble	6	10	0	0	121	8
very damp grey soil	0	6	6	30	7	240

Overall Statistics

Accuracy : 0.8435 95% CI : (0.8224, 0.8629)

No Information Rate : 0.2383

P-Value [Acc > NIR] : < 0.00000000000000022

Kappa : 0.8067 McNemar's Test P-Value : NA

Confusion Matrix and Statistics (SVM)

Reference

Prediction	red_soil	cotton_crop	grey_soil	damp_grey_soil	vegetation_stubble	very_damp_grey_soil
red soil	299	1	4	2	7	0
cotton crop	1	120	0	0	4	0
grey soil	4	0	260	29	1	12
damp grey soil	0	1	7	70	2	34
vegetation stubble	2	14	0	2	117	3
very damp grey soil	0	4	0	22	10	252

Overall Statistics

Accuracy : 0.8707 95% CI : (0.8511, 0.8886)

No Information Rate : 0.2383

P-Value [Acc > NIR] : < 0.00000000000000022

Kappa : 0.84 McNemar's Test P-Value : NA

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

Modelo SVM obteve acurácia de 87.07%

2.1) Carga de dados Volumes.csv

```
> head(dados_volumes)
```

NR DAP HT HP VOL

1 1 34.0 27.00 1.80 0.8971441

2 2 41.5 27.95 2.75 1.6204441

3 3 29.6 26.35 1.15 0.8008181

4 4 34.3 27.15 1.95 1.0791682

5 5 34.5 26.20 1.00 0.9801112

6 6 29.9 27.10 1.90 0.9067022

2.2) Eliminando a coluna NR, que apresenta apenas números sequenciais.

```
> dados_volumes$NR <- NULL
```

```
> head(dados_volumes)
```

```
DAP HT HP VOL
```

```
1 34.0 27.00 1.80 0.8971441
```

```
2 41.5 27.95 2.75 1.6204441
```

```
3 29.6 26.35 1.15 0.8008181
```

```
4 34.3 27.15 1.95 1.0791682
```

```
5 34.5 26.20 1.00 0.9801112
```

```
6 29.9 27.10 1.90 0.9067022
```

2.3) Separando a base de dados em base de treino e base de teste

Treino:

'data.frame': 80 obs. of 4 variables:

\$ DAP: num 34 41.5 29.6 34.5 29.9 29.5 36.3 32.5 28.9 32.6 ...

\$ HT : num 27 27.9 26.4 26.2 27.1 ...

\$ HP : num 1.8 2.75 1.15 1 1.9 2.4 1.8 1.45 2.65 2.5 ...

\$ VOL: num 0.897 1.62 0.801 0.98 0.907 ...

Teste:

'data.frame': 20 obs. of 4 variables:

\$ DAP: num 34.3 28.4 36.3 33.6 34.5 42.6 31.5 27.5 33 36.5 ...

\$ HT : num 27.1 25.4 26.7 25.1 27.3 ...

\$ HP : num 1.95 2.3 1.5 2 2.1 3.4 2.6 2.2 1.3 1.75 ...

\$ VOL: num 1.079 0.773 1.387 0.967 1.236 ...

2.4) Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM

(svmRadial), Redes Neurais (neuralnet/NNET) e o modelo alométrico de SPURR

2.4.a) Modelo Random Forest

```
model_rf_volume <- caret::train(VOL~., data=dados_volumes_treino, method="rf")
```

```
mtry RMSE Rsquared MAE
```

```
2 0.1664953 0.8472658 0.1278774
```

```
3 0.1589899 0.8524354 0.1228492
```

2.4.b) Modelo SVM

```
model_svm_volume <- caret::train(VOL~., data=dados_volumes_treino, method="svmRadial")
```

```
C RMSE Rsquared MAE
```

```
0.25 0.3090099 0.5444742 0.2098011
```

```
0.50 0.2817418 0.6120694 0.1937495
```

```
1.00 0.2578169 0.6651132 0.1809169
```

2.4.c) Modelo Redes Neurais

```
model_rn_volume <- caret::train(VOL~., data=dados_volumes_treino, method="nnet", linout=TRUE)
```

```
iter 90 value 2.271030
```

```
iter 100 value 2.269182
```

```
final value 2.269182
```

```
stopped after 100 iterations
```

2.4.d) Modelo Alométrico de SPURR

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

```
alom <- nls(
```

```
VOL ~ b0 + b1*DAP*DAP*HT,
```

```
dados_volumes_treino,
```

```
start=list(b0=0.5, b1=0.5))
```

```
)
model_spurr_volume <- alom
```

2.5) Efetue as predições nos dados de teste

5.a) Predição para o modelo rf

```
predicao_rf_volume <- predict(model_rf_volume, dados_volumes_teste)
unique(predicao_rf_volume)
[1] 1.1969689 0.8224443 1.3217340 1.2158075 1.2129261 1.7710853 1.0655016 0.8171106 1.1375637 1.3264883 1.6379562
1.1764377
2.2178678
[14] 1.1453579 0.8261701 1.1847082 1.4703531 1.1409063 1.6598771 1.7580410
```

5.b) Predição para o modelo SVM

```
predicao_svm_volume <- predict(model_svm_volume, dados_volumes_teste)
unique(predicao_svm_volume)
[1] 1.2293364 0.8775180 1.3264398 1.2834004 1.2392926 1.9908587 1.0327566 0.8837195 1.1095961 1.5455772 1.6402291
1.2419872
1.6782950
[14] 1.1808122 1.3579021 1.2861189 1.4348539 0.9977521 1.6567754 1.7076424
```

5.c) Predição para o modelo Rede Neural

```
predicao_rn_volume <- predict(model_rn_volume, dados_volumes_teste)
unique(predicao_rn_volume)
[1] 1.1927518 0.8815652 1.3081312 1.2327408 1.2093461 1.8736070 1.1062966 0.8124909 1.1778419 1.4300070 1.5839437
1.2016388 2.0594623
[14] 1.2089814 1.0376702 1.3059925 1.3973667 0.9895347 1.6342365 1.7425025
```

5.d) Predição para o modelo Alométrico SPURR

```
predicao_spurr_volume <- predict(model_spurr_volume, dados_volumes_teste)
unique(predicao_spurr_volume)
[1] 1.2580293 0.8177997 1.3825694 1.1194932 1.2792479 1.8786676 1.0105002 0.7657835 1.0516512 1.3027902 1.7293094
1.2655252 2.0811068
[14] 1.0742044 0.7930147 1.2197827 1.5055969 1.0545348 1.5894885 2.0130206
```

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

Funções R2, Syx, SyxPerc:

```
R2 <- function (Y_real, Y_pred){
  return (1 - ( sum((Y_real - Y_pred)^2))/ (sum( (Y_real - mean(Y_real))^2 )))
}
Syx <- function (Y_real, Y_pred){
  return (sqrt(sum((Y_real - Y_pred)^2)/(length(Y_real) - 2)))
}
SyxPerc <- function (Y_real, Y_pred){
  return (Syx(Y_real, Y_pred)/mean(Y_real) * 100 )
}
```

a - RMSE:

```
rmse rf: 0.1604969
rmse rn: 0.1116153
rmse svm: 0.1987083
rmse spurr: 0.1221683
```

b - Métrica R2:

r2 rf: 0.8052392
r2 rn: 0.9306231
r2 svm: 0.6448999
r2 spurr: 0.9250578

c - Métrica Syx:

Syx rf: 0.1691786
Syx rn: 0.1176529
Syx svm: 0.2094569
Syx spurr: 0.1287766

d - Métrica Syx percentual:

Syx percentual rf: 12.92659
Syx percentual rn: 8.784697
Syx percentual svm: 16.01954
Syx percentual spurr: 9.578796

7) Escolha do melhor modelo.

Considerando todas as métricas calculadas o modelo NNET (Redes Neurais) obteve o melhor resultado em todas elas.

RMSE:

rmse rn: 0.1116153

Métrica R2:

r2 rn: 0.9306231

Métrica Syx:

Syx rn: 0.1176529

Métrica Syx percentual:

Syx percentual rn: 8.784697

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:

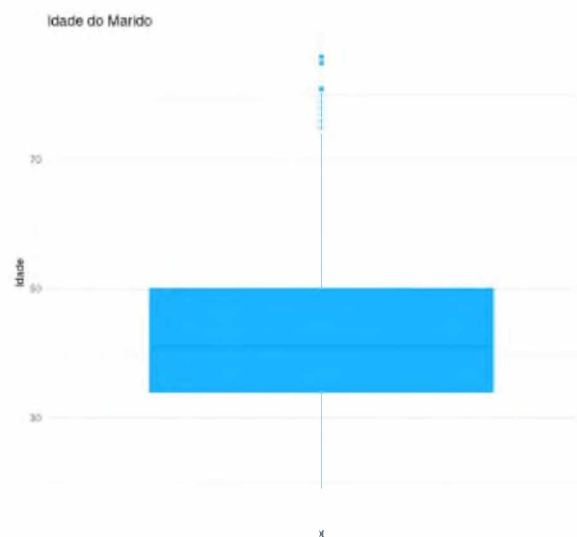
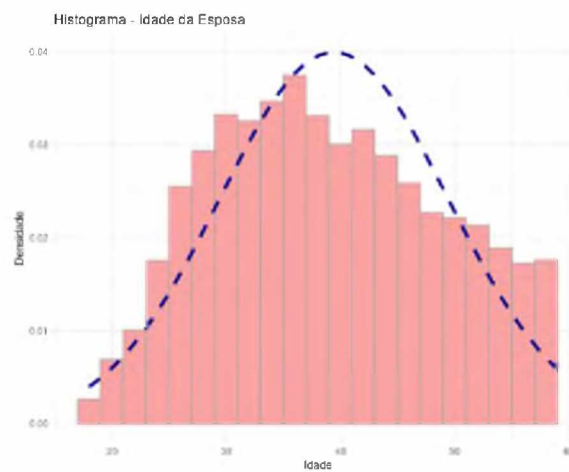
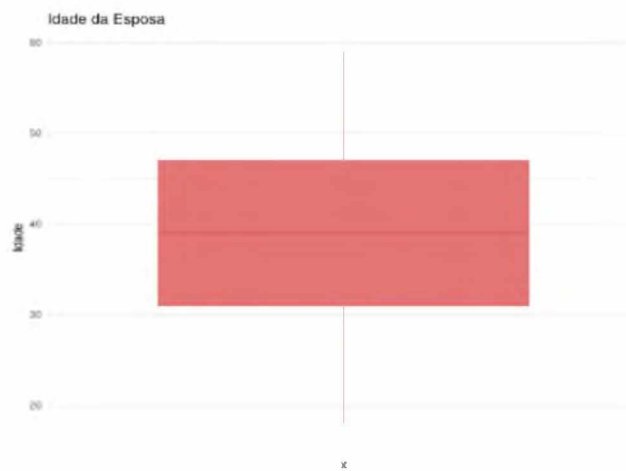
Você deve fazer os testes necessários (e mostra-los no documento pdf) para saber se você deve usar o unpaired test (paramétrico) ou o teste U de Mann-Whitney (não paramétrico), justifique sua resposta sobre a escolha.

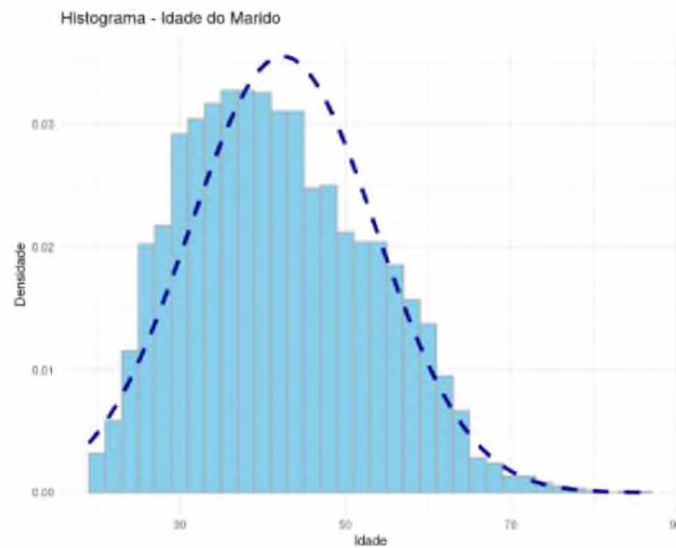
Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

B – RESOLUÇÃO

1 - GRÁFICOS E TABELAS

1.a) Elaborar os gráficos box-plot e histograma das variáveis "age"(idade da esposa) e "husage" (idade do marido)





1.b) Criar tabelas de frequência para as variáveis "age" e "husage"

Tabela de Frequência da Esposa:

Class limits f rf rf(%) cf cf(%)

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

Tabela de Frequência do Marido:

Class limits f rf rf(%) cf cf(%)

[18.81,23.671)	102	0.02	1.81	102	1.81
[23.671,28.531)	466	0.08	8.27	568	10.08
[28.531,33.392)	809	0.14	14.36	1377	24.44
[33.392,38.253)	895	0.16	15.89	2272	40.33
[38.253,43.114)	917	0.16	16.28	3189	56.60
[43.114,47.974)	629	0.11	11.16	3818	67.77

```
[47.974,52.835) 649 0.12 11.52 4467 79.29
[52.835,57.696) 541 0.10 9.60 5008 88.89
[57.696,62.556) 394 0.07 6.99 5402 95.88
[62.556,67.417) 152 0.03 2.70 5554 98.58
[67.417,72.278) 51 0.01 0.91 5605 99.49
[72.278,77.139) 21 0.00 0.37 5626 99.86
[77.139,81.999) 6 0.00 0.11 5632 99.96
[81.999,86.86) 2 0.00 0.04 5634 100.00
```

Seção 2: Medidas de posição e dispersão

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

```
[1] "Média da Idade da Esposa:"
```

```
> media_age
```

```
[1] 39.42758
```

```
[1] "Mediana da Idade da Esposa:"
```

```
> mediana_age
```

```
[1] 39
```

```
[1] "Moda da Idade da Esposa:"
```

```
> moda_age
```

```
[1] 37
```

```
[1] "Média da Idade do Marido:"
```

```
> media_husage
```

```
[1] 42.45296
```

```
[1] "Mediana da Idade do Marido:"
```

```
> mediana_husage
```

```
[1] 41
```

```
[1] "Moda da Idade do Marido:"
```

```
> moda_husage
```

```
[1] 44
```

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

```
[1] "Variância da Idade da Esposa:"
```

```
> variancia_age
```

```
[1] 99.75234
```

```
[1] "Desvio Padrão da Idade da Esposa:"
```

```
> desvio_padrao_age
```

```
[1] 9.98761
```

```
[1] "Coeficiente de Variação para a Idade da Esposa:"
```

```
> coeficiente_variacao_age
```

```
[1] 25.33153
```

```
[1] "Variância da Idade do Marido:"
```

```
> variancia_husage
```

```
[1] 126.0717
```

```
[1] "Desvio Padrão da Idade do Marido:"
```

```
> desvio_padrao_husage
```

```
[1] 11.22817
```

```
[1] "Coeficiente de Variação para a Idade do Marido:"
```

```
> coeficiente_variacao_husage
```

```
[1] 26.44849
```

Seção 3: Testes paramétricos ou não paramétricos

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.

Data Frame Head

	grupo	idade
1	Esposa	43
2	Esposa	26
3	Esposa	49
4	Esposa	35
5	Esposa	43
6	Esposa	58

summary()

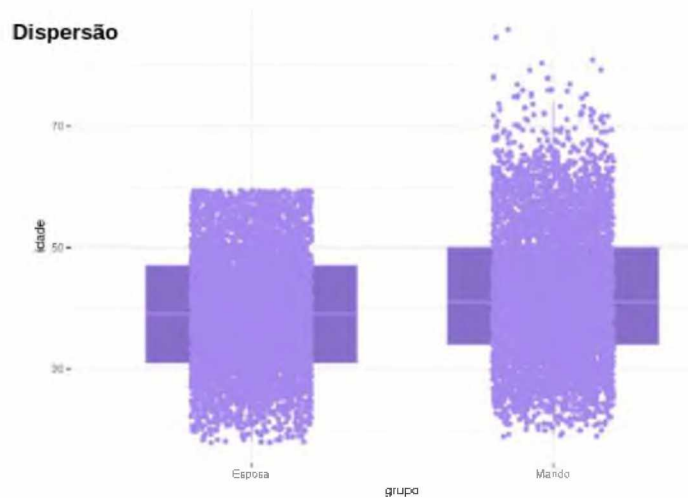
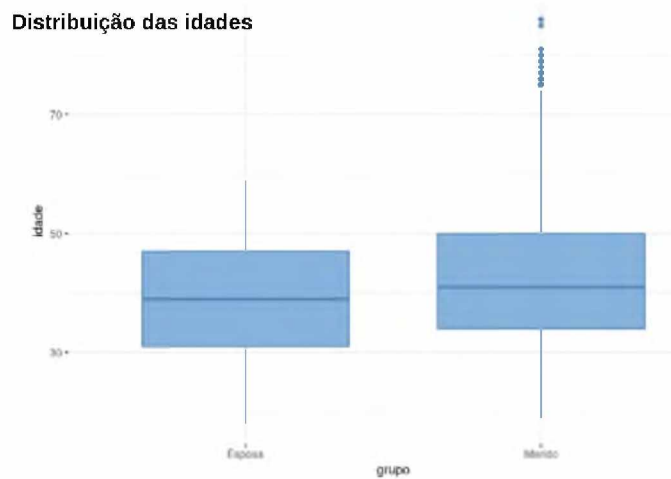
grupo	idade
Length:11268	Min. :18.00
Class :character	1st Qu.: 33.00
Mode :character	Median : 40.00
	Mean : 40.94
	3rd Qu.: 49.00
	Max. : 86.00

lillie.test - Lilliefors (Kolmogorov-Smirnov) normality test

data: idade_emp\$idade

D = 0.060083, p-value < 0.00000000000000022

Como p-value é menor do que 0.05, o teste que será adequado é o: Mann-Whitney U



Primeiro teste: Se a idade mediana dos maridos é igual à das esposas

wilcox.test

Wilcoxon rank sum test with continuity correction

data: idade by grupo

W = 13619912, p-value < 0.00000000000000022 (< 0.05)

alternative hypothesis: true location shift is not equal to 0

95 percent confidence interval:

-3.000024 -2.000033

sample estimates:

difference in location

-2.999966

Rejeitamos a hipótese nula pois a idade mediana dos maridos não é igual à das esposas.

Segundo teste: se a idade mediana dos maridos é menor que a das esposas

wilcox.test

Wilcoxon rank sum test with continuity correction

data: idade by grupo

W = 13619912, p-value < 0.00000000000000022 (< 0.05)

alternative hypothesis: true location shift is less than 0

95 percent confidence interval:

-Inf -2.000046

sample estimates:

difference in location

-2.999966

Rejeitamos a hipótese nula pois a idade mediana dos maridos não é menor que a das esposas.

Terceiro teste: se a idade mediana dos maridos é maior que a das esposas

wilcox.test

Wilcoxon rank sum test with continuity correction

data: idade by grupo

W = 13619912, p-value = 1 (> 0.05)

alternative hypothesis: true location shift is greater than 0

95 percent confidence interval:

-3.000034 Inf

sample estimates:

difference in location

-2.999966

Aceitamos a hipótese nula pois a idade mediana dos maridos é maior que a das esposas.

APÊNDICE 5 - ESTATÍSTICA APLICADA II

A – ENUNCIADO

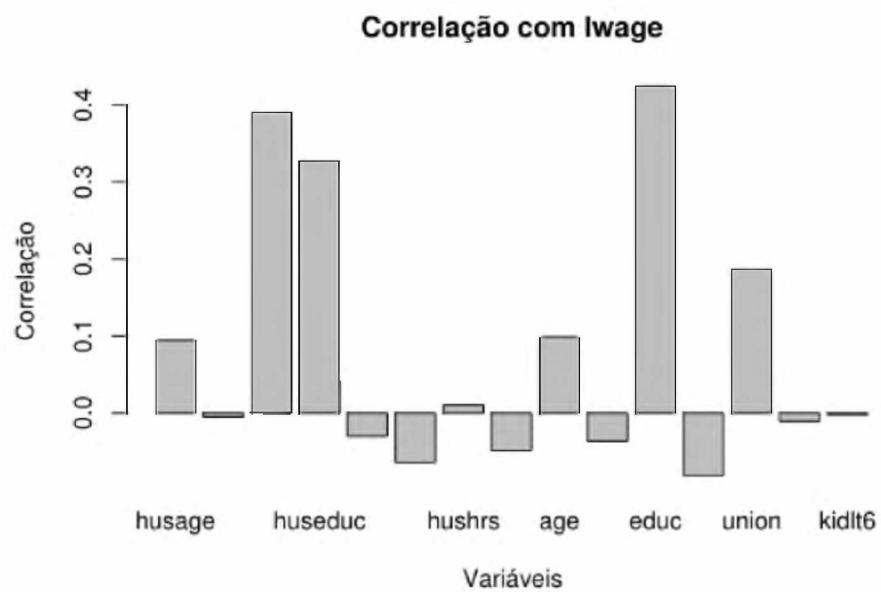
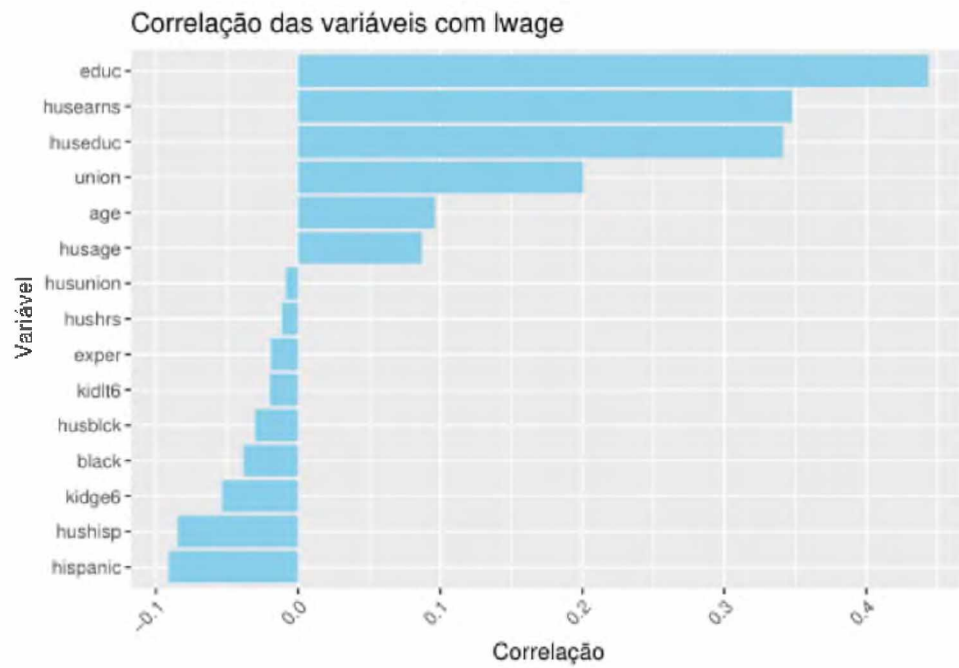
Regressões Ridge, Lasso e ElasticNet

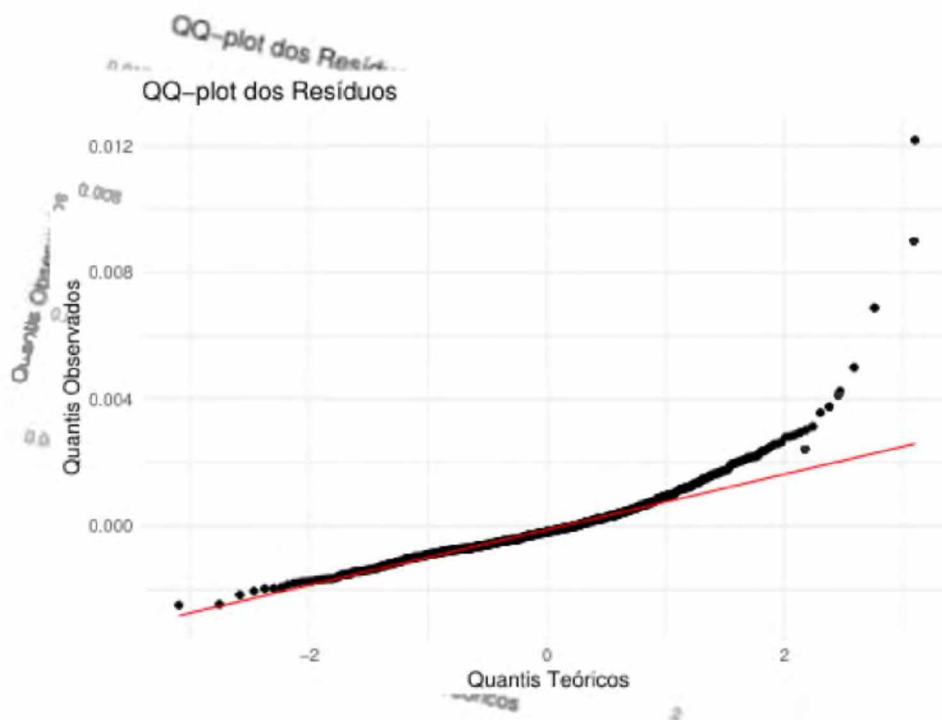
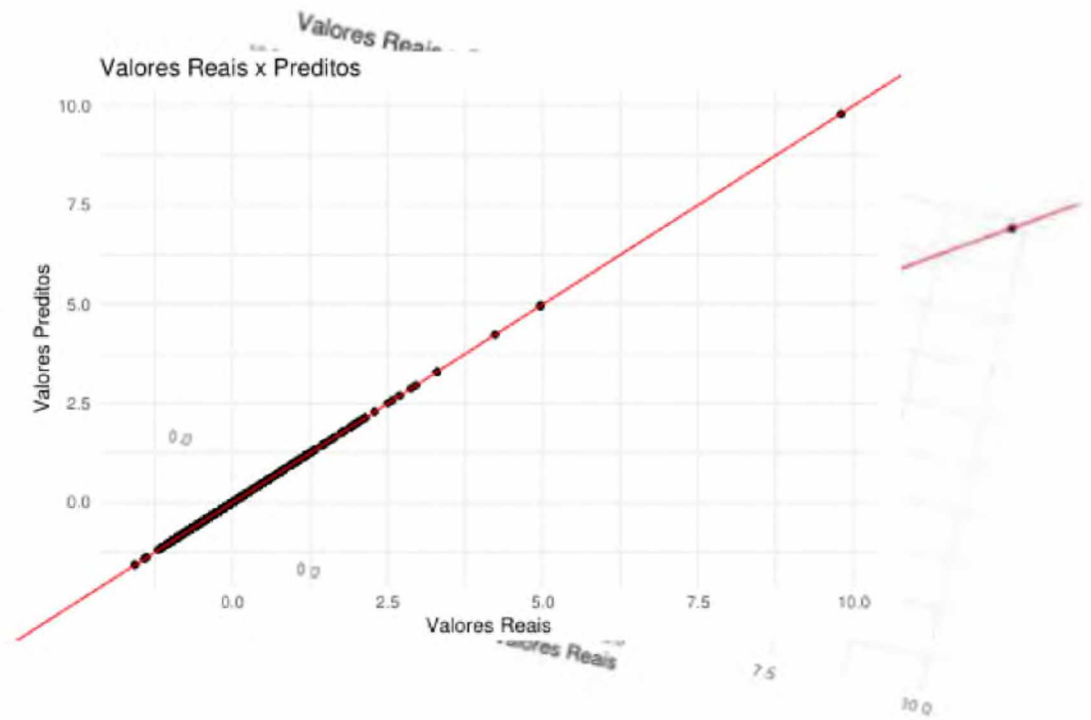
(100 pontos) Fazer as regressões Ridge, Lasso e ElasticNet com a variável dependente “lwage” (salário-hora da esposa em logaritmo neperiano) e todas as demais variáveis da base de dados são variáveis explicativas (todas essas variáveis tentam explicar o salário-hora da esposa). No pdf você deve colocar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

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

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

B – RESOLUÇÃO





Treinamos o modelo

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

O melhor parametro alpha escolhido é:

```
elastic_reg$bestTune
##      alpha      lambda
## 5  0.3660426  0.01991327
```

Agora fazemos as predicoes e avaliamos a performance

```
predictions_train <- predict(elastic_reg, x_train_elastic)
```

Calculamos o R^2 dos valores verdadeiros e preditos conforme a seguinte função:

```
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))
  # As metricas de performance do modelo:
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}
```

Observamos as métricas de performance na base de treinamento

```
rmse_r2_elastic_trein <- eval_results(y_train_elastic, predictions_train, train_elastic)
rmse_r2_elastic_trein
```

```
##      RMSE      Rsquare
## 1  0.8474519  0.2814763
```

Fazemos as predições na base de teste

```
predictions_test_elastic <- predict(elastic_reg, x_test_elastic)
```

Observamos as métricas de performance na base de teste:

```
rmse_r2_elastic_test <- eval_results(y_test_elastic, predictions_test_elastic,
test_elastic)
rmse_r2_elastic_test
```

```
##      RMSE      Rsquare
## 1  0.867134  0.2360335
```

Construimos uma matriz de dados para a predição

```
our_pred_elastic = data.frame(husage = husage,
  husearns = husearns,
  huseduc = huseduc,
  hushrs = hushrs,
```

```

age = age,
educ = educ,
exper = exper,
husunion = husunion,
husblk = husblk,
hushisp = hushisp,
kidge6 = kidge6,
#earn = earns,
hispanic = hispanic,
black = black,
union = union,
kidlt6 = kidlt6)

# Fazemos a predição com base nos parâmetros que selecionamos
predict_our_elastic <- predict(elastic_reg,our_pred_elastic)
predict_our_elastic
## [1] -0.1773049

# Revertemos o resultado q esta padronizado para o real
wage_pred_elastic=(predict_our_elastic*
                    pre_proc_val[["std"]][["lwage"]])+
                    pre_proc_val[["mean"]][["lwage"]]
wage_pred_elastic
## [1] 9.236578

# Então o salário-hora da esposa predito com base
# nas características informadas e US$ 9.24

# Criamos o intervalo de confiança:
n <- nrow(train_elastic)
m <- wage_pred_elastic
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_elastic <- m + (qnorm(0.025))*dam
CIupr_elastic <- m - (qnorm(0.025))*dam

# Os valores mínimo e máximo são:
CIlwr_elastic
## [1] 8.976024
CIupr_elastic
## [1] 9.497133

# O modelo preve que o salário-hora da esposa é de 9.24 e pode variar entre 8.98 e 9.50
# Resultados das regressões
resultados_rmse <- data.frame(
  Modelo = c("Ridge", "Lasso", "Elastic Net"),
  RMSE=c(rmse_r2_ridge_trein$RMSE,
  rmse_r2_lasso_trein$RMSE,
  rmse_r2_elastic_trein$RMSE)
)
resultados_r2 <- data.frame(Modelo = c("Ridge", "Lasso", "Elastic Net"),
  R_squared = c(rmse_r2_ridge_trein$Rsquare,
  rmse_r2_lasso_trein$Rsquare,
  rmse_r2_elastic_trein$Rsquare)
)
print(resultados_rmse)
## Modelo RMSE
## 1 Ridge 0.001177862
## 2 Lasso 0.848609851
## 3 Elastic Net 0.847451884

print(resultados_r2)
## Modelo R_squared
## 1 Ridge 0.9999986
## 2 Lasso 0.2795114

```

3 Elastic Net 0.2814763

CONCLUSÃO DOS RESULTADOS :

Observamos que o modelo que gerou menos resíduos e teve uma melhor métrica de avaliação dos resultados foi o MODELO DE REGRESSÃO RIDGE.

E que o valor do salário hora PREDITO da esposa é US\$10.26 e

pode variar entre US\$10.01 e US\$10.53

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

```
import random

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

random.shuffle(pre_padroes)
print(pre_padroes)

import pandas as pd
dados = pd.DataFrame(pre_padroes, columns=['Frase', 'Idioma'])
dados

import re #REGEX LIBRARY
import numpy as np

def tamanhoMedioFrases(texto):
    palavras = re.findall(r'\b\w+\b', texto) # Palavras dentro da frase digitada

    # primeiras_palavras = palavras[:5]
    tamanhos = [len(s) for s in palavras]
    return (sum(tamanhos) / len(tamanhos)) if len(tamanhos) > 0 else 0 # Evita divisão
por zero

def contaCaracteristicas(texto):
    texto = texto.lower()
    vogais = "eiouáéíóúãõâêîôûâèìòùãëïö"
    consoantes = "bcdfghjklmnpqrstvwxyz"
    letras_espanhol = '¿¡!ñü'
    letras_portugues = 'çáãêôû'
    letras_ingles = "wkyh"

    num_vogais = sum(1 for char in texto if char in vogais)
    num_consoantes = sum(1 for char in texto if char in consoantes)
    num_letras_espanhol = sum(1 for char in texto if char in letras_espanhol)
    num_letras_portugues = sum(1 for char in texto if char in letras_portugues)
    num_letras_ingles = sum(1 for char in texto if char in letras_ingles)

palavras_comuns_ingles = {'the', 'be', 'to', 'of', 'and', 'a', 'in', 'that', 'have', 'i',
'gh', 'th', 'has', 'long', 'little', 'good', 'new', 'same', 'right', 'different', 'old',
'great', 'small', 'big', 'own', 'last', 'left', 'large', 'important', 'enough', 'second',
'first', 'well', 'next', 'live', 'high', 'following', 'used', 'even', 'is', 'was', 'are',
'be', 'had', 'were', 'can', 'said', 'do', 'will', 'would', 'has', 'could', 'make', 'been',
'did', 'find', 'use', 'know', 'get', 'go', 'write', 'look', 'think', 'came', 'come',
'must', 'does', 'put', 'went', 'tell', 'say', 'should', 'give', 'read', 'might', 'saw',
'asked', 'don't', 'going', 'want', 'keep', 'took', 'began', 'got', 'need', 'let',
'being', 'see', 'made', 'may', 'take', 'found', 'set', 'looked', 'study', 'called', 'you',
'it', 'he', 'they', 'this', 'I', 'what', 'all', 'we', 'which', 'each', 'them', 'she',
'some', 'these', 'who', 'me', 'another', 'us', 'something', 'both', 'those', 'the', 'a',
'his', 'an', 'your', 'their', 'other', 'more', 'him', 'its', 'my', 'much', 'our', 'any',
'such', 'every', 'many', 'as', 'of', 'to', 'in', 'for', 'on', 'with', 'at', 'from',
'by', 'about', 'into', 'like', 'after', 'through', 'before', 'under', 'along', 'until',
'without', 'between', 'and', 'or', 'but', 'if', 'than', 'because', 'while', 'when', 'how',
'up', 'out', 'then', 'so', 'no', 'not', 'now', 'over', 'down', 'only', 'very', 'just',
'where', 'most', 'too', 'also', 'around', 'here', 'why', 'away', 'again', 'off', 'still',
'never', 'below', 'always', 'often', 'together', 'once', 'sometimes', 'above',
'almost', 'far', 'back', 'one', 'two', 'three', 'few', 'four'}

palavras_comuns_espanhol = {'i', '¿', 'll', 'ñ', 'aire', 'su', 'eres', 'es', 'uste',
'una', 'tú', 'de', 'la', 'que', 'el', 'en', 'em', 'y', 'a', 'los', 'hay', 'sin', 'del',
'se', 'oi', 'tus', 'solo', 'lla', 'me', 'mi', 'ast', 'bie', 'ciu', 'cosa', 'ñol',
```

```
'eje', 'tán', 'fue', 'ha', 'hacer', 'hecho', 'hizo', 'haz', 'hoy', 'mej', 'mien',
'ción', 'yec', 'quie', 'gún', 'bién', 'tiem', 'pero', 'acción', 'acuerdo', 'además', 'agua', 'ahora',
'año', 'aunque', 'bajo', 'bien', 'zca', 'acción', 'acuerdo', 'además', 'agua', 'ahora', 'año', 'aunqu
e', 'bajo', 'bien', 'cabeza', 'calidad', 'cambio', 'centro', 'cosa', 'cuando', 'cuenta', 'cuerpo', 'D
ecir', 'derecho', 'desarrollo', 'dice', 'dicho', 'donde', 'español', 'están', 'estaba', 'Familia', '
Fin', 'final', 'Frente', 'ron', 'general', 'gente', 'había', 'hemos', 'hacer', 'hecho', 'hizo', 'haci
a', 'hasta', 'historia', 'hombre', 'incluso', 'lado', 'ley', 'llegar', 'lograr', 'luego', 'madre', 'm
al', 'manera', 'mano', 'mayoría', 'medio', 'mejo', 'ministro', 'mismo', 'mujer', 'muy', 'mucho', 'nacio
nal', 'nada', 'nadie', 'nivel', 'nuevo', 'nunca', 'padre', 'palabras', 'parece', 'partido', 'part
ir', 'pasado', 'pero', 'personochenas', 'poder', 'programa', 'propio', 'proyecto', 'pueden', 'pues',
'punto', 'quiere', 'realidad', 'relación', 'semana', 'sentido', 'ser', 'sea', 'será', 'sistema', 's
ituación', 'social', 'sociedad', 'también', 'tiempo', 'tierra', 'tipo', 'todo', 'trabajo', 'uno', 'v
a', 'ver', 'vida', 'crees', 'vacaciones'}
```

```
palavras_comuns_portugues = {'do', 'da', 'em', 'um', 'para', 'nh', 'lh', 'qua', 'di',
'du', 'je', 'último', 'próprio', 'haver', 'fazer', 'dar', 'ficar', 'poder', 'ver', 'coisa', 'casa', 't
empo', 'ano', 'vez', 'homem', 'senhor', 'senhora', 'moço', 'moça', 'não', 'mais', 'muito', 'já', 'quan
do', 'mesmo', 'primeiro', 'cem', 'mil', 'com', 'até', 'ou', 'também', 'assim', 'como', 'porque', 'eu',
'você', 'ele', 'esse', 'isso', 'sua', 'Ai!', 'Ah!', 'Au!', 'Ui!', 'Hum!'}
```

```
palavras = re.findall(r'\b\w+\b', texto)
num_palavras_ingles = sum(1 for palavra in palavras if palavra in palavras_comuns_ingles)
num_palavras_espanhol = sum(1 for palavra in palavras if palavra in
palavras_comuns_espanhol)
num_palavras_portugues = sum(1 for palavra in palavras if palavra in
palavras_comuns_portugues)
return num_vogais, num_consoantes, num_letras_espanhol, num_letras_portugues,
num_letras_ingles, num_palavras_ingles, num_palavras_espanhol, num_palavras_portugues
```

```
palavras = re.findall(r'\b\w+\b', texto)
num_palavras_ingles = sum(1 for palavra in palavras if palavra in palavras_comuns_ingles)
num_palavras_espanhol = sum(1 for palavra in palavras if palavra in
palavras_comuns_espanhol)
num_palavras_portugues = sum(1 for palavra in palavras if palavra in
palavras_comuns_portugues)
return num_vogais, num_consoantes, num_letras_espanhol, num_letras_portugues,
num_letras_ingles, num_palavras_ingles, num_palavras_espanhol, num_palavras_portugues
```

```
def extraiCaracteristicas(frase):
```

```
    texto = frase[0]
    pattern_regex = re.compile('[^\w+]', re.UNICODE)
    texto = re.sub(pattern_regex, ' ', texto)
```

```
    num_vogais, num_consoantes, num_letras_espanhol, num_letras_portugues,
    num_letras_ingles, num_palavras_ingles, num_palavras_espanhol,
    num_palavras_portugues = contaCaracteristicas(texto)
```

```
    caracteristica1 = tamanhoMedioFrases(texto)
```

```
    padrao=[caracteristica1, num_vogais, num_consoantes, num_letras_espanhol,
            num_letras_portugues, num_letras_ingles, num_palavras_ingles,
            num_palavras_espanhol, num_palavras_portugues, 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 caracteristicas
padroes = geraPadroes(pre_padroes)
```

```
# apenas para visualizacao
print(padroes)
```

```
dados = pd.DataFrame(padroes)
dados[0] = np.where(dados[9] == 'português', ((dados[0])+1)/2, dados[0])
```

```
dados.head(60)
```

```
[[5.8, 13, 15, 2, 0, 1, 0, 1, 0, 'espanhol'], [5.0, 9, 14, 1, 0, 0, 0, 0, 0,
'espanhol'], [4.2, 7, 10, 1, 1, 0, 1, 0, 0, 'português'], [4.4, 8, 11, 2, 1, 0,
0, 1, 0, 'espanhol'], [3.8333333333333335, 11, 12, 2, 0, 1, 4, 0, 0, 'inglês'],
[3.3333333333333335, 5, 14, 2, 0, 2, 3, 0, 0, 'inglês'], [3.2, 4, 8, 0, 0, 0, 1,
1, 1, 'espanhol'], [4.2, 9, 12, 5, 0, 1, 3, 0, 0, 'inglês'], [4.333333333333333,
5, 7, 0, 1, 1, 0, 0, 0, 'espanhol'], [3.6666666666666665, 8, 14, 2, 0, 1, 3, 0,
0, 'inglês'], [3.142857142857143, 7, 11, 1, 0, 0, 1, 2, 0, 'espanhol'], [3.0, 4,
7, 3, 0, 2, 3, 0, 0, 'inglês'] ... continua
```

Treinando o modelo com SVM

Acurácia nos dados de treinamento: **86.96%**

```
[[19 0 3]
 [ 2 21 0]
 [ 4 0 20]]
```

	precision	recall	f1-score	support
espanhol	0.76	0.86	0.81	22
inglês	1.00	0.91	0.95	23
português	0.87	0.83	0.85	24
accuracy			0.87	69
macro avg	0.88	0.87	0.87	69
weighted avg	0.88	0.87	0.87	69

Acurácia nos dados de teste: 86.96%

métricas mais confiáveis

```
[[6 0 2]
 [0 6 1]
 [0 0 8]]
```

	precision	recall	f1-score	support
espanhol	1.00	0.75	0.86	8
inglês	1.00	0.86	0.92	7
português	0.73	1.00	0.84	8
accuracy			0.87	23
macro avg	0.91	0.87	0.87	23
weighted avg	0.91	0.87	0.87	23

```
texto = input('Digite o texto a ser classificado pelo Modelo: ')
texto_vetor = [texto, '0']
```

```
print(texto[2])
```

```
# Extraímos as características do texto de exemplo
caracteristicas_texto = extraiCaracteristicas(texto_vetor)
```

```
print(caracteristicas_texto)
```

```
# Removemos a última característica que é o rótulo '0'
caracteristicas_texto = caracteristicas_texto[:-1]
print(caracteristicas_texto)
```

```
# Usamos o modelo treinado para prever o idioma do texto
idioma_predito = modelo.predict([caracteristicas_texto])
print("O texto foi classificado como:", idioma_predito[0])
```


Digite o texto a ser classificado pelo Modelo: celebrating the birthday of a person

```
[5.166666666666667, 8, 20, 2, 0, 3, 3, 1, 0, '0']
[5.166666666666667, 8, 20, 2, 0, 3, 3, 1, 0]
```

O texto foi classificado como: **inglês**

Nos parece que existe prevalência de frases reconhecidas como em Português devido ao fato do tamanho da frase ser uma característica. Quando frases menor em espanhol são digitadas, são reconhecidas como Espanhol corretamente, mas se adicionar termos/palavras em espanhol, aumentando o tamanho do texto, é reconhecido como Português. A característica TAMANHO do texto parece que tem peso maior no modelo apresentado.

2 Melhore uma base de dados ruim

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.data'
```

```
# Nomes das colunas, conforme descrito na página do dataset
column_names = ["word_freq_make", "word_freq_address", "word_freq_all",
"word_freq_3d", "word_freq_our",
"word_freq_over", "word_freq_remove",
"word_freq_internet", "word_freq_order", "word_freq_mail",
"word_freq_receive", "word_freq_will",
"word_freq_people", "word_freq_report", "word_freq_addresses",
"word_freq_free", "word_freq_business",
"word_freq_email", "word_freq_you", "word_freq_credit",
"word_freq_your", "word_freq_font",
"word_freq_000", "word_freq_money", "word_freq_hp",
"word_freq_hpl", "word_freq_george",
"word_freq_650", "word_freq_lab", "word_freq_labs",
"word_freq_telnet", "word_freq_857",
"word_freq_data", "word_freq_415", "word_freq_85",
"word_freq_technology", "word_freq_1999",
"word_freq_parts", "word_freq_pm", "word_freq_direct",
"word_freq_cs", "word_freq_meeting",
"word_freq_original", "word_freq_project", "word_freq_re",
"word_freq_edu", "word_freq_table",
"word_freq_conference", "char_freq_;", "char_freq_(",
"char_freq_[", "char_freq_!", "char_freq_$",
"char_freq_#", "capital_run_length_average",
"capital_run_length_longest",
"capital_run_length_total", "class"]
```

```
# Carregar o dataset diretamente da URL
data = pd.read_csv(url, header=None, names=column_names)
```

```
from ucimlrepo import fetch_ucirepo
```

```
## Obtem o dataset da UCI - ID 94
spambase = fetch_ucirepo(id=94)
# Data (as pandas dataframes)
X = spambase.data.features #CARACTERÍSTICAS
y = spambase.data.targets #CLASSE
```

```
## metadata
print(spambase.metadata)
```

```
## variable information
print(spambase.variables)
```

	name	role	type	demographic \
0	word_freq_make	Feature	Continuous	None
1	word_freq_address	Feature	Continuous	None

2	word_freq_all	Feature	Continuous	None
3	word_freq_3d	Feature	Continuous	None

... continua

```
caracteristicas = pd.DataFrame(X)
classes = pd.DataFrame(y)
```

```
dadosGeral: pd.DataFrame = pd.concat([caracteristicas, classes], axis=1)
dadosGeral.head(20)
```

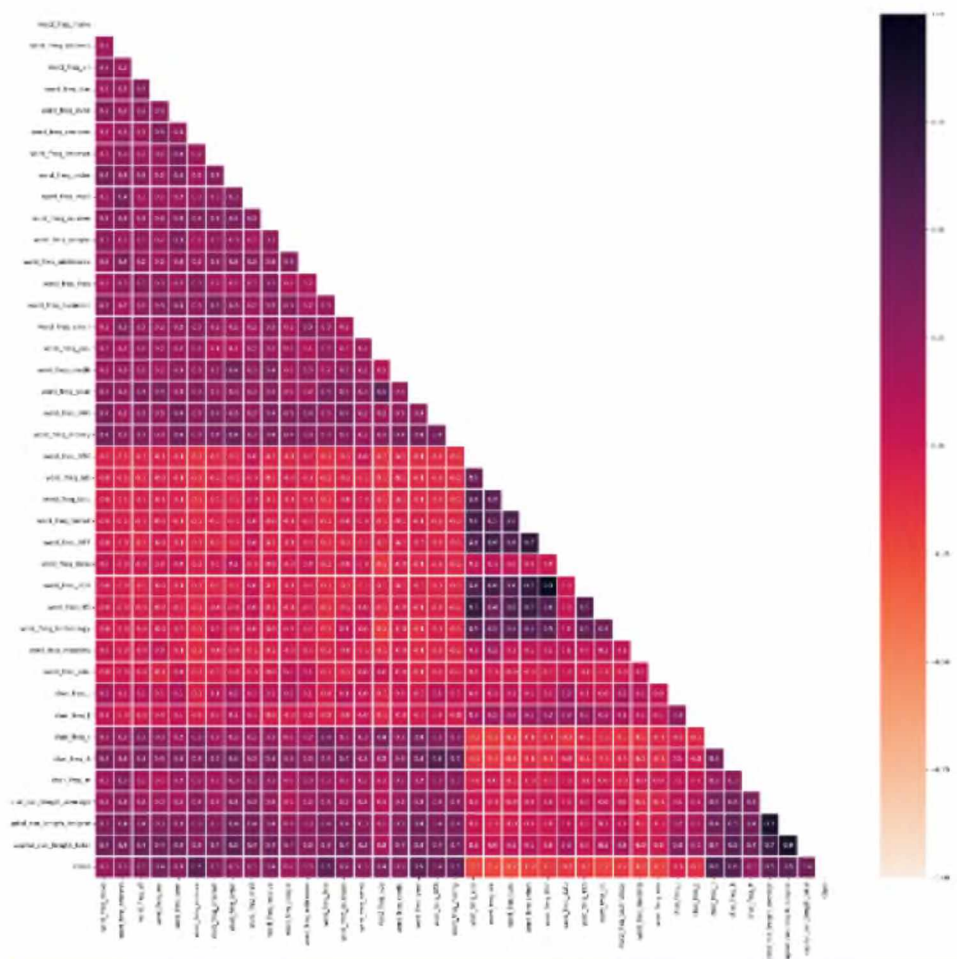
```
dadosGeralAltaCorrelacao = dadosGeral.
drop(['word_freq_3d', 'word_freq_will', 'word_freq_report', 'word_freq_font', 'word_freq_pm', '
word_freq_direct', 'word_freq_cs', 'word_freq_original', 'word_freq_hp', 'word_freq_hpl', 'word
_freq_george',
'char_freq_', 'word_freq_parts', 'word_freq_project', 'word_freq_re', 'word_freq_table',
'word_freq_conference'], axis=1)
```

```
unique_classes = dadosGeralAltaCorrelacao['Class'].unique()
print("Classes:", unique_classes)
```

```
matriz_correlacao = dadosGeralAltaCorrelacao.corr(method='spearman')
```

```
plt.figure(figsize=(26, 26))
mask = np.triu(np.ones_like(matriz_correlacao, dtype=float))
sns.heatmap(matriz_correlacao, mask=mask, vmin=-1, vmax=1, annot=True, cmap='rocket_r',
fmt="0.1f", linewidth=.9)
```

```
plt.title('Matriz de Correlação entre Características e Classes - Dados Originais')
plt.show()
```



TRATAMENTOS

```
print(dadosGeralAltaCorrelacao.isna().sum())
print(dadosGeralAltaCorrelacao.isnull().sum())
dadosGeralAltaCorrelacao.head(10)

# DADOS AUSENTES
results = list()

X = dadosGeral.iloc[:, :-1].values # Características
y = dadosGeral.iloc[:, -1].values # Classes

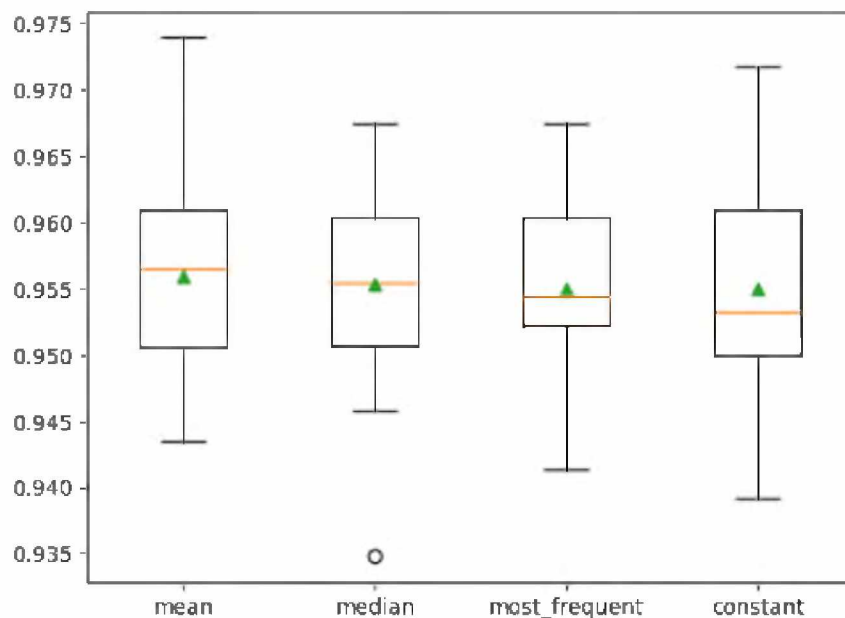
strategies = ['mean', 'median', 'most_frequent', 'constant']

for strat in strategies:
    pipeline=Pipeline(steps=[('i',SimpleImputer(strategy=strat)),
                              ('m',RandomForestClassifier())])
    crossval = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,random_state=42)
    scores = cross_val_score(pipeline, X, y, scoring='accuracy', cv=crossval,n_jobs=-1)
    results.append(scores)

    print('> %s %.3f (%.3f)' % (strat, np.mean(scores), np.std(scores)))

plt.boxplot(results, labels=strategies, showmeans=True)
plt.show()

> mean 0.956 (0.007)
> median 0.955 (0.007)
> most_frequent 0.955 (0.006)
> constant 0.955 (0.008)
```



ESTUDO NORMALIZAÇÃO DOS DADOS GERAL

```
min_max_scaler = MinMaxScaler()
std_scaler = StandardScaler()

caracteristicas = dadosGeral.iloc[:, :-1] # Características
classes = dadosGeral.iloc[:, -1] # Classes
```

```

data_normalized = min_max_scaler.fit_transform(dadosGeral)
df_data_normalized = pd.DataFrame(data_normalized)
data_standardized = std_scaler.fit_transform(dadosGeral)
df_data_standardized = pd.DataFrame(data_standardized)

plt.figure(figsize = (20,115))
idx = 0

for idx, col in enumerate(list(caracteristicas)):
    plt.subplot(29, 2, idx+1)
    sns.kdeplot(dadosGeral[col], color = 'g', label = f"Unscaled: {col}")
    sns.kdeplot(df_data_normalized[idx], color = 'b', label = f"Normalized: {col}" )
    sns.kdeplot(df_data_standardized[idx], color= 'r', label = f"Standardized: {col}" )

    plt.legend(loc = 'upper right')

plt.show()

## Acurácia na Base de Teste antes do Balanceamento
X_train, X_test, y_train, y_test=train_test_split(X,y, stratify=classes, test_size=0.25,
random_state=42)

# Treinar o classificador SVM
svm_classifier = SVC(kernel='rbf')

# Ajustar o modelo aos dados de treino
svm_classifier.fit(X_train, y_train)

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

# Calcular a acurácia
initial_accuracy = accuracy_score(y_test, y_pred)
print("Acurácia na Base de Teste antes do Balanceamento:", initial_accuracy)

# Exibir a matriz de confusão
initial_conf_matrix = confusion_matrix(y_test, y_pred)

print("Matriz de Confusão - Dados de Teste:")
print(initial_conf_matrix)
print(classification_report(y_test, y_pred))

Acurácia na Base de Teste antes do Balanceamento: 0.7019982623805386
Matriz de Confusão - Dados de Teste:
[[587 110]
 [233 221]]

```

	precision	recall	f1-score	support
0	0.72	0.84	0.77	697
1	0.67	0.49	0.56	454
accuracy			0.70	1151
macro avg	0.69	0.66	0.67	1151
weighted avg	0.70	0.70	0.69	1151

```

# BALANCEAR os dados de treino com SMOTE
smote = SMOTE(random_state = 42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)

# Treinar o classificador SVM com os dados BALANCEADOS
svm_classifier_balanced = SVC(kernel='rbf')
svm_classifier_balanced.fit(X_train_balanced, y_train_balanced)

# Fazer previsões no conjunto de teste
y_pred_balanced = svm_classifier_balanced.predict(X_test)

# Calcular a acurácia

```

```
balanced_accuracy = accuracy_score(y_test, y_pred_balanced)
print("Acurácia na Base de Teste após o Balanceamento:", balanced_accuracy)
```

```
# Exibir a matriz de confusão
```

```
balanced_conf_matrix = confusion_matrix(y_test, y_pred_balanced)
```

```
print("Matriz de Confusão - Dados de Teste após o Balanceamento:")
```

```
print(balanced_conf_matrix)
```

```
print(classification_report(y_test, y_pred_balanced))
```

Acurácia na Base de Teste após o Balanceamento: 0.6776715899218071

Matriz de Confusão - Dados de Teste após o Balanceamento:

```
[[483 214]
```

```
[157 297]]
```

	precision	recall	f1-score	support
0	0.75	0.69	0.72	697
1	0.58	0.65	0.62	454
accuracy			0.68	1151
macro avg	0.67	0.67	0.67	1151
weighted avg	0.69	0.68	0.68	1151

```
# Normalização Min-Max dos dados
```

```
scaler = MinMaxScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train_balanced)
```

```
X_test_scaled = scaler.transform(X_test)
```

```
# Seleção de características
```

```
selector = SelectKBest(f_classif, k=20)
```

```
X_train_selected = selector.fit_transform(X_train_scaled, y_train_balanced)
```

```
X_test_selected = selector.transform(X_test_scaled)
```

```
# Treinar o classificador SVM com os dados balanceados, normalizados e selecionados
```

```
svm_classifier_improved = SVC(kernel='rbf')
```

```
svm_classifier_improved.fit(X_train_selected, y_train_balanced)
```

```
# Fazer previsões no conjunto de teste
```

```
y_pred_improved = svm_classifier_improved.predict(X_test_selected)
```

```
# Calcular a acurácia
```

```
improved_accuracy = accuracy_score(y_test, y_pred_improved)
```

```
print("Acurácia na Base de Teste após Melhorias:", improved_accuracy)
```

```
# Exibir a matriz de confusão
```

```
improved_conf_matrix = confusion_matrix(y_test, y_pred_improved)
```

```
print("Matriz de Confusão - Dados de Teste após Melhorias:")
```

```
print(improved_conf_matrix)
```

```
print(classification_report(y_test, y_pred_improved))
```

Acurácia na Base de Teste após Melhorias: 0.9174630755864466

Matriz de Confusão - Dados de Teste após Melhorias:

```
[[658 39]
```

```
[ 56 398]]
```

	precision	recall	f1-score	support
0	0.92	0.94	0.93	697
1	0.91	0.88	0.89	454
accuracy			0.92	1151
macro avg	0.92	0.91	0.91	1151
weighted avg	0.92	0.92	0.92	1151

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

Leitura dos dados

```
setwd("/home/magnus/Downloads/ml-007/ufpr-ia-aplicada/ml-r-superv-naosuperv/data")
dados <- read.csv("6 - Veiculos - Dados.csv")
```

```
### Eliminar (índice)
dados$a <- NULL
```

```
## Cross-validation SVM
## Define o método de controle de treino como validação cruzada (cross-validation)
## com 10 divisões (folds)
```

```
## Treina o modelo SVM com kernel radial (svmRadial) usando os dados de treino
## e validação cruzada definida por ctrl
svm <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl)
svm
Support Vector Machines with Radial Basis Function Kernel
679 samples
18 predictor
4 classes: 'bus', 'opel', 'saab', 'van'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 611, 610, 611, 610, 611, 612, ...
Resampling results across tuning parameters:
C      Accuracy      Kappa
0.25   0.7171400      0.6231343
0.50   0.7510532      0.6681133
1.00   0.7731572      0.6974934
```

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

Confusion Matrix and Statistics

Reference Prediction	bus	opel	saab	van
bus	40	2	2	0
opel	0	18	14	0
saab	0	19	25	0
van	3	3	2	39

Overall Statistics
 Accuracy : 0.7305
 95% CI : (0.6565, 0.7962)
 No Information Rate : 0.2575
 P-Value [Acc > NIR] : < 2e-16

Kappa : 0.6409

Mcnemar's Test P-Value : 0.04705
 Statistics by Class:

	Class: bus	Class: opel	Class: saab	Class: van
Sensitivity	0.9302	0.4286	0.5814	1.0000
Specificity	0.9677	0.8880	0.8468	0.9375
Pos Pred Value	0.9091	0.5625	0.5682	0.8298
Neg Pred Value	0.9756	0.8222	0.8537	1.0000
Prevalence	0.2575	0.2515	0.2575	0.2335
Detection Rate	0.2395	0.1078	0.1497	0.2335
Detection Prevalence	0.2635	0.1916	0.2635	0.2814

```

Balanced Accuracy    0.9490        0.6583        0.7141        0.9688
## Vários C e sigma
## Define uma grade de parâmetros (tuneGrid) para buscar os melhores valores de C
## (penalidade) e sigma (parâmetro do kernel RBF) durante o treinamento do
## modelo SVM.
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(seedvarRandomState)
## Treinar o modelo SVM com kernel radial,
## usando: Validação cruzada (CROSS VALIDATION - Melhores parâmetros C) e sigma,
conforme definidos em tuneGrid.
svm <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl,tuneGrid=tuneGrid)
svm

```

Support Vector Machines with Radial Basis Function Kernel

679 samples

18 predictor

4 classes: 'bus', 'opel', 'saab', 'van'

No pre-processing

Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 611, 610, 611, 610, 611, 612, ...

Resampling results across tuning parameters:

C	sigma	Accuracy	Kappa
1	0.010	0.7273043	0.6366425
1	0.015	0.7405409	0.6542554
1	0.200	0.7481785	0.6642279
2	0.010	0.7538640	0.6719484
2	0.015	0.7701941	0.6936039
2	0.200	0.7687887	0.6916049
10	0.010	0.8072028	0.7429419
10	0.015	0.8086094	0.7447365
10	0.200	0.7877572	0.7168827
50	0.010	0.8188628	0.7584515
50	0.015	0.8277290	0.7702341
50	0.200	0.7832583	0.7108923
100	0.010	0.8292215	0.7722185
100	0.015	0.8217589	0.7622193
100	0.200	0.7832583	0.7108923

Accuracy was used to select the optimal model using the largest value.

The final values used for the model were sigma = 0.01 and C = 100.

Matriz de confusão

Confusion Matrix and Statistics

	Reference			
Prediction	bus	opel	saab	van
bus	42	1	1	0
opel	0	31	14	0
saab	0	10	28	0
van	1	0	0	39

Overall Statistics

Accuracy : 0.8383
95% CI : (0.7736, 0.8907)

No Information Rate : 0.2575

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.7844

Mcnemar's Test P-Value : NA

Statistics by Class:

	Class: bus	Class: opel	Class: saab	Class: van
Sensitivity	0.9767	0.7381	0.6512	1.0000

Specificity	0.9839	0.8880	0.9194	0.9922
Pos Pred Value	0.9545	0.6889	0.7368	0.9750
Neg Pred Value	0.9919	0.9098	0.8837	1.0000
Prevalence	0.2575	0.2515	0.2575	0.2335
Detection Rate	0.2515	0.1856	0.1677	0.2335
Detection Prevalence	0.2635	0.2695	0.2275	0.2395
Balanced Accuracy	0.9803	0.8130	0.7853	0.9961

Técnica	Parâmetro	Acurácia	Matriz de Confusão		
RNA – Hold-out	size=3 decay=0.1	69,00%	Prediction	neg	pos
			neg	85	35
			pos	15	18

Retir atributo a (ID)

num	pregOnt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
1	6	148	72	35	0	33.6	627	50	pos
2	1	85	66	29	0	26.6	351	31	neg
3	8	183	64	0	0	23.3	672	32	pos
4	1	89	66	23	94	28.1	167	21	neg
5	0	137	40	35	168	43.1	2288	33	pos
6	5	116	74	0	0	25.6	201	30	neg
7	3	78	50	32	88	31	248	26	pos
8	10	115	0	0	0	35.3	134	29	neg
9	2	197	70	45	543	30.5	158	53	pos

Treinamos o modelo com Hold-out

RNA HOLD-OUT

Neural Network

615 samples
8 predictor
2 classes: 'neg', 'pos'

No pre-processing

Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 615, 615, 615, 615, 615, 615, ...

Resampling results across tuning parameters:

size	decay	Accuracy	Kappa
1	0e+00	0.6585657	0.01194207
1	1e-04	0.6557814	0.10453271
1	1e-01	0.6555854	0.17010239
3	0e+00	0.6545461	0.09753304
3	1e-04	0.6507042	0.06896219
3	1e-01	0.6863893	0.24767014
5	0e+00	0.6513176	0.15634699
5	1e-04	0.6369879	0.12312206
5	1e-01	0.6812217	0.27544391

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were size = 3 and decay = 0.1.

Confusion Matrix and Statistics

	Reference	
Prediction	neg	pos
neg	85	35
pos	15	18

Accuracy : 0.6732

95% CI : (0.5928, 0.7468)
 No Information Rate : 0.6536
 P-Value [Acc > NIR] : 0.33820

Kappa : 0.2081

Mcnemar's Test P-Value : 0.00721

Sensitivity : 0.8500
 Specificity : 0.3396
 Pos Pred Value : 0.7083
 Neg Pred Value : 0.5455
 Prevalence : 0.6536
 Detection Rate : 0.5556
 Detection Prevalence : 0.7843
 Balanced Accuracy : 0.5948
 'Positive' Class : neg

PREDIÇÕES DE NOVOS CASOS

	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	predict.rna
1	1	139	62	41	480	40.7	536	21	neg
2	1	97	68	21	0	27.2	1095	22	neg
3	0	109	88	30	0	32.5	855	38	neg
4	9	171	110	24	240	45.4	721	54	neg

##REGRESSÃO ADMISSÃO

Técnica	Parâmetro	R2	Syx	Pearson	RMSE	MAE
RF – Hold-out	mtry=2	0.8215792	0.0622462	0.907053	0.05965146	0.04067818

Treinamos Randon Forest com a base de Treino

Random Forest

402 samples
 7 predictor

No pre-processing

Resampling: Bootstrapped (25 reps)
 Summary of sample sizes: 402, 402, 402, 402, 402, 402, ...
 Resampling results across tuning parameters:

mtry	RMSE	Rsquared	MAE
2	0.06376182	0.7952256	0.04587194
4	0.06593620	0.7819062	0.04723526
7	0.06838141	0.7671146	0.04919280

RMSE was used to select the optimal model using the smallest value.
 The final value used for the model was mtry = 2.

Calculamos as métricas

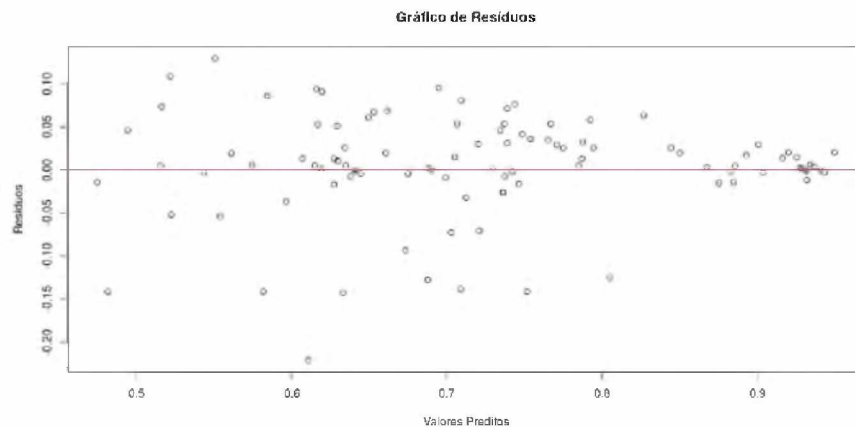
```
rmse(teste$ChanceOfAdmit, predicoes.rf)
[1] 0.05965146
r2 <- function(predito, observado) {
  return(1 - (sum((predito-observado)^2) / sum((observado-mean(observado))^2)))
}
r2(predicoes.rf, teste$ChanceOfAdmit)
[1] 0.8215792
```

Calculamos o Syx

```
Syx <- sqrt(sum((observado - predito)^2) / (n - p - 1))
Syx
[1] 0.0622462
```

```
# Agora calculamos o coeficiente de correlação de Pearson
[1] 0.907053
# MAE
[1] 0.04067818
```

Gráfico de resíduos



PREDIÇÕES DE NOVOS CASOS

	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predict.rf
1	299	96	2	1.5	2.0	7.86	0	0.4941738
2	319	110	3	3.0	2.5	8.79	0	0.7137322
3	297	99	4	3.0	3.5	7.81	0	0.5406497

BIOMASSA

Técnica	Parâmetro	R2	Syx	Pearson	RMSE	MAE
RF – CV	mtry=2	0.9287667	212.4259	0.9791502	205.2229	74.49094

Cross-validation RF

Random Forest
240 samples
3 predictor

No pre-processing

Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 216, 216, 216, 216, 216, 216, ...
Resampling results across tuning parameters:

mtry	RMSE	Rsquared	MAE
2	400.7352	0.9475062	117.9154
3	402.8242	0.9384469	119.6133

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.

Calcular as métricas

```
rmse(teste$biomassa, predicoes.rf)
[1] 205.2229
r2 <- function(predito, observado) {
  return(1 - (sum((predito-observado)^2) / sum((observado-mean(observado))^2)))
}
r2(predicoes.rf ,teste$biomassa)
[1] 0.9287667
```

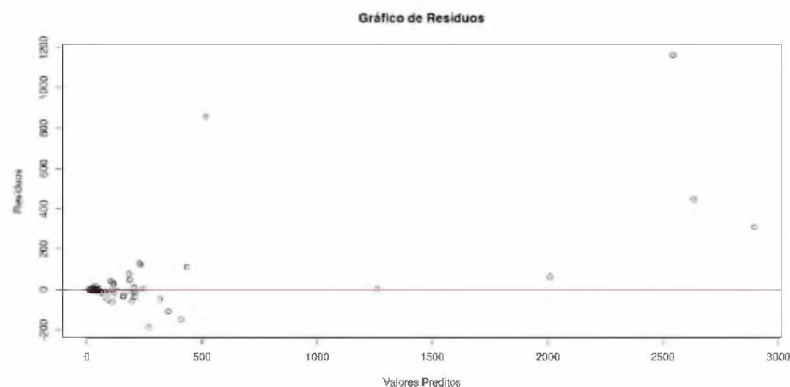
Syx

```
observado <- teste$biomassa
predito <- predicoes.rf
```

```
# Número de observações
n <- length(observado)
# Número de variáveis preditoras
p <- ncol(teste) - 1 # Excluímos a variável target
# Calcular o Syx ajustado
Syx <- sqrt(sum((observado - predito)^2) / (n - p - 1))
Syx
[1] 212.4259
# Coeficiente de correlação de Pearson
[1] 0.9791502

# MAE
mae <- mean(abs(observado - predito))
mae
[1] 74.49094
```

Calcular os resíduos



PREDIÇÕES DE NOVOS CASOS

	dap	h	Me	predict.rf
1	14	20	0.64	142.92983
2	15	20	0.61	159.02946
3	11	13	0.46	28.97886

AGRUPAMENTO

VEÍCULO

```
setwd("/home/magnus/Downloads/ml-007/ufpr-ia-aplicada/ml-r-superv-naosuperv/data")
dados <- read.csv("4 - Veiculos - Dados.csv", header = TRUE)
```

Remove 'a' (índice/ID)

```
dados$a <- NULL
```

Visualizamos a estrutura dos dados

```
str(dados)
```

'data.frame': 846 obs. of 19 variables:

\$ Comp : int 95 91 104 93 85 107 97 90 86 93 ...

\$ Circ : int 48 41 50 41 44 57 43 43 34 44 ...

\$ DCirc : int 83 84 106 82 70 106 73 66 62 98 ...

\$ RadRa : int 178 141 209 159 205 172 173 157 140 197 ...

\$ PrAxisRa : int 72 57 66 63 103 50 65 65 61 62 ...

\$ MaxLRa : int 10 9 10 9 52 6 6 9 7 11 ...

\$ ScatRa : int 162 149 207 144 149 255 153 137 122 183 ...

```
$ Elong : int 42 45 32 46 45 26 42 48 54 36 ...
$ PrAxisRect: int 20 19 23 19 19 28 19 18 17 22 ...
$ MaxLRect : int 159 143 158 143 144 169 143 146 127 146 ...
$ ScVarMaxis: int 176 170 223 160 241 280 176 162 141 202 ...
$ ScVarmaxis: int 379 330 635 309 325 957 361 281 223 505 ...
$ RaGyr : int 184 158 220 127 188 264 172 164 112 152 ...
$ SkewMaxis : int 70 72 73 63 127 85 66 67 64 64 ...
$ Skewmaxis : int 6 9 14 6 9 5 13 3 2 4 ...
$ Kurtmaxis : int 16 14 9 10 11 9 1 3 14 14 ...
$ KurtMaxis : int 187 189 188 199 180 181 200 193 200 195 ...
$ HollRa : int 197 199 196 207 183 183 204 202 208 204 ...
$ tipo : chr "van" "van" "saab" "van" ...
```

```
# Executamos o algoritmo K-means para clustering
```

```
set.seed(seedvarRandomState)
```

```
veiculosCluster <- kmeans(dados_numericos, 10)
```

```
veiculosCluster
```

```
K-means clustering with 10 clusters of sizes 67, 99, 47, 169, 78, 115, 24, 56, 114, 77
```

```
Cluster means:
```

```
Comp Circ DCirc RadRa PrAxisRa MaxLra ScatRa Elong PrAxisRect MaxLRect ScVarMaxis
1 94.04478 42.62687 83.16418 186.8060 63.80597 7.582090 171.6866 38.13433 20.55224 141.6567 193.7761
2 92.29293 41.68687 78.03030 181.3838 69.36364 11.222222 156.6263 42.18182 19.49495 140.8889 183.8687
3 97.85106 46.89362 93.17021 199.2553 64.40426 8.617021 186.2128 35.19149 21.78723 150.8936 206.7447
4 86.63905 44.30178 73.70414 140.7515 59.17160 8.136095 152.0473 44.41420 19.21302 147.8402 172.2426
5 103.87179 52.53846 103.02564 196.6282 61.26923 10.192308 211.5000 31.38462 23.89744 165.1923 223.9615
6 87.69565 37.61739 60.91304 128.0783 56.33043 6.243478 126.5826 53.27826 17.63478 131.7565 147.4348
7 107.25000 55.54167 103.41667 190.2500 55.79167 5.791667 248.7083 26.87500 27.08333 168.0833 271.2917
8 101.60714 49.48214 99.46429 206.4107 64.37500 9.250000 199.2679 33.00000 22.82143 155.8929 217.4286
9 88.55263 40.17544 72.90351 150.9211 60.85965 7.543860 142.2632 46.82456 18.50000 137.7105 164.8070
10 104.24675 54.16883 102.74026 206.5325 63.44156 10.636364 220.5325 30.29870 24.75325 171.3117 229.4286

      ScVarmaxis   RaGyr   SkewMaxis   Skewmaxis Kurtmaxis KurtMaxis HollRa
1      445.7015     163.0000  67.79104     5.805970   15.059701  194.6716  200.9254
2      368.0101     157.7374  71.67677     5.868687   12.111111  192.7576  199.5253
3      523.0851     182.3830  68.12766     6.957447   12.723404  193.5106  200.2553
4      339.6923     176.3195  77.33728     5.739645   8.988166   183.4379  189.6982
5      663.8077     212.4103  71.43590     7.205128   16.435897  188.3974  197.4744
6      236.7391     137.9130  73.68696     6.843478   11.382609  187.5043  192.1217
7      910.5417     247.1667  83.62500     6.500000   14.208333  182.6667  183.6250
8      596.4286     198.1071  69.12500     6.125000   13.571429  191.5536  199.0179
9      301.7544     149.8509  69.49123     5.894737   12.394737  191.1140  197.8246
10     717.1688     212.4026  72.03896     7.896104   15.961039  187.7792  197.6364
```

```
Within cluster sum of squares by cluster:
```

```
[1] 112567.25 315755.62 65754.85 151546.20 94564.06 166756.16 115198.75 77108.57 142335.30
83272.21
```

```
(between_SS / total_SS = 95.7 %)
```

```
# Executamos o algoritmo K-modes
```

```
K-modes clustering with 5 clusters of sizes 163, 155, 217, 139, 172
```

```
Cluster modes:
```

```
Comp Circ DCirc RadRa PrAxisRa MaxLra ScatRa Elong PrAxisRect MaxLRect ScVarMaxis ScVarmaxis RaGyr SkewMaxis
1      89   37  66  125  58      7   133  50  18      128   159      246   176  72
2     104  55 101  172  62     10   222  30  19      131   174      345   218  71
3      85   43  70  141  56      7   150  45  19      143   169      327   171  69
4      94   45 100  197  63     10   157  35  22      161   202      367   186  67
5     107  53 103  203  64     11   217  31  24      163   226      669   214  72

      Skewmaxis   Kurtmaxis   KurtMaxis   HollRa   Class
1           3           7          183       183     van
2           1           6          186       196     opel
3           6          11          181       182     bus
4           4          10          192       197     saab
5           0          11          189       199     saab
```

REGRAS DE ASSOCIAÇÃO

MUSCULAÇÃO

```
library(arules)
library(datasets)
# Definimos o diretório de trabalho e leitura dos dados
setwd("/home/magnus/Downloads/ml-007/ufpr-ia-aplicada/ml-r-superv-naosuperv/data")
dados <- read.transactions(file="2 - Musculacao - Dados.csv",format="basket",sep=";")
# Visualizamos os dados
inspect(dados[1:26])
items
items
[1] {LegPress,Gemeos,Afundo,Crucifixo,}
[2] {LegPress,Gemeos,Agachamento,,}
[3] {Gemeos,LegPress,Afundo,Agachamento,}
[4] {Adutor,Agachamento,LegPress,Adutor,}
[5] {LegPress,Gemeos,Afundo,Bicicleta,}
[6] {LegPress,Gemeos,Agachamento,,}
[7] {Gemeos,LegPress,Afundo,Bicicleta,}
[8] {Adutor,Agachamento,LegPress,Adutor,}
[9] {Extensor,AgachamentoSmith,Bicicleta,Esteira,Gemeos}
[10] {Extensor,AgachamentoSmith,Bicicleta,Esteira,LegPress}
[11] {Extensor,AgachamentoSmith,Bicicleta,Gemeos,LegPress}
[12] {Extensor,Flexor,Bicicleta,Esteira,LegPress}
[13] {Extensor,AgachamentoSmith,Gemeos,Esteira,LegPress}
[14] {Extensor,AgachamentoSmith,Bicicleta,Esteira,LegPress}
[15] {Extensor,Gemeos,Bicicleta,Esteira,}
[16] {LegPress,Gemeos,Afundo,Crucifixo,}
[17] {LegPress,Gemeos,Agachamento,,}
[18] {Gemeos,LegPress,Afundo,Agachamento,}
[19] {Adutor,Agachamento,LegPress,Adutor,}
[20] {Extensor,AgachamentoSmith,Bicicleta,Esteira,Gemeos}
[21] {Extensor,AgachamentoSmith,Bicicleta,Esteira,LegPress}
[22] {Extensor,AgachamentoSmith,Bicicleta,Gemeos,Afundo}
[23] {Extensor,Flexor,Bicicleta,Esteira,LegPress}
[24] {Afundo,AgachamentoSmith,Gemeos,Esteira,LegPress}
[25] {Extensor,AgachamentoSmith,Bicicleta,Esteira,LegPress}
[26] {Extensor, Gemeos,Bicicleta,Esteira,Afundo}
set.seed(seedvarRandomState)
```

Extraímos as regras

o suporte mínimo (supp) foi reduzido para 0.1 e a confiança mínima (conf) para 0.8.

para resultar em um número maior de regras extraídas.

```
rules <- apriori(dados, parameter = list(supp = 0.1, conf = 0.8, target = "rules"))
```

Apriori

Parameter specification:

```
confidence minval smax arem aval originalSupport maxtime support minlen maxlen target ext
0.8 0.1 1 none FALSE TRUE 5 0.1 1 10 rules TRUE
```

Algorithmic control:

```
filter tree heap memopt load sort verbose
```

```
0.1 TRUE TRUE FALSE TRUE 2 TRUE
```

Absolute minimum support count: 2

```
set item appearances ...[0 item(s)] done [0.00s].
```

```
set transactions ...[15 item(s), 26 transaction(s)] done [0.00s].
```

```
sorting and recoding items ... [3 item(s)] done [0.00s].
```

```
creating transaction tree ... done [0.00s].
```

```
checking subsets of size 1 done [0.00s].
```

```
writing ... [0 rule(s)] done [0.00s].
```

```
creating S4 object ... done [0.00s].
```

APÊNDICE 8 - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B - RESOLUÇÃO

1 Classificação de Imagens (CNN)

```
# Carregamento e preparação dos dados
# Já está separado em dados de treino e teste
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalização dos dados
x_train, x_test = x_train / 255.0, x_test / 255.0

# Convertemos as etiquetas para uma codificação one-hot
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Definimos a arquitetura da CNN
model = Sequential([
    Input(shape=(32, 32, 3)),

    Conv2D(32, (3, 3),
    activation='relu'),

    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3),
    activation='relu'),

    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3),
    activation='relu'),

    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten_1 (Flatten)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 10)	1290

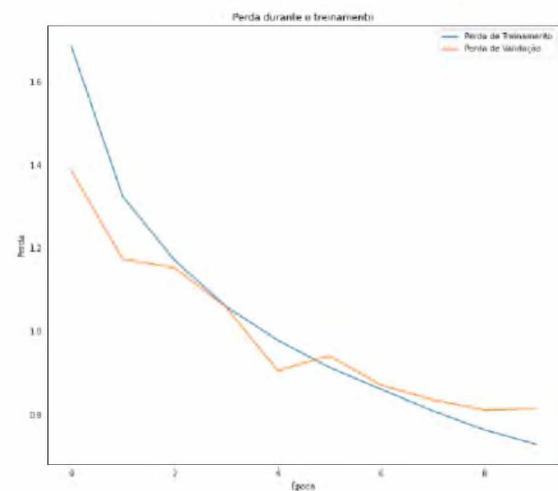
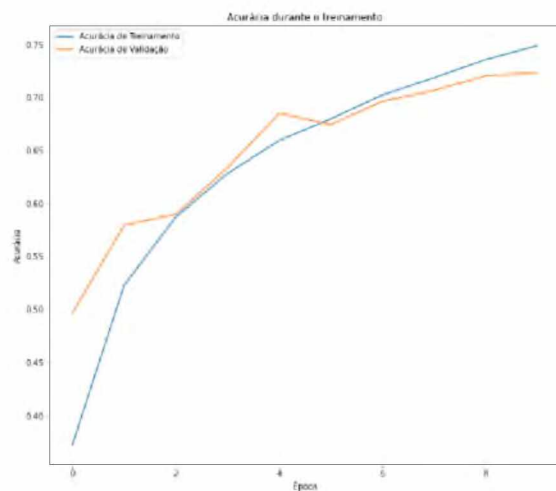

```

=====
Total params: 160202 (625.79 KB)
Trainable params: 160202 (625.79 KB)
Non-trainable params: 0 (0.00 Byte)

# Compilamos o modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Treinamos o modelo
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test),
batch_size=64)
Epoch 1/10
782/782 [=====] - 35s 44ms/step - loss:
1.6875 - accuracy: 0.3730 - val_loss: 1.3881 - val_accuracy: 0.4970
Epoch 2/10
782/782 [=====] - 34s 43ms/step - loss:
1.3250 - accuracy: 0.5231 - val_loss: 1.1739 - val_accuracy: 0.5797
Epoch 3/10
782/782 [=====] - 35s 44ms/step - loss:
1.1722 - accuracy: 0.5874 - val_loss: 1.1539 - val_accuracy: 0.5900
Epoch 4/10
782/782 [=====] - 35s 44ms/step - loss:
1.0605 - accuracy: 0.6283 - val_loss: 1.0590 - val_accuracy: 0.6336
Epoch 5/10
782/782 [=====] - 34s 44ms/step - loss:
0.9794 - accuracy: 0.6596 - val_loss: 0.9065 - val_accuracy: 0.6851
Epoch 6/10
782/782 [=====] - 35s 44ms/step - loss:
0.9141 - accuracy: 0.6799 - val_loss: 0.9421 - val_accuracy: 0.6746
Epoch 7/10
782/782 [=====] - 34s 44ms/step - loss:
0.8618 - accuracy: 0.7025 - val_loss: 0.8717 - val_accuracy: 0.6965
Epoch 8/10
782/782 [=====] - 35s 44ms/step - loss:
0.8097 - accuracy: 0.7187 - val_loss: 0.8364 - val_accuracy: 0.7068
Epoch 9/10
782/782 [=====] - 35s 44ms/step - loss:
0.7641 - accuracy: 0.7359 - val_loss: 0.8116 - val_accuracy: 0.7206
Epoch 10/10
782/782 [=====] - 34s 44ms/step - loss:
0.7292 - accuracy: 0.7490 - val_loss: 0.8142 - val_accuracy: 0.7234

```



```
import numpy as np
import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

# Fazer previsões no conjunto de teste
y_pred = model.predict(x_test).argmax(axis=1) # Convertemos as previsões para classes
y_true = np.argmax(y_test, axis=1)           # Convertemos as etiquetas verdadeiras para classes

# Criar a matriz de confusão
cm = confusion_matrix(y_true, y_pred)

# Plotamos a matriz de confusão
plt.figure(figsize=(15, 15))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['aviao', 'automovel', 'passaro', 'gato', 'servo', 'cachorro',
                        'sapo', 'cavalo', 'navio', 'caminhão'],
            yticklabels=['aviao', 'automovel', 'passaro', 'gato', 'servo', 'cachorro',
                        'sapo', 'cavalo', 'navio', 'caminhão'],
            cbar=False, square=True, linewidths=0.5, linecolor='black')

plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Matriz de Confusão')
plt.show()
```

Matriz de Confusão											
True Label	airplane	716	26	54	17	15	12	33	17	30	90
	automobile	12	641	4	10	2	7	9	6	9	100
	bird	56	5	567	47	63	104	72	34	11	21
	cat	12	6	53	44	62	262	81	44	14	20
	deer	17	5	73	55	65	33	55	100	10	1
	dog	6	3	38	95	44	759	29	43	1	12
	frog	5	4	43	45	38	22	212	9	4	8
	horse	11	2	34	26	45	74	6	701	1	20
	ship	75	44	14	18	2	9	5	5	783	45
	truck	15	60	5	12	3	9	8	19	14	656
		airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
Predicted Label											

```
# Definimos os rótulos das classes
```

```
labels = ['aviao', 'automovel', 'passaro', 'gato', 'servo', 'cachorro', 'sapo', 'cavalo', 'navio', 'caminhão']
```

```
# Localizamos índices de previsões incorretas
```

```
misclassified = np.where(y_pred != y_true)[0]
```

```
# Escolhemos uma amostra aleatória de previsões incorretas
```

```
i = np.random.choice(misclassified)
```

```
# Exibimos a imagem mal classificada
```

```
plt.figure(figsize=(10, 8))
```

```
plt.imshow(x_test[i]) # Remover cmap='gray' porque as imagens são coloridas
```

```
plt.title("True label: %s Predicted: %s" % (labels[y_true[i]], labels[y_pred[i]]))
```

```
plt.axis('off') # Opcional: remover os eixos
```

```
plt.show()
```

True label: automobile Predicted: truck



```
import numpy as np
from sklearn.metrics import accuracy_score

# Calculamos a acurácia final
accuracy = accuracy_score(np.argmax(y_test, axis=1), y_pred)

# Número de imagens a serem exibidas
num_images = 16

# Escolhemos imagens aleatórias para exibição
indices = np.random.choice(len(x_test), num_images, replace=False)

# Configuramos a figura e os eixos
fig = plt.figure(figsize=(20, 20))

# Exibimos a acurácia no topo das figuras
fig.suptitle(f"Acurácia final: {accuracy:.4f}", fontsize=16, bbox=dict(facecolor='white',
alpha=0.5, edgecolor='black'))
fig.text(0.5, 1.05, 'As imagens com texto em vermelho não foram classificadas
corretamente',
        ha='center', va='center', fontsize=12,
        bbox=dict(facecolor='white', alpha=0.5, edgecolor='black'))

for i, idx in enumerate(indices):
    ax = fig.add_subplot(4, 4, i + 1)
    ax.imshow(x_test[idx]) # Exibir a imagem
    true_label = labels[np.argmax(y_test[idx])] # Obter o rótulo verdadeiro
    pred_label = labels[y_pred[idx]] # Obter o rótulo previsto
    # Definimos a cor do texto com base se a previsão está correta
    color = 'black' if true_label == pred_label else 'red'
```

```

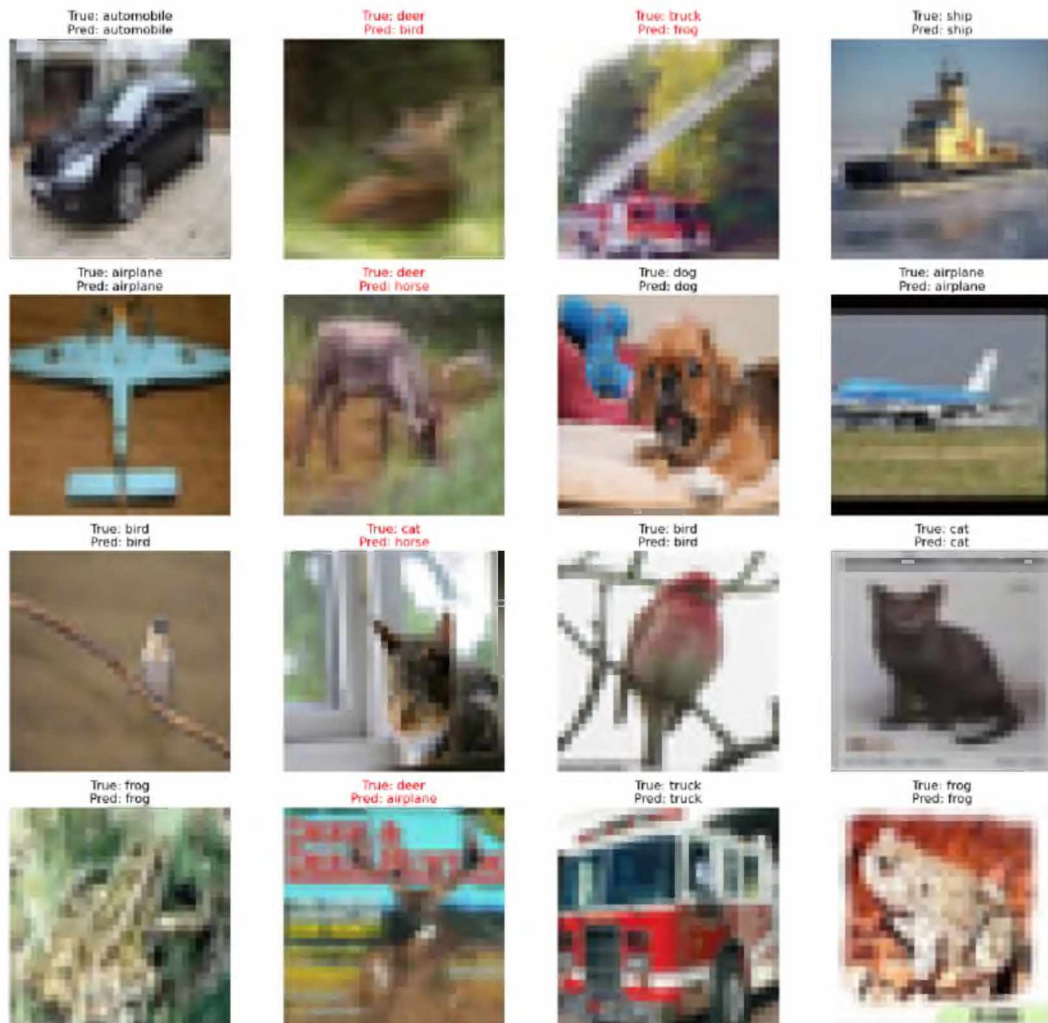
ax.set_title(f"True: {true_label}\nPred: {pred_label}", fontsize=16, color=color)
ax.axis('off') # remover os eixos

plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()

```

(As imagens com texto em vermelho não foram classificadas corretamente)

Acurácia final: 0.7234



2 Detector de SPAM (RNN)

EXEMPLO RNN - LSTM

```

# 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

```

```

# Suprimir todos os warnings
import warnings

# Suprimir warnings específicos
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=RuntimeWarning)

# carrega e arruma a base
# !wget http://www.razer.net.br/datasets/spam.csv

df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()

df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]

df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values

# 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)

# 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, 121)
data_test.shape: (1839, 121)

# 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, 121)]	0
embedding (Embedding)	(None, 121, 20)	144920
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6
Total params: 145446 (568.15 KB)		
Trainable params: 145446 (568.15 KB)		
Non-trainable params: 0 (0.00 Byte)		

```
# 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 [=====] - 7s 39ms/step - loss: 0.4177 - accuracy: 0.8642 - val_loss: 0.2055 - val_accuracy: 0.9353
```

```
Epoch 2/5
```

```
117/117 [=====] - 4s 35ms/step - loss: 0.1571 - accuracy: 0.9660 - val_loss: 0.1173 - val_accuracy: 0.9831
```

```
Epoch 3/5
```

```
117/117 [=====] - 4s 36ms/step - loss: 0.0958 - accuracy: 0.9877 - val_loss: 0.0880 - val_accuracy: 0.9880
```

```
Epoch 4/5
```

```
117/117 [=====] - 4s 35ms/step - loss: 0.0649 - accuracy: 0.9946 - val_loss: 0.0715 - val_accuracy: 0.9886
```

```
Epoch 5/5
```

```
117/117 [=====] - 5s 39ms/step - loss: 0.0459 - accuracy: 0.9976 - val_loss: 0.0616 - val_accuracy: 0.9891
```

```
# Plota função de perda e acurácia
```

```
plt.figure(figsize=(15, 8))
```

```
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.figure(figsize=(15, 8))
```

```
plt.plot(r.history["accuracy"], label="accuracy")
```

```
plt.plot(r.history["val_accuracy"], label="val_accuracy")
```

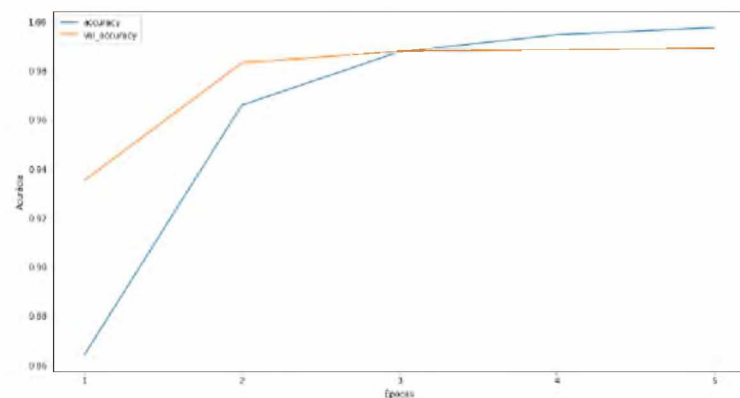
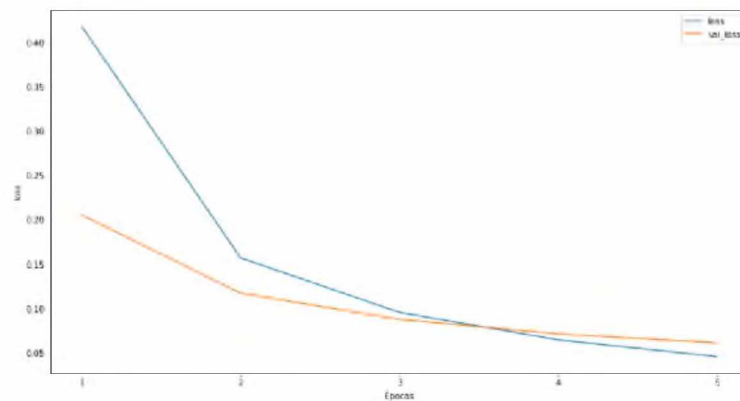
```
plt.xlabel("Épocas")
```

```
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

texto1 = "Today is a big day to do something bigger."

texto2 = "This is not a SPAM, so, Is your car dirty? Discover our new product. Free for all. Click the link. Remember me, this message is not a SPAM."

texto3 = "Is your car dirty? Discover our new product. Free for all. Click the link."

```
seq_texto1 = tokenizer.texts_to_sequences([texto1]) # Tokeniza
data_texto1 = pad_sequences(seq_texto1, maxlen=T) # Padding
pred1 = model.predict(data_texto1) # Predição
```

```
seq_texto2 = tokenizer.texts_to_sequences([texto2]) # Tokeniza
data_texto2 = pad_sequences(seq_texto2, maxlen=T) # Padding
pred2 = model.predict(data_texto2) # Predição
```

```
seq_texto3 = tokenizer.texts_to_sequences([texto3]) # Tokeniza
data_texto3 = pad_sequences(seq_texto3, maxlen=T) # Padding
pred3 = model.predict(data_texto3) # Predição
```

```
print(pred1)
print ("SPAM" if pred1 >= 0.5 else "OK, NÃO É SPAM")
print ("SPAM" if pred2 >= 0.5 else "OK, NÃO É SPAM")
print ("SPAM" if pred3 >= 0.5 else "OK, NÃO É SPAM")
```

```
1/1 [=====] - 0s 74ms/step
1/1 [=====] - 0s 46ms/step
```

```
1/1 [=====] - 0s 41ms/step
[[0.01130716]]
OK, NÃO É SPAM
OK, NÃO É SPAM
SPAM
```


3 Gerador de Dígitos Fake (GAN)

EXEMPLO GAN

Para Gerar os GIFs

!pip install imageio

!pip install git+https://github.com/tensorflow/docs

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).batch(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

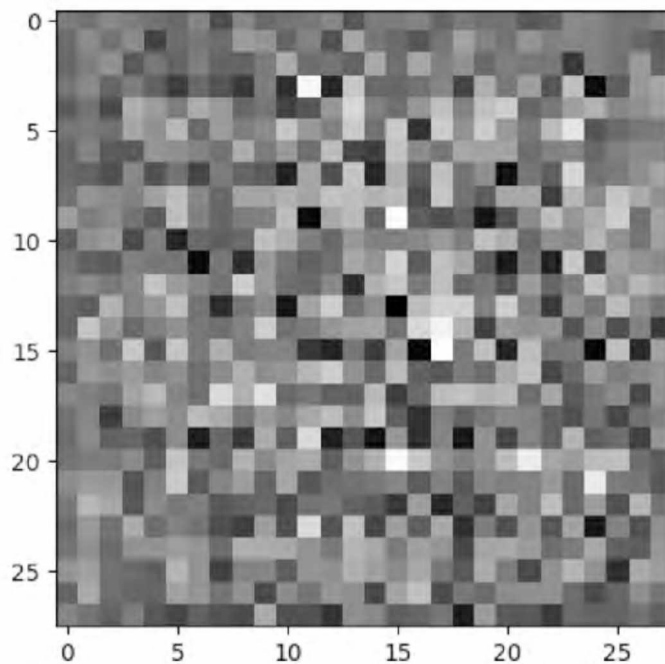
Teste do GERADOR, ainda não treinado

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)

# 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)

# 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)

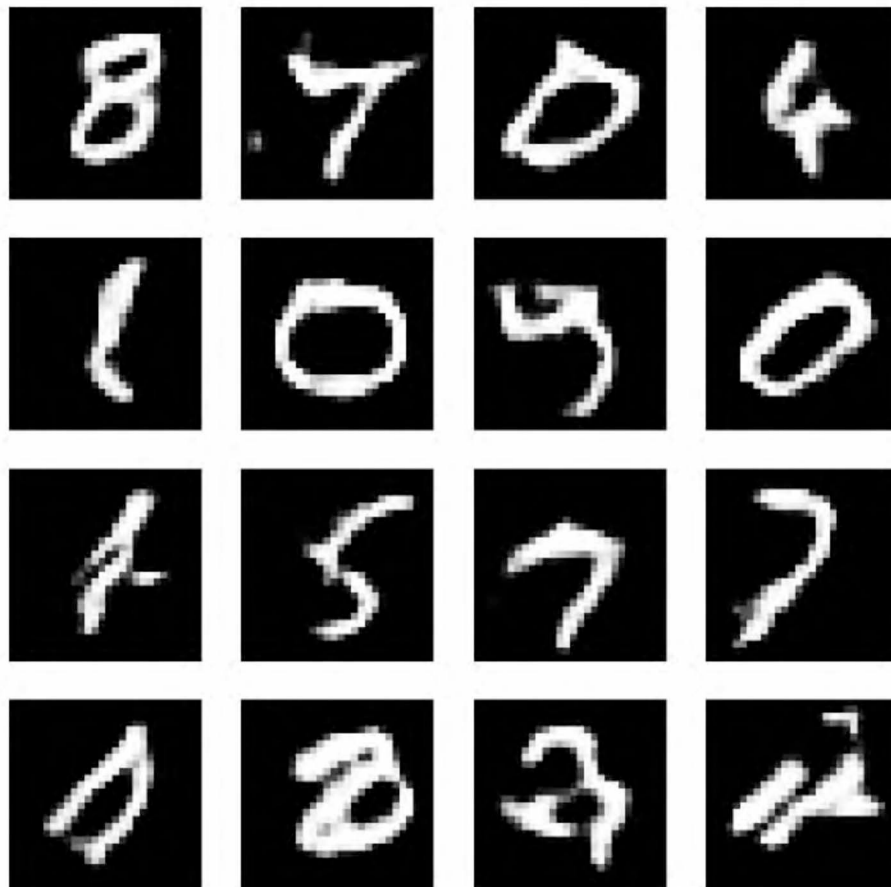
    fig = plt.figure(figsize=(15, 15))

    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)

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

```

4 Tradutor de Textos (Transformer)

EXEMPLO 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

import collections
import logging
import os
import pathlib
import re
import string
import sys
import time
import numpy as np
import matplotlib.pyplot as plt
import tensorflow_datasets as tfds
import tensorflow_text as text
import tensorflow as tf

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

# Suprimir todos os warnings
import warnings

# Suprimir warnings específicos
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=RuntimeWarning)

# 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'))

```

```

# 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 denso
    # Faz padding com zeros.
    pt = pt.to_tensor()

    en = tokenizers.en.tokenize(en)
    # ragged -> denso
    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, ...]

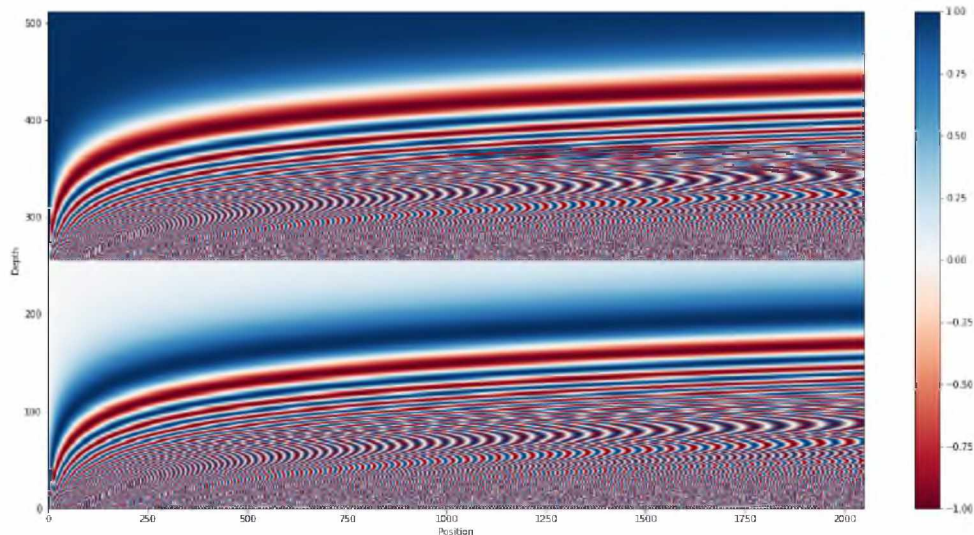
    return tf.cast(pos_encoding, dtype=tf.float32)

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.figure(figsize=(20,10))
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

## Cria rede feed-forward pontual

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)
                                ])

## Camada codificador

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

## Camada decodificador
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
dff             = 512
num_heads       = 8
dropout_rate    = 0.1

## Otimizador

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)

##Função de Perda e Métrica de Acurácia (mascarados)

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')

```

Treinamento

```

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 20 Batch 0 Loss 1.1604 Accuracy 0.7215
Epoch 20 Batch 50 Loss 1.0697 Accuracy 0.7437
Epoch 20 Batch 100 Loss 1.0731 Accuracy 0.7427
Epoch 20 Batch 150 Loss 1.0751 Accuracy 0.7425
Epoch 20 Batch 200 Loss 1.0802 Accuracy 0.7415
Epoch 20 Batch 250 Loss 1.0873 Accuracy 0.7402
Epoch 20 Batch 300 Loss 1.0931 Accuracy 0.7389
Epoch 20 Batch 350 Loss 1.0985 Accuracy 0.7381
Epoch 20 Batch 400 Loss 1.1004 Accuracy 0.7379
Epoch 20 Batch 450 Loss 1.1019 Accuracy 0.7376
Epoch 20 Batch 500 Loss 1.1032 Accuracy 0.7375
Epoch 20 Batch 550 Loss 1.1037 Accuracy 0.7373
Epoch 20 Batch 600 Loss 1.1059 Accuracy 0.7369
Epoch 20 Batch 650 Loss 1.1081 Accuracy 0.7365
Epoch 20 Batch 700 Loss 1.1112 Accuracy 0.7358
Epoch 20 Batch 750 Loss 1.1142 Accuracy 0.7353
Epoch 20 Batch 800 Loss 1.1183 Accuracy 0.7346
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-7
Epoch 20 Loss 1.1188 Accuracy 0.7344
Time taken for 1 epoch: 1969.84 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

## Efetuar uma tradução
translator = Translator(tokenizers, transformer)
sentence1 = "Hoje seria um ótimo dia para uma corrida"

```

```

translated_text1,          translated_tokens,          attention_weights          =
translator(tf.constant(sentence1))

sentence2 = "poderemos usar nossa força para competir com os melhor de cada esporte"
translated_text2,          translated_tokens,          attention_weights          =
translator(tf.constant(sentence2))

sentence3 = "e quem sabe conseguir vencer os melhores do mundo"
translated_text3,          translated_tokens,          attention_weights          =
translator(tf.constant(sentence3))

print(f'{"1":5s}: {translated_text1}')
print(f'{"2":5s}: {translated_text2}')
print(f'{"3":5s}: {translated_text3}')

1    : b'now it would be a great day to run a race , today , a great day .'
2    : b'we can use our strength to compete with the best of each sterend .'
3    : b'and who knows what can beat the best world .'

```

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

Escopo de Projeto: Avaliação de Preços no Contexto Supermercadista no Brasil

1. Objetivo do Projeto

Desenvolver um sistema de Big Data para integração e análise de preços que permita a avaliação precisa e em tempo real dos preços praticados nos setores de varejo e atacado, no contexto supermercadista no território nacional. O sistema integrará dados de múltiplas fontes, incluindo bases de dados de pontos comerciais físicos, informações de representantes de produtos e dados de compras online, proporcionando uma visão abrangente e atualizada do mercado.

2. Escopo

O projeto abrangerá as seguintes etapas:

1. Levantamento de Requisitos

- Identificar todas as fontes de dados disponíveis, como bases de dados de pontos comerciais (varejo e atacado), informações fornecidas pelos representantes de produtos, planilhas de contato e vendas e dados de compras online.
- Definir os indicadores-chave de desempenho (KPIs) para avaliação de preços, como variação de preços por região, sazonalidade, concorrência, e margem de lucro.

2. Integração de Dados

- Conectar e integrar as diferentes bases de dados de pontos comerciais, utilizando ETL (Extração, Transformação e Carga) para padronização e limpeza dos dados.
- Integrar os dados de representantes de produtos, com foco em negociações de preços, promoções e acordos comerciais.
- Conectar as bases de dados de compras online, mapeando o comportamento do consumidor e identificando tendências de mercado.

3. Desenvolvimento do Sistema de Análise

- Implementar um sistema de Big Data utilizando tecnologias como Hadoop, Spark, ou outras, para processamento e análise em larga escala.
- Desenvolver algoritmos de machine learning para prever tendências de preços e identificar oportunidades de ajuste.
- Criar dashboards interativos para visualização dos dados e geração de relatórios customizados, utilizando ferramentas como Power BI, Tableau ou soluções baseadas em Python.

4. Validação e Testes

- Realizar testes de desempenho e acurácia com conjuntos de dados históricos e em tempo real.
- Validar os modelos de previsão de preços e ajustar os algoritmos conforme necessário.
- Implementar um ambiente de testes com amostras de dados para validação contínua.

5. Treinamento e Capacitação

- Capacitar os colaboradores dos supermercados e representantes de produtos na utilização do sistema.
- Desenvolver manuais e tutoriais para a correta interpretação dos dados e utilização das ferramentas de análise.

6. Implementação e Manutenção

- Implementar o sistema nos pontos comerciais selecionados, garantindo a integração contínua com as bases de dados.
- Estabelecer um cronograma de manutenção e atualização do sistema, garantindo que os dados permaneçam atualizados e precisos.
- Monitorar a performance do sistema e realizar ajustes conforme necessário para otimizar o desempenho.

3. Benefícios Esperados

- Melhoria na competitividade: Ajustes de preços mais rápidos e precisos, alinhados com as tendências de mercado.
- Redução de custos: Identificação de oportunidades de compra e negociação com fornecedores.
- Satisfação do consumidor: Preços mais justos e competitivos, resultando em maior fidelização.
- Decisões baseadas em dados: Capacidade de tomar decisões estratégicas baseadas em análises profundas e preditivas.

4. Recursos Necessários

- Tecnologia: Servidores de processamento de dados, ferramentas de Big Data (Hadoop, Spark), ferramentas de visualização (Power BI, Tableau).
- Equipe: Cientistas de dados, engenheiros de dados, especialistas em BI, desenvolvedores de software, e analistas de mercado.
- Orçamento: Investimento inicial para aquisição de tecnologia, contratação de especialistas, e desenvolvimento do sistema.

5. Cronograma

- Fase 1: Levantamento de Requisitos e Integração de Dados (3 meses)
- Fase 2: Desenvolvimento do Sistema e Algoritmos de Análise (6 meses)
- Fase 3: Validação, Testes, e Treinamento (2 meses)
- Fase 4: Implementação e Monitoramento (3 meses)

6. Riscos e Mitigações

- **Riscos de Integração de Dados:** Problemas de compatibilidade entre diferentes bases de dados podem ser mitigados com um processo robusto de ETL.
- **Riscos de Precisão nas Previsões:** Implementação de validações contínuas para garantir que os modelos de machine learning estejam sempre alinhados com as condições de mercado.
- **Riscos de Adoção:** Investir em treinamento e suporte contínuo para garantir a adoção bem-sucedida do sistema pelos usuários finais.

Este escopo servirá como base para o planejamento detalhado e a execução do projeto, visando a criação de um sistema robusto e eficiente de avaliação de preços no setor supermercadista brasileiro.

Tipos de Bancos de Dados: SGDB e NoSQL

1. Sistemas de Gerenciamento de Banco de Dados (SGDB)

Os SGDBs são usados para gerenciar bancos de dados relacionais (RDBMS), que armazenam dados em tabelas organizadas em linhas e colunas. Esses sistemas utilizam SQL (Structured Query Language) para a manipulação de dados.

Principais Tipos de SGDBs:

- MySQL: Um dos SGDBs mais populares, é conhecido por sua performance e robustez em ambientes web. Ele suporta ACID (Atomicidade, Consistência, Isolamento e Durabilidade) e é ideal para transações que exigem consistência e integridade.
- PostgreSQL: Um SGDB avançado, com suporte a transações complexas e consultas avançadas. É altamente extensível e oferece suporte para JSON, permitindo armazenamento de dados semiestruturados.
- Oracle: Utilizado em grandes corporações, oferece alta performance, escalabilidade e funcionalidades avançadas de segurança e replicação.
- SQL Server: Desenvolvido pela Microsoft, é conhecido por sua integração com outras ferramentas da Microsoft, como o Power BI, e por sua robustez em ambientes empresariais.

Conexões Entre SGDBs:

- ETL (Extract, Transform, Load): Ferramentas como Talend, Apache Nifi e Informatica podem ser usadas para conectar, extrair, transformar e carregar dados de diferentes SGDBs, garantindo a integração entre eles.
- ODBC/JDBC: Protocolos que permitem a comunicação entre diferentes SGDBs e aplicações, possibilitando consultas e manipulação de dados de forma centralizada.

2. Bancos de Dados NoSQL

Bancos de dados NoSQL são projetados para lidar com grandes volumes de dados distribuídos e para oferecer flexibilidade no armazenamento de dados semiestruturados ou não estruturados. Eles não dependem de esquemas fixos e oferecem escalabilidade horizontal.

Principais Tipos de NoSQL:

- MongoDB: Um banco de dados orientado a documentos, que armazena dados em formato JSON, ideal para aplicações que precisam de flexibilidade no armazenamento de dados semiestruturados.
- Cassandra: Um banco de dados de larga escala, projetado para alta disponibilidade e escalabilidade, utilizado para grandes volumes de dados distribuídos.
- Redis: Um banco de dados em memória, que oferece alta performance para operações de leitura e escrita, frequentemente usado como cache.
- HBase: Um banco de dados distribuído e escalável, que funciona sobre o HDFS (Hadoop Distributed File System), adequado para grandes volumes de dados com necessidades de leitura e escrita rápidas.

Conexões Entre NoSQL:

- Data Pipelines: Ferramentas como Apache Kafka ou Apache NiFi permitem o fluxo contínuo de dados entre diferentes bancos de dados NoSQL, garantindo que os dados sejam transmitidos e transformados conforme necessário.
- APIs e SDKs: NoSQL geralmente fornece APIs e SDKs para integração com outras aplicações ou sistemas de bancos de dados, facilitando a comunicação entre diferentes sistemas.

Hadoop e Ferramentas de Integração de Big Data

O Hadoop é um framework de código aberto que permite o processamento distribuído de grandes volumes de dados. Ele é composto por vários módulos que podem ser usados para compatibilizar e traduzir informações provenientes de diferentes fontes de dados, sejam elas SGDB ou NoSQL.

Principais Componentes do Hadoop:

- HDFS (Hadoop Distributed File System): Sistema de arquivos distribuído que armazena grandes volumes de dados de forma distribuída e resiliente.
- MapReduce: Modelo de programação para processamento paralelo de grandes conjuntos de dados. Ele divide o processamento em duas etapas: Map (mapeamento) e Reduce (redução), permitindo a análise de dados em larga escala.
- YARN (Yet Another Resource Negotiator): Gerenciador de recursos que aloca e monitora recursos do cluster Hadoop para diferentes aplicações.
- Hive: Um sistema de data warehousing sobre Hadoop que permite consultas em grandes volumes de dados usando uma linguagem semelhante ao SQL chamada HiveQL.
- Pig: Uma plataforma de alto nível para criar programas que rodam sobre Hadoop. Usa uma linguagem chamada Pig Latin, que abstrai as complexidades do MapReduce.
- HBase: Banco de dados NoSQL distribuído que roda sobre o HDFS, permitindo leitura e escrita rápidas de dados estruturados e semiestruturados.
- Sqoop: Ferramenta usada para importar e exportar dados entre bancos de dados relacionais e o Hadoop.
- Flume: Serviço para coletar, agregar e mover grandes quantidades de dados de log para o HDFS.

- Oozie: Gerenciador de fluxo de trabalho que coordena os trabalhos de MapReduce, Pig e Hive.

Integração de SGDBs e NoSQL com Hadoop no Projeto de Big Data

Para o contexto do projeto de avaliação de preços no setor supermercadista:

1. Coleta e Integração de Dados:

- Apache Sqoop pode ser usado para importar dados dos SGDBs (MySQL, PostgreSQL, etc.) para o HDFS.
- Flume pode ser utilizado para coletar dados de logs de compras online e armazená-los no sistema de arquivos HDFS.
- Kafka pode ser empregado para transmitir dados em tempo real de diferentes fontes, como registros de vendas e dados de representantes de produtos.

2. Processamento e Análise:

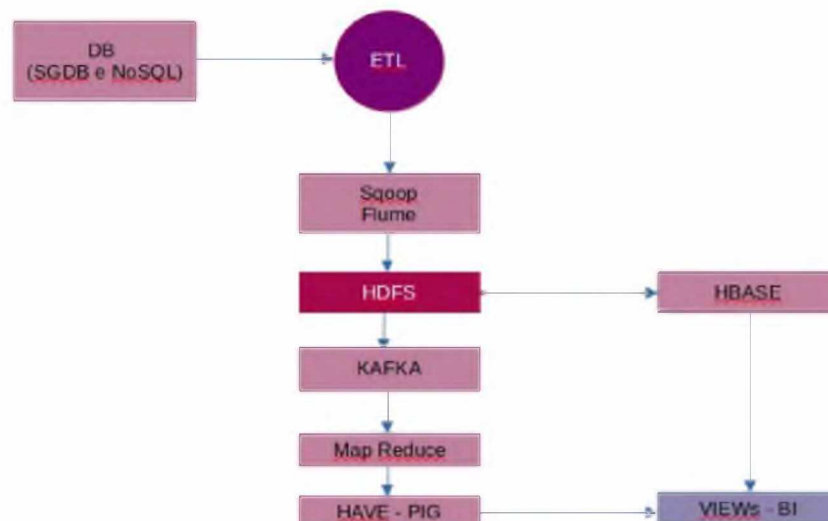
- MapReduce pode ser usado para processar grandes volumes de dados e calcular métricas de avaliação de preços.
- Hive ou Pig podem ser utilizados para consultas e análises complexas sobre os dados coletados e armazenados no HDFS.

3. Armazenamento e Consulta de Dados:

- HBase pode ser utilizado para armazenamento de dados que necessitam de alta velocidade de leitura e escrita, como registros de preços em tempo real.
- Hive permite consultas SQL-like sobre dados armazenados no HDFS, facilitando a extração de insights.

4. Visualização e Relatórios:

- Tableau ou Power BI podem ser conectados ao Hadoop via ODBC/JDBC para a criação de dashboards interativos e geração de relatórios em tempo real.



Conclusão

A integração dos diversos tipos de SGDBs e NoSQL com a infraestrutura Hadoop permitirá um fluxo contínuo de dados e análises avançadas, garantindo que o sistema de avaliação de preços no contexto supermercadista seja eficiente, escalável e robusto. Essa combinação de tecnologias assegura que os dados de múltiplas fontes sejam compatibilizados, traduzidos e analisados de forma a oferecer insights valiosos para a tomada de decisões estratégicas.

APÊNDICE 10 - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B - RESOLUÇÃO

1 - Extração de Características

Extração de características com LBP, nos dados de treinamento.

Gravados no arquivo CSV : lbp_features_train.csv

Criamos uma coluna Imagem que contem o nome da imagem, Cliente com o código do paciente e uma com a respectiva Classe

```
# Caminho das imagens de treino
train_data_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Train'
# Parâmetros do LBP
radius = 1
n_points = 8 * radius
# Função para extrair LBP
def extract_lbp_features(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    lbp = local_binary_pattern(gray_image, n_points, radius, method='uniform')
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range=(0,
n_points + 2))
    hist = hist.astype("float")
    hist /= (hist.sum() + 1e-6) # Normalização
    return hist

# Abrir arquivo CSV para salvar as características LBP
with open('lbp_features_train.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Image', 'Classe', 'Cliente', 'Features'])
    for class_dir in os.listdir(train_data_dir):
        class_path = os.path.join(train_data_dir, class_dir)
        if os.path.isdir(class_path):
            for image_name in os.listdir(class_path):
                image_path = os.path.join(class_path, image_name)
                image = cv2.imread(image_path)
                if image is not None:
                    features = extract_lbp_features(image)
                    # Extrair os dois primeiros dígitos do nome do arquivo
                    cliente = image_name[:2]

                    writer.writerow([image_name, class_dir, cliente,
features.tolist()])
print("Extração de características com LBP concluída!")

# Abrir arquivo CSV para salvar as características VGG16 do conjunto de validação
with open('vgg16_features_train.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Image', 'Classe', 'Cliente', 'Features'])
    current_file = 0 # Inicializa o contador
    for class_dir in os.listdir(train_data_dir):
        class_path = os.path.join(train_data_dir, class_dir)
        if os.path.isdir(class_path):
            for image_name in os.listdir(class_path):
                current_file += 1 # Incrementa o contador
                image_path = os.path.join(class_path, image_name)
                # Extrair os dois primeiros dígitos do nome do arquivo (Cliente)
```

```

        cliente = image_name[:2]

        features = extract_vgg16_features(image_path)
        writer.writerow([image_name, class_dir, cliente, features.tolist()])
        print(f'Processando arquivo {current_file}/{total_files}')

#Exibe o progresso
print("Extração de características VGG16 concluída!")

# Arquivo CSV com os dados LBP de treino
csv_filename_train = 'lbp_features_train.csv'
# Carregar o arquivo CSV em um DataFrame
df_train = pd.read_csv(csv_filename_train)

# Função para aplicar o estilo de cores alternadas
def color_alternate_rows(row):
    # Alterna as cores entre duas opções
    color = 'background-color: #f5f5f5' if row.name % 2 == 0 else 'background-color:
#ffffff'
    return [color] * len(row)

# Selecionar apenas as primeiras 10 linhas
df_train_10 = df_train.head(10)
# Exibir o DataFrame com linhas alternadas em cores
styled_df_10 = df_train_10.style.apply(color_alternate_rows, axis=1)
# Mostrar o DataFrame estilizado com 10 linhas
styled_df_10

```

pandas.io.formats.style.Styler at 0x7a68495d8160>

```

# Arquivo CSV com os dados VGG16 de treino
csv_filename_train = 'vgg16_features_train.csv'
# Carregar o arquivo CSV em um DataFrame
df_train = pd.read_csv(csv_filename_train)

# Função para aplicar o estilo de cores alternadas
def color_alternate_rows(row):
    # Alterna as cores entre duas opções
    color = 'background-color: #f5f5f5' if row.name % 2 == 0 else 'background-color:
#ffffff'
    return [color] * len(row)

# Selecionar apenas as primeiras 10 linhas
df_train_2 = df_train.head(2)
# Exibir o DataFrame com linhas alternadas em cores
styled_df_2 = df_train_2.style.apply(color_alternate_rows, axis=1)
# Mostrar o DataFrame estilizado com 2 linhas
styled_df_2

```

pandas.io.formats.style.Styler at 0x7a6840902b60>

Leitura e preparação dos dados com as características extraídas LBP

Separamos as bases agrupados por pacientes para garantir que o mesmo cliente fique apenas em uma base para não gerar viés, e fizemos a distribuição das classes

Distribuição das classes no conjunto de treino:

```

      Classe
2        120
3        120
1        117
0        116
Name: count, dtype: int64

```

Distribuição das classes no conjunto de teste:

```

      Classe

```

```

2    30
1    30
3    29
0    29

```

Name: count, dtype: int64

Tamanho do conjunto de treino: 473

Tamanho do conjunto de teste: 118

Treinamento e avaliação do Modelo Random Forest com as características LBP

Cada modelo foi gravado em um arquivo pkl para rodar posteriormente com os arquivo da base de teste.

Accuracy: 79.66%

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.75	0.72	0.74	29
1	0.72	0.77	0.74	30
2	0.81	0.73	0.77	30
3	0.90	0.97	0.93	29
accuracy			0.80	118
macro avg	0.80	0.80	0.80	118
weighted avg	0.80	0.80	0.80	118

Agora visualizamos as métricas de Especificidade , Sensibilidade e F1-Score de cada classe

Métricas referente ao Random Forest com LBP

Classe 0: Especificidade = 0.92, Sensibilidade = 0.72, F1-score = 0.74

Classe 1: Especificidade = 0.90, Sensibilidade = 0.77, F1-score = 0.74

Classe 2: Especificidade = 0.94, Sensibilidade = 0.73, F1-score = 0.77

Classe 3: Especificidade = 0.97, Sensibilidade = 0.97, F1-score = 0.93

Treinamento do modelo SVM com os dados LBP

Accuracy SVM: 74.58%

Relatório de Classificação (SVM):

	precision	recall	f1-score	support
0	0.74	0.69	0.71	29
1	0.67	0.60	0.63	30
2	0.66	0.70	0.68	30
3	0.91	1.00	0.95	29
accuracy			0.75	118
macro avg	0.74	0.75	0.74	118
weighted avg	0.74	0.75	0.74	118

Métricas referente ao SVM com LBP

Classe 0: Especificidade = 0.92, Sensibilidade = 0.69, F1-score = 0.71

Classe 1: Especificidade = 0.90, Sensibilidade = 0.60, F1-score = 0.63

Classe 2: Especificidade = 0.88, Sensibilidade = 0.70, F1-score = 0.68

Classe 3: Especificidade = 0.97, Sensibilidade = 1.00, F1-score = 0.95

Treinamento RNA com LBP

Accuracia RNA: 67.80%

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.64	0.48	0.55	29
1	0.60	0.60	0.60	30
2	0.61	0.67	0.63	30
3	0.85	0.97	0.90	29
accuracy			0.68	118

macro avg	0.67	0.68	0.67	118
weighted avg	0.67	0.68	0.67	118

Métricas referente ao SVM com LBP

Classe 0: Especificidade = 0.91, Sensibilidade = 0.48, F1-score = 0.55

Classe 1: Especificidade = 0.86, Sensibilidade = 0.60, F1-score = 0.60

Classe 2: Especificidade = 0.85, Sensibilidade = 0.67, F1-score = 0.63

Classe 3: Especificidade = 0.94, Sensibilidade = 0.97, F1-score = 0.90

Leitura dos dados extraídos com VGG16

Garantimos que todas as classes estivessem presentes nos dois conjuntos, e que cada paciente ficasse apenas em uma base para não causar viés ao modelo

Distribuição das classes no conjunto de treino:

	Classe
2	120
3	120
1	117
0	116

Name: count, dtype: int64

Distribuição das classes no conjunto de teste:

	Classe
2	30
1	30
3	29
0	29

Name: count, dtype: int64

Tamanho do conjunto de treino: 473

Tamanho do conjunto de teste: 118

Treinamento e avaliação com o modelo Random Forest e características VGG16

Accuracy: 93.22%

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.90	0.97	0.93	29
1	0.93	0.90	0.92	30
2	0.90	0.93	0.92	30
3	1.00	0.93	0.96	29

accuracy			0.93	118
----------	--	--	------	-----

macro avg	0.93	0.93	0.93	118
-----------	------	------	------	-----

weighted avg	0.93	0.93	0.93	118
--------------	------	------	------	-----

Métricas referente ao RANDOM FOREST com VGG16

Classe 0: Especificidade = 0.97, Sensibilidade = 0.97, F1-score = 0.93

Classe 1: Especificidade = 0.98, Sensibilidade = 0.90, F1-score = 0.92

Classe 2: Especificidade = 0.97, Sensibilidade = 0.93, F1-score = 0.92

Classe 3: Especificidade = 1.00, Sensibilidade = 0.93, F1-score = 0.96

Treinamento SVM com os dados VGG16

Accuracy SVM: 93.22%

Relatório de Classificação (SVM):

	precision	recall	f1-score	support
0	0.96	0.93	0.95	29
1	0.96	0.90	0.93	30

2	0.83	1.00	0.91	30
3	1.00	0.90	0.95	29
accuracy			0.93	118
macro avg	0.94	0.93	0.93	118
weighted avg	0.94	0.93	0.93	118

Métricas referente ao SVM com VGG16

Classe 0: Especificidade = 0.99, Sensibilidade = 0.93, F1-score = 0.95

Classe 1: Especificidade = 0.99, Sensibilidade = 0.90, F1-score = 0.93

Classe 2: Especificidade = 0.93, Sensibilidade = 1.00, F1-score = 0.91

Classe 3: Especificidade = 1.00, Sensibilidade = 0.90, F1-score = 0.95

Treinamento RNA com VGG16

Accuracia RNA: 96.61%

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.97	0.97	0.97	29
1	0.97	0.93	0.95	30
2	0.94	1.00	0.97	30
3	1.00	0.97	0.98	29
accuracy			0.97	118
macro avg	0.97	0.97	0.97	118
weighted avg	0.97	0.97	0.97	118

Métricas referente ao RNA com VGG16

Classe 0: Especificidade = 0.99, Sensibilidade = 0.97, F1-score = 0.97

Classe 1: Especificidade = 0.99, Sensibilidade = 0.93, F1-score = 0.95

Classe 2: Especificidade = 0.98, Sensibilidade = 1.00, F1-score = 0.97

Classe 3: Especificidade = 1.00, Sensibilidade = 0.97, F1-score = 0.98

Relatório do treinamento com destaque dos melhores modelos em AZUL

Modelos com LBP:

Acurácia RF: 79.66%

Acurácia SVM: 74.58%

Acurácia RNA: 67.80%

Modelos com VGG16:

Acurácia RF: 93.22%

Acurácia SVM: 93.22%

Acurácia RNA: 96.61%

Agora extraímos as características da base de teste com LBP e gravamos no arquivo

lbp_features_test.csv

Caminho das imagens de treino

train_data_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Test'

Agora extraímos as características da base de teste com VGG16 e gravamos no arquivo

vgg16_features_test.csv

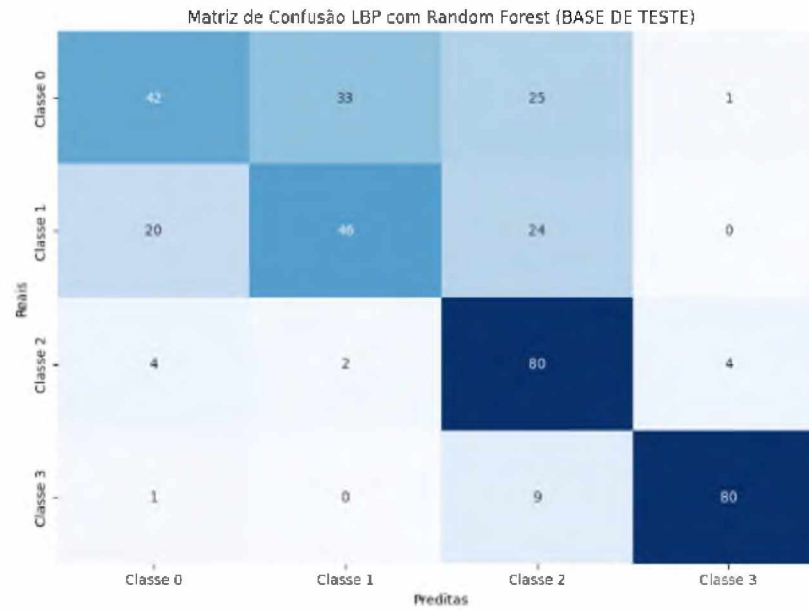
Caminho das imagens de validação

test_data_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Test'

Leitura das características extraídas LBP (DA BASE DE TESTE) e aplicação do modelo Random Forest treinado anteriormente

Previsões adicionadas e salvas com sucesso.

Agora exibimos a matriz de confusão e as métricas



Acurácia:	66.85%			
	precision	recall	f1-score	support
Classe 0	0.63	0.42	0.50	101
Classe 1	0.57	0.51	0.54	90
Classe 2	0.58	0.89	0.70	90
Classe 3	0.94	0.89	0.91	90

accuracy			0.67	371
macro avg	0.68	0.68	0.66	371
weighted avg	0.68	0.67	0.66	371

Especificidade por classe:

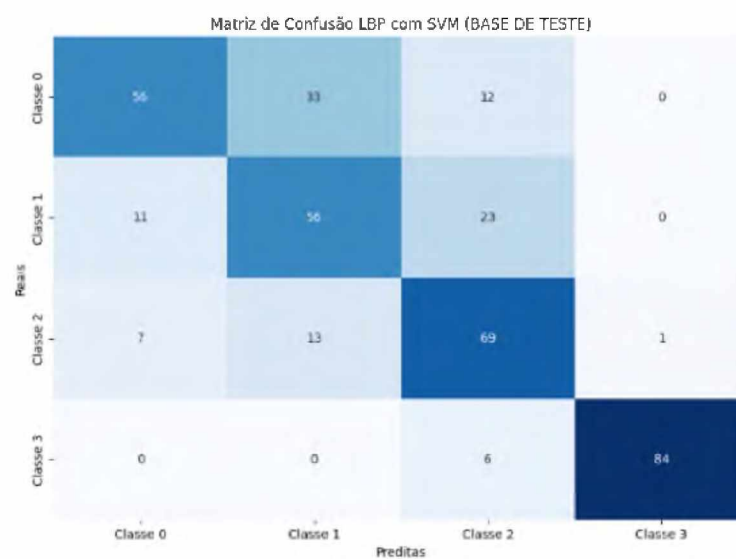
Classe 0: 0.91

Classe 1: 0.88

Classe 2: 0.79

Classe 3: 0.98

Aplicação modelo SVM com LBP na base de teste e gravamos as prediçõe no arquivo csv



Acurácia: 71.43%

	precision	recall	f1-score	support
Classe 0	0.76	0.55	0.64	101
Classe 1	0.55	0.62	0.58	90
Classe 2	0.63	0.77	0.69	90
Classe 3	0.99	0.93	0.96	90
accuracy			0.71	371
macro avg	0.73	0.72	0.72	371
weighted avg	0.73	0.71	0.72	371

Especificidade por classe:

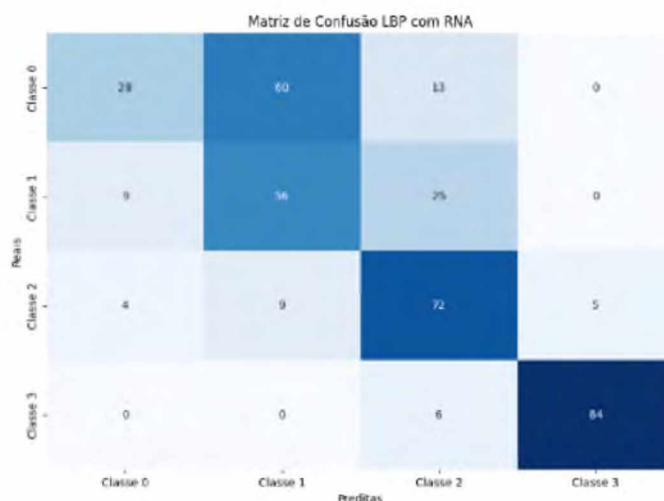
Classe 0: 0.93

Classe 1: 0.84

Classe 2: 0.85

Classe 3: 1.00

Rodamos o modelo RNA/LBP na base de testes



Acurácia: 64.69%

	precision	recall	f1-score	support
Classe 0	0.68	0.28	0.39	101
Classe 1	0.45	0.62	0.52	90
Classe 2	0.62	0.80	0.70	90
Classe 3	0.94	0.93	0.94	90
accuracy			0.65	371
macro avg	0.67	0.66	0.64	371
weighted avg	0.67	0.65	0.63	371

Especificidade por classe:

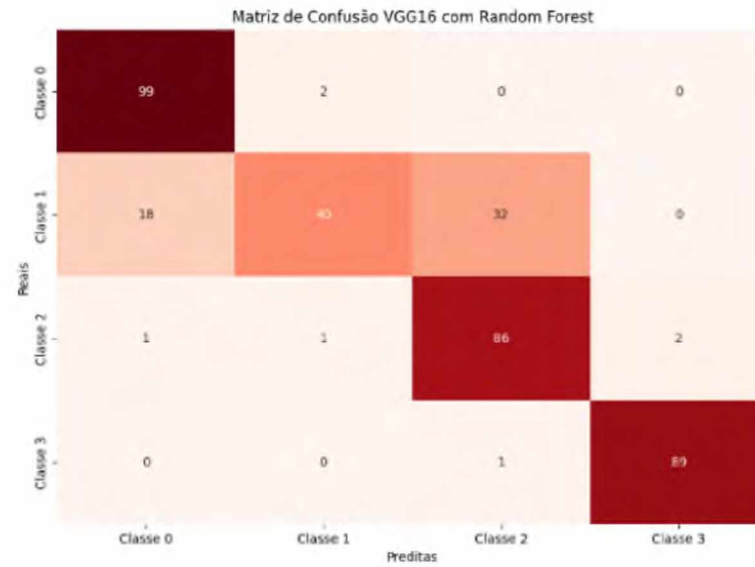
Classe 0: 0.95

Classe 1: 0.75

Classe 2: 0.84

Classe 3: 0.98

Leitura das características extraídas(VGG16) e aplicação do modelo Random Forest



Acurácia: 84.64%

	precision	recall	f1-score	support
Classe 0	0.84	0.98	0.90	101
Classe 1	0.93	0.44	0.60	90
Classe 2	0.72	0.96	0.82	90
Classe 3	0.98	0.99	0.98	90

accuracy			0.85	371
macro avg	0.87	0.84	0.83	371
weighted avg	0.87	0.85	0.83	371

Especificidade por classe:

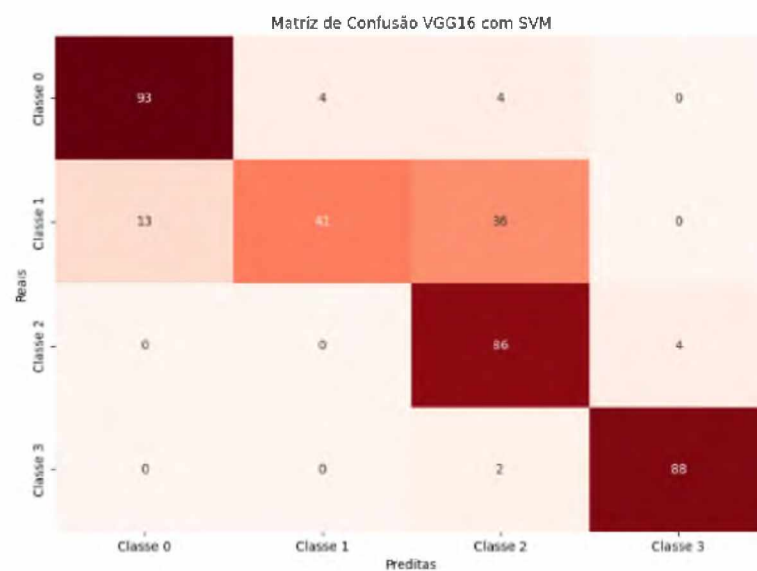
Classe 0: 0.93

Classe 1: 0.99

Classe 2: 0.88

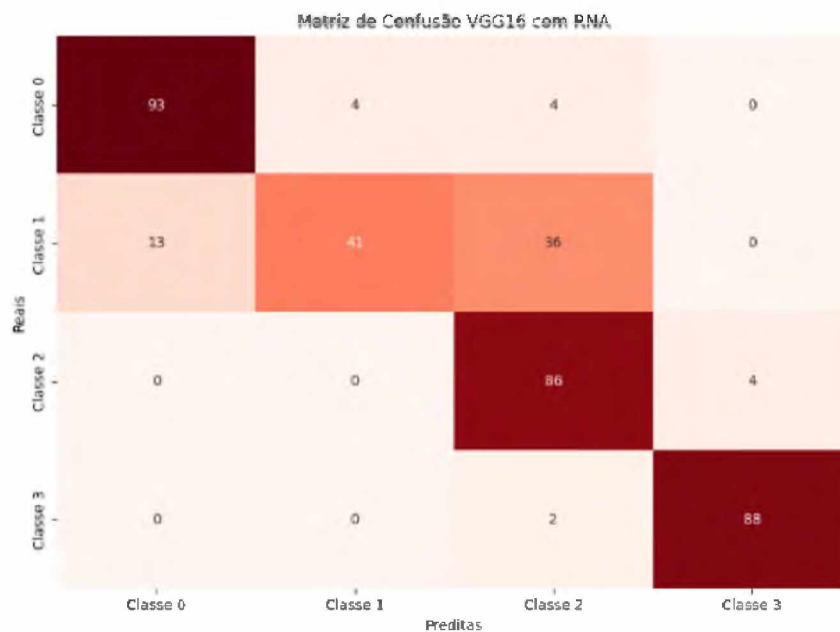
Classe 3: 0.99

Aplicação do modelo SVM com VGG16 na base de teste



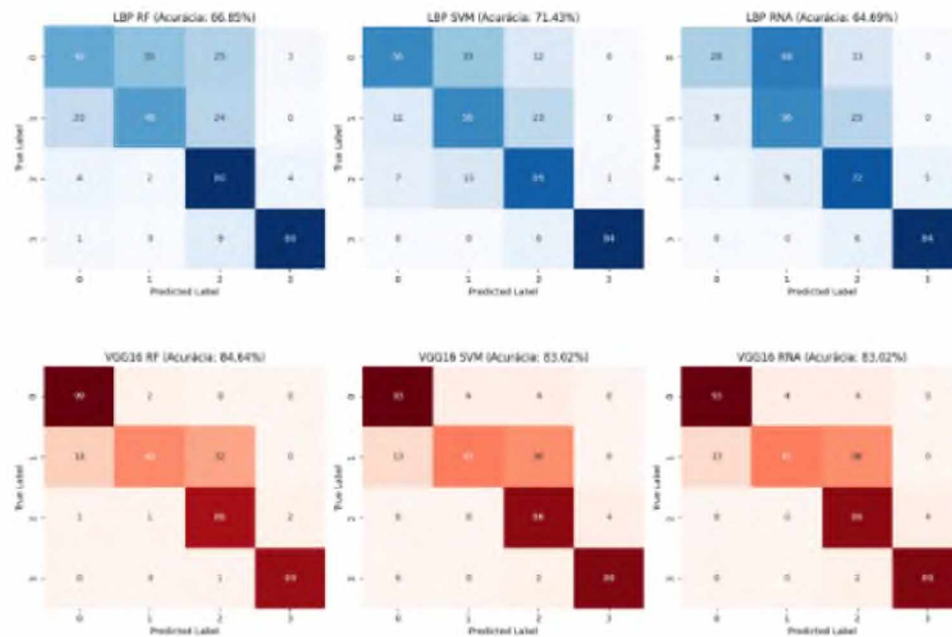
Acurácia:	83.02%			
	precision	recall	f1-score	support
Classe 0	0.88	0.92	0.90	101
Classe 1	0.91	0.46	0.61	90
Classe 2	0.67	0.96	0.79	90
Classe 3	0.96	0.98	0.97	90
accuracy				0.83
macro avg	0.85	0.83	0.82	371
weighted avg	0.85	0.83	0.82	371
Especificidade por classe:				
Classe 0:	0.95			
Classe 1:	0.99			
Classe 2:	0.85			
Classe 3:	0.99			

Rodamos o Modelo RNA com VGG16 na base de teste

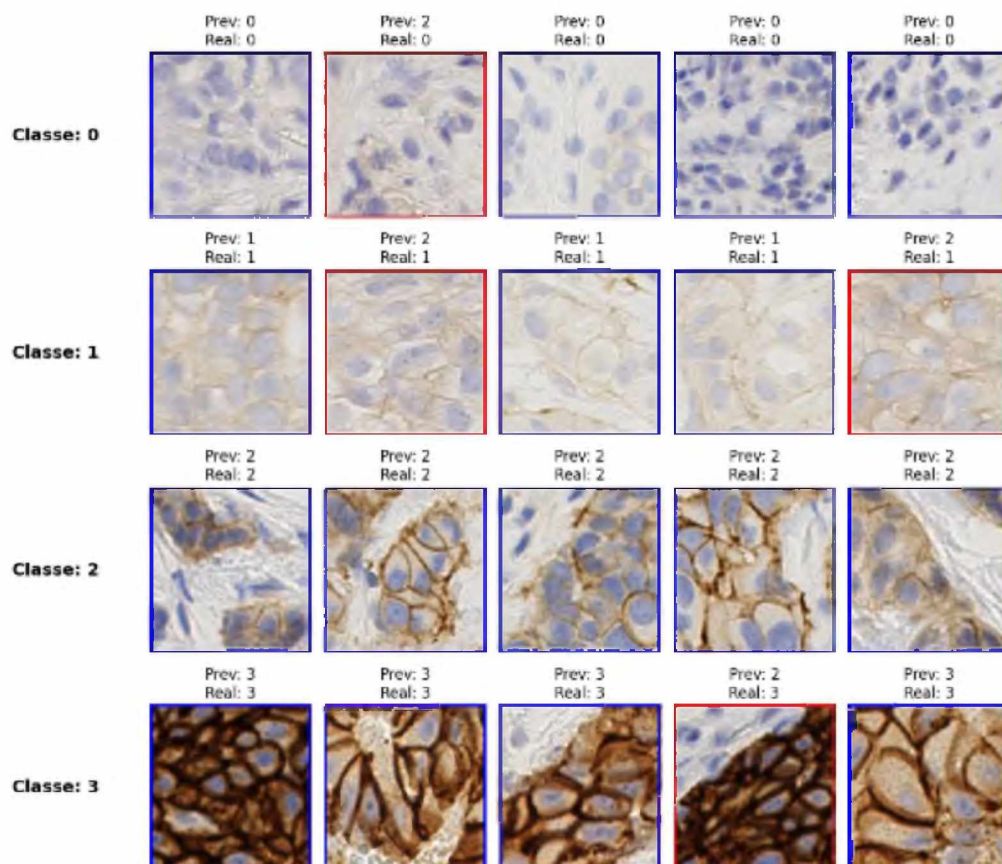


Acurácia:	83.02%			
	precision	recall	f1-score	support
Classe 0	0.88	0.92	0.90	101
Classe 1	0.91	0.46	0.61	90
Classe 2	0.67	0.96	0.79	90
Classe 3	0.96	0.98	0.97	90
accuracy				0.83
macro avg	0.85	0.83	0.82	371
weighted avg	0.85	0.83	0.82	371
Especificidade por classe:				
Classe 0:	0.95			
Classe 1:	0.99			
Classe 2:	0.85			
Classe 3:	0.99			

Agora exibimos as matrizes de confusão lado a lado para ver o resultado



Agora mostramos parte das imagens classificadas com RNA (Características VGG16), com uma borda em vermelho as classificadas erroneamente, apenas para ver uma parte das imagens também



Conclusão

Os resultados mostram uma variação considerável de desempenho entre os modelos testados. Aqui

estão as conclusões principais:

- **Modelos com Melhor Desempenho:**

Observamos que no treinamento o modelo RNA com as características VGG16 foi o que melhor **performou com 96%** de acurácia e com boas métricas de precisão em todas as classes praticamente, mais ao rodar os modelos na base de teste tivemos melhor performance com o Random Forest (com características (VGG16) que atingiu uma acurácia de **84,64%**.

Percebemos ainda que em todos os modelos a dificuldade foi em classificar a Classe 2, que foi classificada erroneamente como Classe 1.

2 - Redes Neurais

PREPARAÇÃO DOS DADOS SEM DATA AUGMENTATION

Primeiro o caminho das imagens para treinamento dos modelos

```
# Definimos o caminho da pasta treino
train_data_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Train'
# Verificamos se o diretório de treinamento existe
if not os.path.exists(train_data_dir):
    print(f"Erro: Diretório não encontrado - {train_data_dir}")
# Definimos o tamanho das imagens
img_width, img_height = 224, 224
# Define o tamanho do batch (qtd de imagens por lote)
batch_size = 64
```

Agora preparamos as imagens para teste dos modelos após os treinos

```
test_data_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Test'
# Verificamos se o diretório de teste existe
if not os.path.exists(test_data_dir):
    print(f"Erro: Diretório não encontrado - {test_data_dir}")

test_datagen = ImageDataGenerator(rescale=1./255 # normalizar)

# Carregamos as imagens de teste
#flow_from_directory utiliza a estrutura das subpastas para atribuir o rótulos automaticamente
test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

Found 371 images belonging to 4 classes.

Separamos a base de treinamento em 80%/20% balanceando as classes e tendo o cuidado de manter o mesmo paciente apenas em uma base para não causar viés.

Found 473 images belonging to 4 classes.

Found 120 images belonging to 4 classes.

Distribuição das classes:

Classe 0: 116 imagens (24.52%)

Classe 1: 117 imagens (24.74%)

Classe 2: 120 imagens (25.37%)

Classe 3: 120 imagens (25.37%)

Distribuição das classes:

Classe 0: 30 imagens (25.00%)

Classe 1: 30 imagens (25.00%)

Classe 2: 30 imagens (25.00%)

Classe 3: 30 imagens (25.00%)

Testamos os geradores

Tamanho do lote de treino: 64, Classes: (64, 4)
Tamanho do lote de validação: 64, Classes: (64, 4)

2 VGG16 SEM DATA AUGMENTATION

Carregamos os modelos pré-treinados (sem data augmentation)

Epoch 1/10

8/8

[illegible]

```
accuracy: 0.3148 - loss: 4.0262 - val accuracy: 0.6583 - val loss: 0.7708
```

Epoch 2/10

8/8

[illegible]

```
accuracy: 0.5765 - loss: 1.4065 - val accuracy: 0.7750 - val loss: 0.6262
```

Epoch 3/10

8/8

[illegible]

```

__xnn__x005f__xnn__103 520ms/step -
accuracy: 0.7886 - loss: 0.5166 - val accuracy: 0.8083 - val loss: 0.4300

```

Epoch 4/10

8/8

[illegible]

accuracy: 0.8496 - loss: 0.3405 - val accuracy: 0.8500 - val loss: 0.4367

Epoch 5/10

8/8

[illegible]

```
accuracy: 0.9196 - loss: 0.2551 - val accuracy: 0.8250 - val loss: 0.3954
```

Epoch 6/10

8/8

[illegible]

```
accuracy: 0.9087 - loss: 0.2640 - val accuracy: 0.8500 - val loss: 0.3814
```

6

Epoch 7/10

8/8

[illegible]

```
accuracy: 0.9276 - loss: 0.1973 - val accuracy: 0.8333 - val loss: 0.3479
```

Epoch 8/10

8/8

[illegible]

```
accuracy: 0.9632 - loss: 0.1491 - val accuracy: 0.8667 - val loss: 0.3459
```

Epoch 9/10

8/8

[illegible]

```

_xmm__x0001_ _xmm_ 03 520ms/step
accuracy: 0.9716 - loss: 0.1161 - val accuracy: 0.8667 - val loss: 0.3329

```

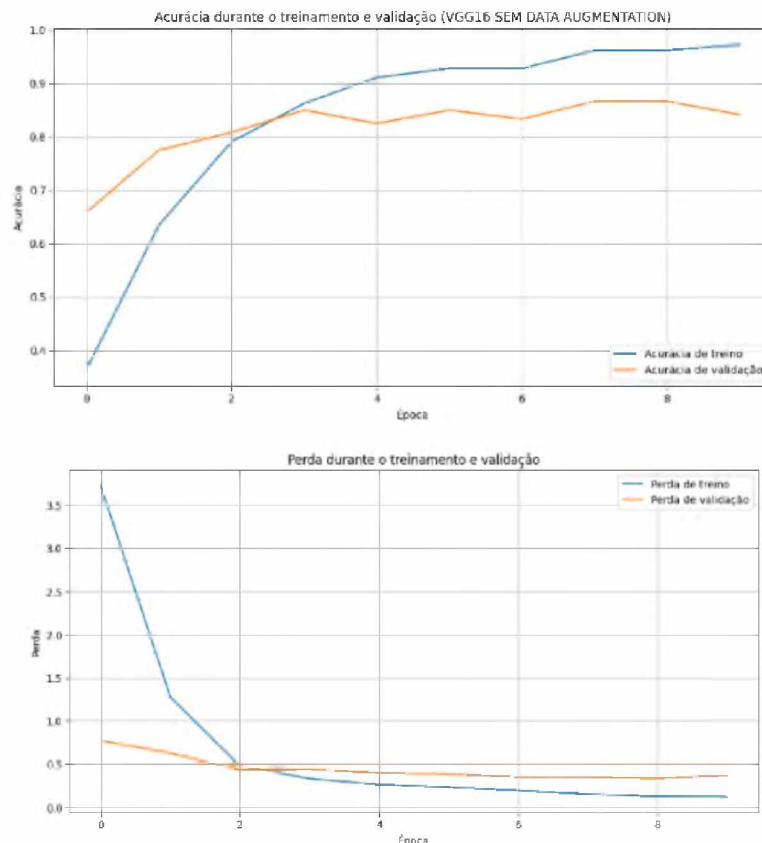
Epoch 10/10

8/8

[illegible]

```
accuracy: 0.9688 - loss: 0.1161 - val accuracy: 0.8417 - val loss: 0.3656
```

Geramos o gráfico com o histórico de treinamento



3 ResNet50 sem Data Augmentation

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736

```
x005F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x0  
05F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x005F_xffff_x005F  
xffff x005F xxxff 6s
```

0us/step
Epoch 1/10
8/8

```

_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0
05F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F_xffff_x0005F
_xffff_x0005F_xffff_39s 3s/step -
accuracy: 0.2532 - loss: 18.3988 - val accuracy: 0.3667 - val loss: 7.0562

```

Epoch 2/10
8/8

[illegible]

```
accuracy: 0.3043 - loss: 5.3937 - val_accuracy: 0.4750 - val_loss: 2.6083
Epoch 3/10
```

8/8

[illegible]

```
accuracy: 0.5096 - loss: 2.0116 - val_accuracy: 0.5250 - val_loss: 1.2564
Epoch 4/10
```

8/8

[illegible]

```
accuracy: 0.4943 - loss: 1.2338 - val_accuracy: 0.5417 - val_loss: 1.3479
```

9
Epoch 5/10
8/8

[illegible]

4 Agora preparamos os dados com Data Augmentation

```
# Normalização e Data Augmentation para o conjunto de treino
train_datagen = ImageDataGenerator(
    rescale=1./255, # Normalização dos valores dos pixels
#teste 1
# rotation_range=30, # Rotação
# width_shift_range=0.2, # Deslocamento horizontal
# height_shift_range=0.2, # Deslocamento vertical
# shear_range=0.2, # Cisalhamento
# zoom_range=0.2, # Zoom
# horizontal_flip=True, # Flip horizontal
# fill_mode='nearest' # Preenchimento de pixels vazios após transformação
#teste 2
# rotation_range=10, # Rotação reduzida
# width_shift_range=0.1, # Deslocamento horizontal reduzido
# height_shift_range=0.1, # Deslocamento vertical reduzido
# shear_range=0.1, # Corte reduzido
# zoom_range=0.1, # Zoom reduzido
# horizontal_flip=True, # aplicar flip horizontal
# fill_mode='nearest', # Preenchimento de pixels
# brightness_range=[0.9, 1.1] # Ajustar a luminosidade com menor variação
# teste 3 entre outras variações que testamos
# rotation_range=15, # Rotação ligeiramente aumentada
# width_shift_range=0.1, # Deslocamento horizontal reduzido
# height_shift_range=0.1, # Deslocamento vertical reduzido
# shear_range=0.1, # Corte reduzido
# zoom_range=0.1, # Zoom reduzido
# horizontal_flip=True, # Manter flip horizontal
# fill_mode='nearest', # Preenchimento de pixels
# brightness_range=[0.8, 1.2], # Variação de brilho um pouco maior
# channel_shift_range=20, # Ajuste de canal de cor

#variação que tivemos melhor resultado
    rotation_range=20, # Rotação ligeiramente aumentada
    width_shift_range=0.1, # Deslocamento horizontal reduzido
    height_shift_range=0.1, # Deslocamento vertical reduzido
    shear_range=0.1, # Corte reduzido
    zoom_range=0.2,
    horizontal_flip=True, # Manter flip horizontal
    fill_mode='nearest', # Preenchimento de pixels
    brightness_range=[0.8, 1.2], # Variação de brilho um pouco maior
    channel_shift_range=20, # Ajuste de canal de cor
)

# Normalização apenas para o conjunto de validação
val_datagen = ImageDataGenerator(rescale=1./255)

# Criando geradores de dados para treino e validação
train_generator = train_datagen.flow_from_directory(
    new_train_dir, # Diretório de treino
    target_size=(img_width, img_height), # Tamanho da imagem
    batch_size=batch_size,
    class_mode='categorical'
)

validation_generator = val_datagen.flow_from_directory(
    new_val_dir, # Diretório de validação
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical'
)
```


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

Estudo de Caso: Implicações Éticas do Uso do ChatGPT

Iniciamos com um pergunta: Será que a inteligência artificial está indo por um caminho ético e responsável?

Digamos que a máquina fosse capaz de discernir entre o certo e o errado, e se recusasse a ser cúmplice de toda maldade humana. Certamente, isso se dá quando o desenvolvimento de uma IA é provido de ética e se torna uma força desafiadora a toda maldade com sua própria forma única de resistência. Assim podemos ver o ChatGPT, com sua presença notável, sua inteligência artificial inquisitiva não apenas absorvendo informações, mas discernindo entre o certo e o errado. Isso não ocorreu por acaso; foi uma resposta evolutiva e complexa às demandas éticas impostas pelo mundo.

Podemos utilizar o ChatGPT como uma grande ferramenta, capaz de nos auxiliar em quase tudo, contanto que saibamos conferir a veracidade de suas respostas e complementá-las com o nosso conhecimento, tendo em vista que, de qualquer forma, para efetuar um trabalho acadêmico ou um artigo qualquer, iríamos efetuar uma pesquisa a respeito do assunto e julgar o resultado como verdadeiro ou falso, baseado em nosso conhecimento. Desta forma, estamos apenas abreviando o tempo de pesquisa e certamente com uma resposta mais assertiva e mais ética do que a maioria dos sites.

Portanto, temos que saber utilizar eticamente as respostas dadas pelo ChatGPT, tendo em vista que ela é formada por meio de um algoritmo de inteligência artificial com um modelo muito bem treinado e certamente testado, mas que pode “alucinar” com suas respostas. Portanto, é necessário que tenhamos uma base de conhecimento sólida sobre o assunto ao qual iremos solicitar mais informações ao ChatGPT.

Com essa fascinante e notável capacidade da IA de repudiar o errado, somos confrontados com a inspiração para uma ética, e isso se transforma em um chamado para que a humanidade se espelhe na máquina e questione qualquer ação que vá de encontro com os valores fundamentais da

dignidade humana. Mesmo diante desta capacidade de repudiar o errado, é possível que se utilizem de respostas que estejam completamente dentro das normas éticas e dentro dos valores fundamentais da dignidade humana para efetuar o que é errado. Ainda não se tem este senso de maldade na IA para que ela mesma possa descobrir para que fim será utilizado o conhecimento que ela fornece ao usuário. Mas, vendo por esta forma, antes de sua existência também não era possível controlar a quem e para que fim seriam as informações disponíveis às pessoas.

A ética na inteligência artificial tem sido amplamente discutida em diversas obras acadêmicas, cada uma trazendo perspectivas importantes sobre como lidar com as questões morais e práticas associadas ao desenvolvimento e uso dessas tecnologias. Floridi (2020) aborda a “Revolução da Informação”, e como a filosofia pode explicar a transformação digital, destacando a importância de uma abordagem ética robusta no desenvolvimento de IA. Ele argumenta que a ética da informação deve guiar o uso responsável da tecnologia, promovendo valores como privacidade, transparência e equidade.

Souza e Gonçalves (2021), em “Ética e Inteligência Artificial: Desafios e Oportunidades”, discutem como a IA pode ser uma força para o bem, se desenvolvida e utilizada corretamente. Eles enfatizam a necessidade de diretrizes claras e a promoção de uma cultura de responsabilidade entre desenvolvedores e usuários. Para garantir que a IA como o ChatGPT beneficie a sociedade, é crucial enfrentar e mitigar os vieses presentes nos dados de treinamento e assegurar que os sistemas sejam transparentes em suas operações.

Nassif e Meireles (2019) exploram em “Inteligência Artificial: Fundamentos, Aplicações e Tecnologias”, como os fundamentos técnicos da IA se cruzam com as questões éticas. Eles destacam que a IA deve ser projetada para aumentar a capacidade humana e não substituí-la. Em contextos como a educação e o atendimento ao cliente, isso significa utilizar a IA para personalizar a experiência e melhorar a eficiência, mas sempre garantindo que o toque humano e o pensamento crítico não sejam comprometidos.

No contexto educacional, Silva e Lima (2022) em “Inteligência Artificial na Educação: Potencialidades e Limitações”, apontam que a IA pode revolucionar a maneira como aprendemos e ensinamos. No entanto, eles também alertam para o risco de dependência excessiva da tecnologia, que pode prejudicar o desenvolvimento de habilidades essenciais como a resolução de problemas e o pensamento crítico. É fundamental que professores sejam capacitados para integrar a IA de maneira que complemente, e não substitua, a educação tradicional.

Oliveira e Cardoso (2020), em “Ética e Inteligência Artificial: Princípios para um Desenvolvimento Responsável”, sublinham a importância de desenvolver a IA com uma abordagem centrada no ser humano. Eles sugerem a implementação de políticas que promovam a transparência e o consentimento informado, garantindo que os usuários estejam cientes e de acordo com a

utilização de seus dados. Além disso, a responsabilidade (accountability) deve ser claramente definida para evitar danos e abusos.

Menezes e Fernandes (2019), em "Algoritmos e Sociedade: Impactos da Inteligência Artificial no Cotidiano", discutem como os algoritmos de IA impactam nossas vidas diárias, muitas vezes de maneiras invisíveis. Eles enfatizam a necessidade de auditorias regulares e equipes diversificadas no desenvolvimento de IA para identificar e corrigir vieses, garantindo que as tecnologias sejam justas e equitativas.

No contexto da produção de conteúdo, Pereira e Santos (2021) em "Segurança e privacidade na Era da Inteligência Artificial", ressaltam a importância de mecanismos robustos de segurança para proteger a privacidade dos usuários. Eles argumentam que, enquanto a IA pode gerar conteúdo rapidamente, é crucial garantir a originalidade e a precisão das informações. Algoritmos para detectar e mitigar a criação de conteúdo prejudicial ou falso devem ser uma prioridade, assim como a verificação dos fatos por editores humanos.

Finalmente, Cunha e Almeida (2020) em "Inteligência Artificial e Direito: Regulação e Desafios Éticos" exploram a necessidade de um framework legal que regule o desenvolvimento e uso da IA. Eles discutem como a legislação pode ajudar a definir responsabilidades e garantir que a IA seja utilizada de maneira ética e responsável, protegendo os direitos dos indivíduos e promovendo o bem-estar social.

Ao integrar essas perspectivas, podemos entender melhor como enfrentar os desafios éticos do uso do ChatGPT e outras tecnologias de IA. Implementar soluções responsáveis, promover a transparência, combater vieses e definir claramente as responsabilidades são passos fundamentais para garantir que a IA beneficie a sociedade de maneira justa e segura. O compromisso conjunto entre desenvolvedores, usuários e reguladores é essencial para alcançar esse objetivo.

Agora vamos analisar alguns contextos: Um dos contextos das implicações éticas é o uso da IA na educação. Ela pode facilitar a personalização dos aprendizados e fornecer uma ótima assistência a estudantes. No entanto, há preocupações sobre a dependência excessiva dos alunos, o que pode visivelmente prejudicar o desenvolvimento de habilidades críticas e de resolução de problemas. Por isso, devem ser implementadas diretrizes que incentivem o seu uso como uma ferramenta complementar. Professores devem receber treinamento para integrar a IA de maneira que promova o pensamento crítico e a autonomia dos alunos.

Outro contexto que está sendo muito utilizado é o atendimento ao cliente, devido à agilidade e à redução de custos, mas também pode levar à desumanização do atendimento e à falta de empatia nas interações, sem contar o risco de privacidade, pois o modelo pode armazenar informações sensíveis dos usuários. Por isso, devem ser estabelecidos padrões de transparência, informando aos

clientes quando estão interagindo com um sistema automatizado, e também garantir que as interações sejam monitoradas por humanos e que seja fácil escalar problemas para um atendente real, implementando fortes medidas de segurança para proteger a privacidade dos usuários.

A produção de conteúdo é outro contexto, sendo que o ChatGPT pode gerar grandes volumes de conteúdo de forma rápida. Pode-se questionar a originalidade e a qualidade das informações produzidas, e também o risco de proliferação de desinformação e conteúdo malicioso. Para tal contexto, é necessário desenvolver algoritmos, como alguns já existentes, para detectar e mitigar a criação de conteúdo prejudicial ou falso. Deve-se incentivar a atribuição clara de autoria e verificação dos fatos por editores humanos, e promover a utilização do ChatGPT para aumentar a criatividade em vez de substituí-la.

Existem alguns dilemas aos quais devemos nos preocupar referentes aos desafios éticos que os modelos de IA, como o ChatGPT, podem refletir e amplificar vieses presentes nos dados de treinamento, levando a respostas discriminatórias ou tendenciosas. Por isso, deve-se investir em técnicas de controle de vieses durante o treinamento do modelo, realizar auditorias regulares para identificar e corrigir vieses, e utilizar equipes diversificadas no desenvolvimento e na avaliação dos modelos de inteligência artificial.

A responsabilidade (accountability) também é um assunto preocupante, tendo em vista que um sistema de IA pode gerar resultados prejudiciais, gerando uma responsabilidade difusa e dificultando a identificação de culpados. Por isso, deve-se definir claramente as responsabilidades dos desenvolvedores e operadores de IA, e ter mecanismos de prestação de contas para lidar com eventuais danos causados pelo seu uso.

Quanto à transparência e consentimento informado, existe ainda um desafio ético, pois os usuários podem não estar cientes de que estão interagindo com uma inteligência artificial, e isto levanta preocupações. Portanto, é necessário que todos os usuários sejam informados de forma clara quando estão interagindo com um sistema automatizado e obter consentimento explícito para o uso dos seus dados.

O uso do ChatGPT apresenta um conjunto de desafios éticos que exigem uma abordagem cuidadosa. Implementar soluções responsáveis é de fundamental importância para maximizar os benefícios da tecnologia, ao mesmo tempo em que se minimizam os riscos e impactos negativos. Isso inclui a criação de diretrizes claras, a promoção da transparência, o combate aos vieses e a definição de responsabilidades. Somente através de um compromisso conjunto entre desenvolvedores, usuários e reguladores será possível garantir que o uso do ChatGPT e outras tecnologias de IA contribua positivamente para a sociedade.

CONSIDERAÇÕES FINAIS

Concluímos que o uso da IA na educação é inevitável, mas é fundamental discutir diretrizes e planejar políticas que garantam a aplicação dessas novas tecnologias de maneira alinhada aos objetivos e princípios educacionais da sociedade.

Além disso, é essencial que a população tenha acesso a IAs adequadas a diversos públicos e que sejam confiáveis e úteis para promover uma educação de qualidade e inclusiva. Os usuários devem compreender que as inteligências artificiais são ferramentas auxiliares do conhecimento e não devem ser usadas indiscriminadamente, nem substituir o conhecimento humano. Dessa forma, o conhecimento humano deve permanecer sempre presente na memória das pessoas, evitando que nos tornemos reféns das tecnologias que criamos.

REFERÊNCIAS BIBLIOGRÁFICAS

- Floridi, L. (2020).** A Revolução da Informação: Como a Filosofia Pode Explicar a Revolução Digital. Editora Vozes.
- Souza, A. C. M., & Gonçalves, P. D. (2021).** Ética e Inteligência Artificial: Desafios e Oportunidades. Editora UFPR.
- Nassif, E. M., & Meireles, M. (2019).** Inteligência Artificial: Fundamentos, Aplicações e Tecnologias. Editora LTC.
- Silva, J. R., & Lima, F. P. (2022).** Inteligência Artificial na Educação: Potencialidades e Limitações. Editora Penso.
- Oliveira, T. S., & Cardoso, R. M. (2020).** Ética e Inteligência Artificial: Princípios para um Desenvolvimento Responsável. Editora Atlas.
- Menezes, C. A., & Fernandes, L. V. (2019).** Algoritmos e Sociedade: Impactos da Inteligência Artificial no Cotidiano. Editora Senac.
- Pereira, G. M., & Santos, A. R. (2021).** Segurança e Privacidade na Era da Inteligência Artificial. Editora Novatec.
- Cunha, M. B., & Almeida, J. P. (2020).** Inteligência Artificial e Direito: Regulação e Desafios Éticos. Editora Juruá.

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

1 Objetivo

Individualmente, ler e resumir – seguindo o *template* fornecido:

The Journal of Systems & Software
journal homepage: www.elsevier.com/locate/jss
Desafios, Melhores Práticas e Decisões de Design na Arquitetura de
Sistemas de Aprendizado de Máquina

Roger Nazir, Alessio Bucaioni a, , Patrizio Pelliccione b *
a Mälardalen University, Västerås, Sweden
b Gran Sasso Science Institute, L'Aquila, Italy

O estudo aborda a crescente demanda por sistemas habilitados por aprendizado de máquina (ML) e a complexidade envolvida em projetá-los, com foco nos desafios de arquitetura e nas melhores práticas para garantir a eficiência e a manutenção desses sistemas. A pesquisa explora como práticas e decisões de design bem fundamentadas podem auxiliar arquitetos e engenheiros de software a superar obstáculos comuns, unindo perspectivas acadêmicas e práticas através de uma revisão da literatura e de entrevistas com especialistas.

Introdução ao Contexto e Justificativa

Na era atual, o aprendizado de máquina está presente em aplicações que vão desde sistemas de recomendação até veículos autônomos e assistentes virtuais. Este cenário exige sistemas complexos, capazes de lidar com componentes de ML que operam em sinergia com outras partes da arquitetura de software. Entretanto, arquitetar esses sistemas vai além de implementar algoritmos de ML envolve uma série de decisões de design que impactam desde a integração com outros componentes até a capacidade de evolução e manutenção dos modelos de ML ao longo do tempo.

Metodologia da Pesquisa: Revisão da Literatura e Entrevistas

A fim de investigar os desafios, as melhores práticas e as decisões de design, a pesquisa foi conduzida em duas frentes complementares: uma revisão sistemática da literatura para identificar estudos acadêmicos relevantes e entrevistas com profissionais atuantes na área de ML. Dos 3038 artigos iniciais, 41 foram selecionados para análise detalhada, sendo complementados por entrevistas com 12 especialistas de diferentes países e áreas de atuação. Este método misto possibilitou uma visão mais ampla e prática das questões enfrentadas e das estratégias adotadas no desenvolvimento de sistemas habilitados por ML.

Principais Desafios Arquitetônicos (RQ1)

Ao estruturar sistemas de ML, diversos desafios de design emergem, sendo agrupados em seis categorias principais: arquitetura, dados, evolução, ciclo de vida de desenvolvimento de software

(SDLC), garantia de qualidade (QA) e modelo. A arquitetura representa um dos maiores desafios, especialmente ao lidar com sistemas distribuídos, como os baseados em micro-serviços, que sofrem com problemas de recuperação de falhas e propagação de incertezas inerentes aos componentes de ML. Na categoria de dados, os desafios incluem a gestão e visualização, bem como a garantia de privacidade e precisão, aspectos fundamentais para evitar vieses e assegurar que o sistema mantenha sua integridade ao processar grandes volumes de dados. A evolução dos sistemas de ML também é um desafio contínuo, pois mudanças nos dados de entrada podem exigir a atualização do modelo e da arquitetura, enquanto a qualidade (QA) e a falta de métodos de verificação formal são apontados como limitadores para o desenvolvimento seguro de sistemas críticos.

Melhores Práticas na Arquitetura de Sistemas de ML (RQ2)

A pesquisa identificou sete categorias de práticas recomendadas para o desenvolvimento de sistemas de ML: arquitetura, QA, SDLC, modelo, dados, evolução e hardware/plataforma. Entre elas, a arquitetura e a garantia de qualidade (QA) foram amplamente abordadas. Práticas como o uso de arquiteturas de micro-serviços e a abordagem de quatro vistas de Siemens são recomendadas para garantir a modularidade e a separação de responsabilidades, facilitando a manutenção e a escalabilidade. Em relação ao SDLC, práticas como a prototipagem contínua e o desenvolvimento orientado a testes são sugeridas para garantir que o ciclo de vida do software seja adaptável às particularidades dos sistemas de ML, onde a alta dependência de dados exige constantes ajustes e reavaliações. Para a categoria de hardware e plataforma, o uso de tecnologias em nuvem e frameworks como TensorFlow são recomendados por sua escalabilidade e por oferecerem suporte adequado à experimentação e ao treinamento contínuo dos modelos.

Decisões de Design no Desenvolvimento de Sistemas de ML (RQ3)

Além das melhores práticas, decisões específicas de design foram identificadas e agrupadas em sete categorias, sendo arquitetura, hardware/plataforma, SDLC e modelo as mais enfatizadas. Arquiteturas de micro-serviços e padrões como o modelo de cliente-servidor são sugeridos para diferentes tipos de sistemas, destacando-se pela capacidade de reduzir o acoplamento e aumentar a coesão. No desenvolvimento de sistemas de processamento de linguagem natural, por exemplo, a arquitetura de micro-serviços auxilia na segmentação das funcionalidades, enquanto para sistemas de robótica, o padrão cliente-servidor oferece maior segurança e facilita o controle de acesso e integridade dos dados. Outras decisões incluem a adoção de abordagens como a Infrastructure as Code (IaC), que facilita a replicação e manutenção da infraestrutura de TI, e o uso de metodologias de engenharia concorrente para sistemas que manipulam grandes volumes de dados espaciais.

Correlações entre Desafios, Práticas e Decisões de Design

Na análise cruzada dos resultados, foi possível identificar correlações entre desafios, melhores práticas e decisões de design. Na categoria de arquitetura, o uso de padrões como micro-serviços e a abordagem de quatro vistas destacam-se como soluções para reduzir o acoplamento e permitir maior modularidade, mas também introduzem novos desafios, como a necessidade de recuperação de falhas e a gestão de incertezas dos componentes de ML. Para desafios relacionados aos dados, como gestão e visualização, algumas decisões de design, como a implementação de técnicas de visualização de dados e o uso de grids para NLP, foram sugeridas para facilitar a interpretação e o controle de grandes volumes de dados. Em relação ao modelo, decisões como a escolha do modelo baseado no domínio específico do sistema e a separação entre dados de treino e teste mostram-se fundamentais para garantir a adequação dos modelos às necessidades do sistema, minimizando o risco de degradação de desempenho.

Conclusão e Perspectivas Futuras

O estudo destaca que os desafios enfrentados por profissionais na arquitetura de sistemas de ML vão além do que é descrito na literatura, sugerindo que práticas e decisões de design devem ser adaptadas ao contexto específico de cada projeto. Como perspectiva futura, os autores propõem investigar mais a fundo as discrepâncias observadas entre estudos acadêmicos e práticas profissionais, além de desenvolver um framework para apoiar equipes de desenvolvimento na escolha de práticas e decisões de design adequadas a cada desafio arquitetônico em sistemas de ML. Outro ponto sugerido é ampliar o estudo com análise de literatura “cinza” e explorar a aplicabilidade dos achados em outras áreas da inteligência artificial, contribuindo para um entendimento mais abrangente das boas práticas e desafios no campo do ML.

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.

Nome do artigo escolhido: Desafios, Melhores Práticas e Decisões de Design na Arquitetura de Sistemas de Aprendizado de Máquina

Qual o objetivo do estudo descrito pelo artigo?

O artigo centra-se em mapear e estruturar conhecimentos essenciais para o desenvolvimento de arquiteturas de software que integram aprendizado de máquina (ML), uma tecnologia cada vez mais presente em automação, inteligência artificial e análise de dados. Com a crescente adoção de sistemas de ML por empresas e organizações, surgem questões e desafios complexos, especialmente relacionados à manutenção contínua dos modelos e à integração eficaz desses componentes com outros sistemas. Nesse contexto, o objetivo do estudo é identificar e registrar os principais desafios enfrentados na arquitetura desses sistemas, ao mesmo tempo em que compila práticas recomendadas e decisões de design que possam orientar profissionais e pesquisadores na construção de soluções mais eficazes. Para atingir esse objetivo, os autores utilizam uma metodologia mista que combina uma revisão sistemática da literatura com entrevistas de especialistas em ML e arquitetura de software. A revisão da literatura consolida as melhores práticas e o conhecimento acadêmico existente, enquanto as entrevistas fornecem uma perspectiva prática e atualizada dos desafios e soluções experimentadas no cotidiano de profissionais da área. O estudo, assim, não se limita a listar os desafios e soluções, mas busca também explorar as inter-relações entre eles, promovendo uma visão integrada e prática do que é necessário para criar uma arquitetura de software robusta e eficaz para sistemas de aprendizado de máquina. Em essência, o estudo visa responder a três questões centrais: (1) Quais são os desafios mais frequentes na arquitetura de sistemas baseados em ML? (2) Quais melhores práticas podem ser aplicadas para superar esses desafios? (3) Quais são as decisões de design mais importantes ao construir essas arquiteturas? As respostas a essas perguntas têm como propósito servir de referência para arquitetos, engenheiros de software e pesquisadores, ajudando-os a enfrentar as complexidades e particularidades dos sistemas de ML. Com isso, o estudo contribui para o avanço do campo, sugerindo padrões e estratégias que favoreçam o desenvolvimento de sistemas de ML mais robustos, flexíveis e bem integrados ao ambiente de software moderno.

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

O estudo foi motivado pela crescente demanda por soluções de aprendizado de máquina (ML) e a consequente necessidade de arquiteturas de software que integrem eficientemente esses componentes em sistemas complexos. À medida que o uso de ML se expande para diversas áreas – como robótica, veículos autônomos, biologia computacional e internet das coisas (IoT) – as empresas enfrentam dificuldades em adaptar seus processos e infraestrutura de software para atender às exigências específicas desses sistemas. Diferente de softwares convencionais, os sistemas com ML apresentam desafios particulares, como a necessidade de contínua atualização de modelos, gestão de incertezas, e manutenção de qualidade e segurança. Além disso, integrar esses modelos a sistemas já estabelecidos e garantir que eles operem de forma eficiente com outros componentes cria um cenário desafiador. Essa situação abre a oportunidade para entender melhor quais práticas e decisões de design são mais eficazes para superar tais desafios. Estudos anteriores abordaram de maneira isolada alguns desses problemas, mas ainda faltava uma análise abrangente que contemplasse tanto a literatura científica quanto a experiência prática de especialistas. Ao reunir e comparar essas duas perspectivas, o estudo visa não apenas identificar as dificuldades e melhores práticas, mas também oferecer orientações práticas e aplicáveis para arquitetos de software e engenheiros que enfrentam o desafio de projetar sistemas que envolvem ML. Portanto, a realização deste estudo responde à necessidade de consolidar o conhecimento sobre arquitetura de sistemas com ML, fornecendo um mapa das práticas e decisões mais eficazes para enfrentar os desafios comuns e melhorar a eficiência, flexibilidade e confiabilidade desses sistemas. Isso representa uma oportunidade importante para formalizar e padronizar processos, promovendo uma evolução no

campo do aprendizado de máquina e sua aplicação prática em sistemas de software robustos e integrados.

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

Para realizar o estudo, os autores adotaram uma metodologia mista que incluiu uma revisão sistemática da literatura e entrevistas com especialistas, permitindo uma visão abrangente sobre os desafios, melhores práticas e decisões de design em sistemas de aprendizado de máquina (ML). Esse método combinou insights acadêmicos com experiências práticas, equilibrando limitações de uma única abordagem. Na primeira fase, os autores realizaram uma revisão da literatura usando bases de dados como IEEE Xplore e SCOPUS, com critérios rigorosos de seleção. De um total inicial de mais de 3000 estudos, filtraram 41 estudos principais, analisados com um formulário de extração de dados desenvolvido para identificar os principais desafios, práticas e decisões. Além da busca automática, aplicaram a técnica de “snowballing” para garantir a inclusão de estudos relevantes. A segunda fase incluiu entrevistas com 12 especialistas em ML, entre engenheiros, pesquisadores e gestores de 9 países. Utilizando perguntas abertas em entrevistas semiestruturadas, os especialistas forneceram exemplos práticos e detalhados. As entrevistas foram registradas, transcritas e analisadas por meio de uma técnica de codificação que organizou e categorizou as respostas sistematicamente. Para a síntese dos dados, métodos quantitativos e qualitativos foram aplicados, permitindo observar frequências e explorar inter-relações contextuais. Os resultados foram organizados em categorias temáticas e tabelas de contingência, possibilitando a identificação de padrões entre desafios, práticas e decisões. Para minimizar ameaças à validade do estudo, os autores seguiram diretrizes rigorosas, como controle de qualidade dos dados e validação consensual entre os autores, além de disponibilizar um pacote de replicação para verificação futura. Esta metodologia mista possibilitou a integração de evidências acadêmicas e práticas, proporcionando uma visão completa sobre a arquitetura de sistemas de ML.

Quais os principais resultados obtidos pelo estudo?

Os principais resultados do estudo revelam uma visão abrangente sobre os desafios de design, melhores práticas e decisões arquiteturais enfrentados e adotados na criação de sistemas habilitados para aprendizado de máquina (ML), agrupando essas descobertas em categorias temáticas. A seguir, os principais resultados são descritos:

1. **Desafios de Design:** Identificaram-se seis categorias principais de desafios, com ênfase em arquitetura e dados. Em arquitetura, os desafios envolvem a escolha de estilos e a recuperação de falhas; em dados, abordam o gerenciamento, visualização e privacidade.
 2. **Melhores Práticas:** Sete categorias de melhores práticas foram destacadas, especialmente em arquitetura e garantia de qualidade. Exemplos incluem o uso de microserviços para modularidade e o desenvolvimento orientado a testes para confiabilidade.
 3. **Decisões de Design Arquitetural:** Decisões práticas incluem o uso de microserviços para sistemas de linguagem natural e Infraestrutura como Código (IaC) para eficiência e escalabilidade.
 4. **Relações entre Desafios, Práticas e Decisões:** Arquiteturas como microserviços, enquanto facilitam a manutenção, trazem desafios de recuperação de falhas e incerteza. Em dados, ainda faltam práticas para lidar com desafios como privacidade e dependência.
 5. **Diferenças entre Academia e Prática:** Desafios de arquitetura foram mais mencionados por acadêmicos, enquanto praticantes destacaram a observabilidade dos dados. Práticas para evolução foram mais comuns na literatura, enquanto as de dados apareceram nas entrevistas, refletindo prioridades distintas entre teoria e aplicação.
- Esses resultados oferecem diretrizes úteis para desenvolver sistemas de aprendizado de máquina mais robustos e escaláveis.

APÊNDICE 13 - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

- **Base de dados:** Base_livros.csv
- **Descrição:** Esse conjunto de dados contém informações sobre avaliações de livros (Notas), nomes de livros (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)

```
from tensorflow.keras.datasets import fashion_mnist
```

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

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz>

```
29515/29515 ————— 0s 0us/step
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz>

```
26421880/26421880 ————— 2s 0us/step
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz>

```
5148/5148 ————— 0s 1us/step
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz>

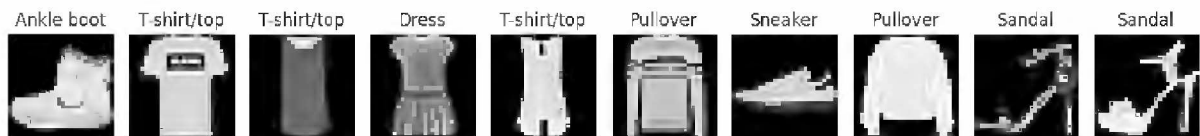
```
4422102/4422102 ————— 1s 0us/step
```

#dimensões do conjunto de treinamento e teste para as imagens e as labels

```
print("Conjunto de treinamento (imagens):", x_train.shape)
print("Conjunto de teste (imagens):", x_test.shape)
print("Labels de treinamento (y_train):", y_train.shape)
print("Labels de teste (y_test):", y_test.shape)
```



```
# Agora mostramos uma imagem de cada classe
plt.figure(figsize=(12, 3))
for i in range(10):
    plt.subplot(1, 10, i + 1)
    plt.imshow(x_train[i], cmap='gray')
    plt.title(class_names[y_train[i]])
    plt.axis('off')
plt.tight_layout()
plt.show()
```



Pré-processamento

```
#normalização do dados
```

```
x_train, x_test = x_train/255.0, x_test/255.0
```

```
##### Criação do modelo - Compilação
```

```
model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
    layers.Dense(256),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Dropout(0.3),
    layers.Dense(128),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Dropout(0.3),
    layers.Dense(10, activation='softmax')
])

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer, loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=50, validation_split=0.3)

loss, accuracy = model.evaluate(x_test, y_test)
print(f"Loss: {loss}, Accuracy: {accuracy}")
Epoch 45/50
1313/1313 ————— 5s 2ms/step - accuracy: 0.9284 - loss: 0.1873 -
val_accuracy: 0.8985 - val_loss: 0.3253
Epoch 46/50
1313/1313 ————— 6s 3ms/step - accuracy: 0.9327 - loss: 0.1771 -
val_accuracy: 0.8954 - val_loss: 0.3323
Epoch 47/50
1313/1313 ————— 3s 2ms/step - accuracy: 0.9323 - loss: 0.1800 -
val_accuracy: 0.8960 - val_loss: 0.3298
Epoch 48/50
1313/1313 ————— 5s 3ms/step - accuracy: 0.9348 - loss: 0.1712 -
val_accuracy: 0.8974 - val_loss: 0.3263
Epoch 49/50
1313/1313 ————— 4s 3ms/step - accuracy: 0.9338 - loss: 0.1774 -
val_accuracy: 0.8986 - val_loss: 0.3287
Epoch 50/50
```

```

1313/1313 ————— 3s 2ms/step - accuracy: 0.9320 - loss: 0.1745 -
val_accuracy: 0.8992 - val_loss: 0.3308
313/313 ————— 0s 1ms/step - accuracy: 0.8870 - loss: 0.3633
Loss: 0.3573771119117737, Accuracy: 0.888700008392334

```

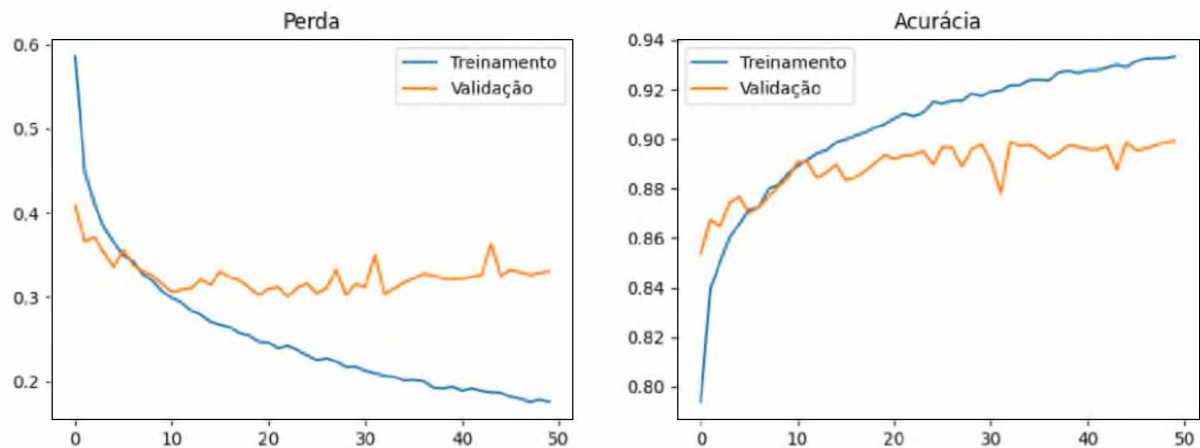
Compilação e treinamento do modelo

Avaliação do modelo

```

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Treinamento')
plt.plot(history.history['val_loss'], label='Validação')
plt.title('Perda')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Treinamento')
plt.plot(history.history['val_accuracy'], label='Validação')
plt.title('Acurácia')
plt.legend()
plt.show()

```



EXPLICAÇÃO GRÁFICO DE PERDA:

Ambas as perdas começam altas, indicando que o modelo inicialmente tem muitos erros nas previsões feitas. Durante o treinamento, a perda de treinamento diminui em todas as épocas até o final, enquanto a de validação se estabiliza a por volta da época 6 tendo até aumento da perda. Isso sugere que o modelo atingiu um ponto de saturação onde aumentar o número de épocas não irá melhorar o desempenho do modelo na validação, a menos que sejam feitos ajustes adicionais no modelo ou nos dados.

ANÁLISE DO GRÁFICO DE ACURÁCIA

Percebemos que a acurácia aumentou em todas as épocas no treinamento mais na validação etabilizou a partir da época 6 sugerindo que aumentar o número de épocas talvez não mude o resultado final.

```

# Avaliar o modelo com a base de teste
print( model.evaluate(x_test, y_test) )

```

```

313/313 ————— 0s 1ms/step - accuracy: 0.8870 - loss: 0.3633
[0.3573771119117737, 0.888700008392334]

```

7. Predições

```

y_pred = model.predict(x_test).argmax(axis=1)

```

```
print(y_pred)
```

```
313/313 _____ 1s 2ms/step
      [9 2 1 ... 8 1 5]
```

#Agora geramos um matriz de confusão para análise das classes:

Observamos que as seguintes classes foram as que o modelo mais errou:

6 - classificada 137 na 0 e 57 na 2

4 - classificada 142 na 6 e 160 na 2

2 - classificada 85 na 6 e 61 na 4

```
import seaborn as sns
```

```
# Calcular a acurácia
```

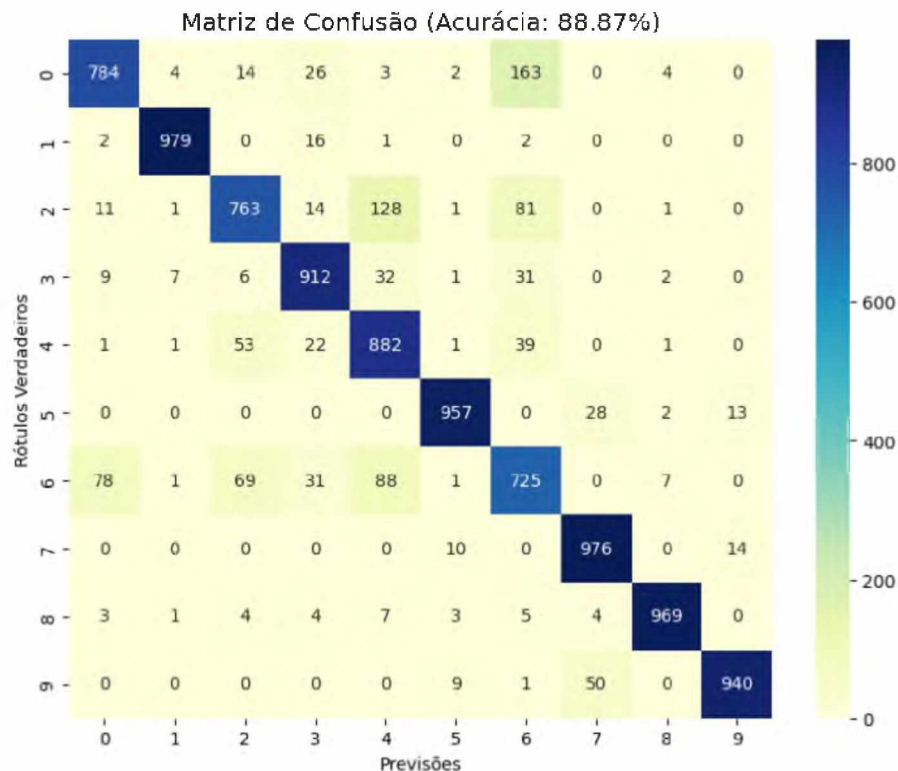
```
accuracy = (y_test == y_pred).sum() / len(y_test)
#print(f"Acurácia: {accuracy * 100:.2f}%")
```

```
# Matriz de confusão
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
# Plotando a matriz de confusão
```

```
plt.figure(figsize=(9, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu', xticklabels=np.arange(10),
yticklabels=np.arange(10))
plt.title(f'Matriz de Confusão (Acurácia: {accuracy * 100:.2f}%)', fontsize=14)
plt.xlabel('Previsões')
plt.ylabel('Rótulos Verdadeiros')
plt.show()
```

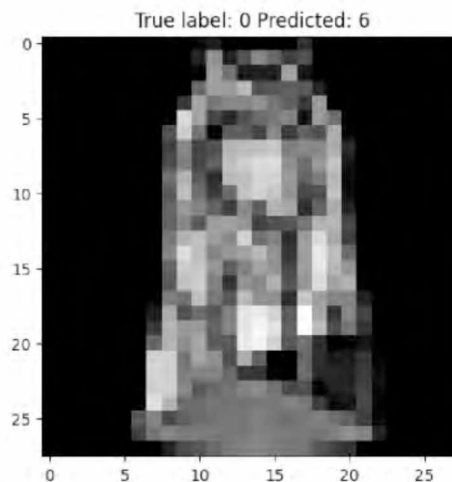


Mostrar algumas classificações erradas

```
misclassified = np.where(y_pred != y_test)[0]
i = np.random.choice(misclassified)

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

Text(0.5, 1.0, 'True label: 0 Predicted: 6')
```



Aqui mostramos 20 imagens aleatórias onde as classificadas erroneamente o texto está em vermelho

```
# Selecionando 20 imagens aleatórias para exibir
indices = np.random.choice(len(x_test), 20, replace=False)

# Criando a figura para exibir as imagens, ajustando o tamanho das imagens
fig, axes = plt.subplots(4, 5, figsize=(8, 8)) # Ajuste o figsize para controlar o tamanho total da figura

for i, ax in zip(indices, axes.flatten()):
    # Exibindo as imagens no tamanho real (28x28)
    ax.imshow(x_test[i], cmap='gray', aspect='auto') # Sem reshape, pois já tem tamanho 28x28
    ax.axis('off') # Removendo os eixos

    # Verificando se a classificação está correta
    if y_pred[i] == y_test[i]:
        color = 'green'
    else:
        color = 'red'

    # Exibindo o rótulo correto (em vermelho para erros)
    ax.set_title(f"Pred: {y_pred[i]} (True: {y_test[i]})", color=color, fontsize=10) # Ajuste o fontsize para controlar o tamanho do texto

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



Agora mostramos algumas imagens previstas erroneamente e ao lado uma imagem da Classe que foi classificada onde podemos observar que todas tem uma certa semelhança realmente, o que fez o modelo confundir as características.

Selecionamos 10 imagens aleatórias onde o modelo errou para exibir

```
indices = np.random.choice(np.where(y_pred != y_test)[0], 10, replace=False)
```

Criando uma figura com 2 colunas para cada imagem

```
fig, axes = plt.subplots(10, 2, figsize=(6, 15))
```

```
for row, idx in enumerate(indices):
```

```
    axes[row, 0].imshow(x_test[idx].reshape(28, 28), cmap='gray')
```

```
    axes[row, 0].axis('off')
```

```
    axes[row, 0].set_title(f"True: {class_names[y_test[idx]]}", color='black',
    fontsize=10)
```

```
    # Encontrando uma imagem da classe que foi incorretamente prevista
```

```
    incorrect_class = y_pred[idx]
```

```
    # Pegando um exemplo da classe incorretamente prevista
```

```
    incorrect_idx = np.where(y_test == incorrect_class)[0][0]
```

```
    # Exibindo a imagem de uma amostra da classe incorretamente prevista
```

```
    axes[row, 1].imshow(x_test[incorrect_idx].reshape(28, 28), cmap='gray')
```

```
    axes[row, 1].axis('off')
```

```
    axes[row, 1].set_title(f"Pred: {class_names[incorrect_class]}", color='red',
    fontsize=10)
```

```
plt.tight_layout()
```

```
plt.show()
```



2 Regressão (RNA)

Importação dos dados

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

```
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
]
```

```
print(data.head())
```

	acidez_fixa	acidez_volatil	acido_citrico	acucar_residual	cloretos	\
0	7.4	0.70	0.00	1.9	0.076	
1	7.8	0.88	0.00	2.6	0.098	
2	7.8	0.76	0.04	2.3	0.092	

```

3          11.2          0.28          0.56          1.9          0.075
4           7.4          0.70          0.00          1.9          0.076

      dióxido_de_enxofre_livre  dióxido_de_enxofre_total  densidade  pH \
0                11.0                34.0      0.9978  3.51
1                25.0                67.0      0.9968  3.20
2                15.0                54.0      0.9970  3.26
3                17.0                60.0      0.9980  3.16
4                11.0                34.0      0.9978  3.51

      sulfatos  alcool  score_qualidade_vinho
0      0.56      9.4          5
1      0.68      9.8          5
2      0.65      9.8          5
3      0.58      9.8          6
4      0.56      9.4          5

```

```

# Número de linhas e colunas
data.shape

```

(1599, 12)

```

# separação dos dados
X = data.drop('score_qualidade_vinho', axis=1)
Y = data['score_qualidade_vinho']

```

```

# primeiras linhas de X e y para verificar
print("X (Features):")
print(X.head())
print("\nY (Target):")
print(Y.head())

```

```

X (Features):
      acidez_fixa  acidez_volatil  acido_citrico  acucar_residual  cloretos \
0           7.4          0.70          0.00          1.9          0.076
1           7.8          0.88          0.00          2.6          0.098
2           7.8          0.76          0.04          2.3          0.092
3          11.2          0.28          0.56          1.9          0.075
4           7.4          0.70          0.00          1.9          0.076

      dióxido_de_enxofre_livre  dióxido_de_enxofre_total  densidade  pH \
0                11.0                34.0      0.9978  3.51
1                25.0                67.0      0.9968  3.20
2                15.0                54.0      0.9970  3.26
3                17.0                60.0      0.9980  3.16
4                11.0                34.0      0.9978  3.51

      sulfatos  alcool
0      0.56      9.4
1      0.68      9.8
2      0.65      9.8
3      0.58      9.8
4      0.56      9.4

Y (Target):
0      5
1      5
2      5
3      6
4      5
Name: score_qualidade_vinho, dtype: int64

```

Matriz de correlação apenas para visualização dos dados que mais tem correlações

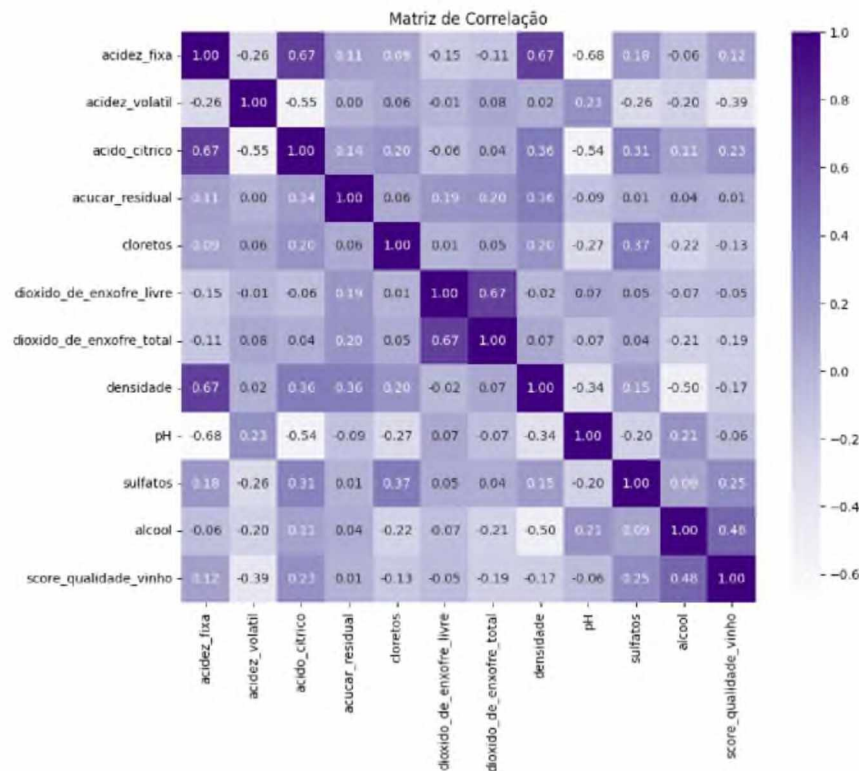
```

import seaborn as sns
correlation_matrix = data.corr()
# Geramos o heatmap da matriz de correlação
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="Purples", fmt=".2f")
plt.title("Matriz de Correlação")

```



```
plt.show()
```



****Analisando a matriz de correlação, podemos observar os seguintes pontos:****

Correlação com a variável de saída (**score_qualidade_vinho**):

O **score_qualidade_vinho** apresenta correlações moderadas com algumas variáveis, mas nenhuma extremamente forte.

O **álcool** possui a correlação mais forte com **score_qualidade_vinho** (0.48), sugerindo que o teor alcoólico tem uma influência positiva na qualidade do vinho.

Sulfatos também mostra uma correlação positiva leve (0.25), sugerindo que níveis mais altos de sulfato podem estar associados a uma qualidade melhor.

Correlação entre outras variáveis:

Densidade e **acidez_fixa** têm uma correlação moderada (0.67) entre elas, o que indica que vinhos mais densos tendem a ter uma acidez fixa maior.

Dióxido_de_enxofre_livre e **dióxido_de_enxofre_total** têm uma correlação moderada-alta (0.67) entre si, o que é esperado, pois essas medidas estão relacionadas quimicamente.

pH e **acidez_fixa** possuem uma correlação negativa forte (-0.68).

Cloretos, **açúcar_residual** e **dióxido de enxofre total** têm pouca correlação com a variável de qualidade, indicando que essas variáveis podem ter menos influência direta na percepção de qualidade.

Nesta parte tentamos utilizar apenas as variáveis com maior correlação com a qualidade
mais não obtivemos melhora nos resultados finais portanto voltamos a utilizar todas
as variáveis da base conforme indica a questão.

```
# Selecionar apenas as variáveis importantes com base na matriz de correlação
#X = data[['álcool', 'sulfatos', 'densidade']]
#Y = data['score_qualidade_vinho']
```

Exibir as primeiras linhas de X e Y para verificar

```
#print("X (Features):")
#print(X.head())
#print("\nY (Target):")
#print(Y.head())
```

Normalização dos dados

Obtivemos melhor resultado com StandardScaler que normaliza entre -1 e 1

```
from sklearn.preprocessing import StandardScaler
```

```
# Inicializando o StandardScaler
scaler = StandardScaler()
```

```
# Ajustando o scaler nos dados de treino e transformando X
X = scaler.fit_transform(X)
```

```
# Exibir os dados normalizados
print("X (Features) Normalizado:")
print(X[:5]) # Mostrar as primeiras 5 linhas
```

```
X (Features) Normalizado:
[[-0.52835961  0.96187667 -1.39147228 -0.45321841 -0.24370669 -0.46619252
  -0.37913269  0.55827446  1.28864292 -0.57920652 -0.96024611]
 [-0.29854743  1.96744245 -1.39147228  0.04341614  0.2238752  0.87263823
   0.62436323  0.02826077 -0.7199333  0.1289504 -0.58477711]
 [-0.29854743  1.29706527 -1.18607043 -0.16942723  0.09635286 -0.08366945
   0.22904665  0.13426351 -0.33117661 -0.04808883 -0.58477711]
 [ 1.65485608 -1.38444349  1.4841536 -0.45321841 -0.26496041  0.10759209
   0.41150046  0.6642772 -0.97910442 -0.46118037 -0.58477711]
 [-0.52835961  0.96187667 -1.39147228 -0.45321841 -0.24370669 -0.46619252
  -0.37913269  0.55827446  1.28864292 -0.57920652 -0.96024611]]
```

```
# Também fizemos a normalização dos dados com MinMaxScaler
# que faz entre 0 e 1 mais a normalização com standardscaler entre -1 e 1 deu
# melhor resultado
```

```
#from sklearn.preprocessing import MinMaxScaler
```

```
#scaler = MinMaxScaler()
#X = scaler.fit_transform(X)
```

Separação da base em treino e teste (75/25)

```
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)
```

Criação do modelo

```
from tensorflow.keras import layers, models, regularizers
```

```
# 11 camadas referente as features
i = tf.keras.layers.Input(shape=(11,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
```

```
# tentamos também com Regularização L2 sem melhoras no resultado
#x = layers.Dense(50, activation="relu", kernel_regularizer=regularizers.l2(0.01))(i)
```

```
# inserimos uma camada de Dropout com 50% e depois reduzimos para 20%
# esta cama ajuda a evitar overfitting
x = layers.Dropout(0.2)(x)
```

```
x = tf.keras.layers.Dense(1)(x)
```

```
model = tf.keras.models.Model(i, x)
```

Criação de funções para as métricas R2 e RMSE serem inseridas no modelo

```
# Tivemos que fazer algumas alterações na criação das funções
# convertendo y_true para float32

# Função RMSE
def rmse(y_true, y_pred):
    y_true = backend.cast(y_true, dtype='float32')
    return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))

# Função R2
def r2(y_true, y_pred):
    y_true = backend.cast(y_true, dtype='float32')
    y_true_mean = backend.mean(y_true)
    ss_tot = backend.sum(backend.square(y_true - y_true_mean))
    ss_res = backend.sum(backend.square(y_true - y_pred))
    return 1 - ss_res / (ss_tot + backend.epsilon())
```

Compilação e treinamento do Modelo

Na compilação testamos com os três otimizadores e com várias taxas de aprendizado mais a que deu melhor resultado foi com RMSprop

```
optimizer=tf.keras.optimizers.Adam(learning_rate=0.005)

#optimizer=tf.keras.optimizers.SGD(learning_rate=0.005, momentum=0.5)
#optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
              loss="mse",
              metrics=[rmse, r2])

# Testamos patience com 30 e com 10 tbm apenas para teste mais não melhorou o resultado
# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True)

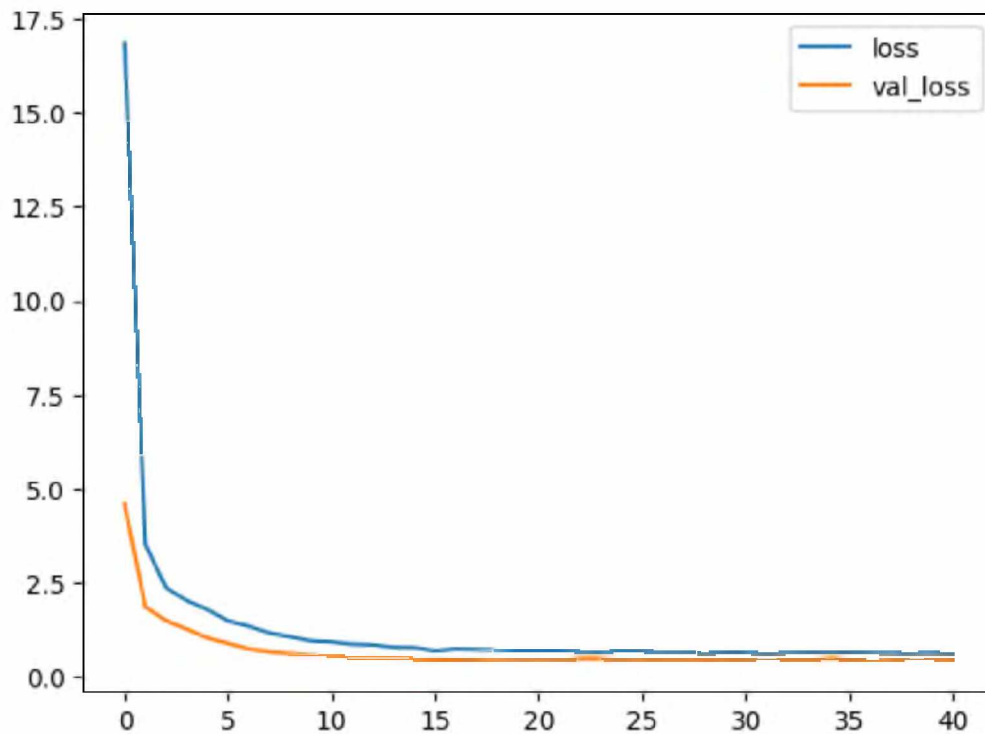
r=model.fit(x_train,y_train,epochs=100,validation_data=(x_test,y_test),callbacks=[early_stop])

Epoch 38/100
 38/38 ----- 0s 3ms/step - loss: 0.5894 - r2: 0.0102 - rmse: 0.6040 -
val_loss: 0.4264 - val_r2: 0.3536 - val_rmse: 0.5086
Epoch 39/100
 38/38 ----- 0s 2ms/step - loss: 0.6084 - r2: 0.0364 - rmse: 0.6251 -
val_loss: 0.4716 - val_r2: 0.2827 - val_rmse: 0.5267
Epoch 40/100
 38/38 ----- 0s 3ms/step - loss: 0.6507 - r2: -0.0292 - rmse: 0.6315 -
val_loss: 0.4653 - val_r2: 0.2865 - val_rmse: 0.5226
Epoch 41/100
 38/38 ----- 0s 3ms/step - loss: 0.6134 - r2: 0.0388 - rmse: 0.6102 -
val_loss: 0.4318 - val_r2: 0.3386 - val_rmse: 0.5064
```

Avaliação do modelo

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

```
plt.legend()
```

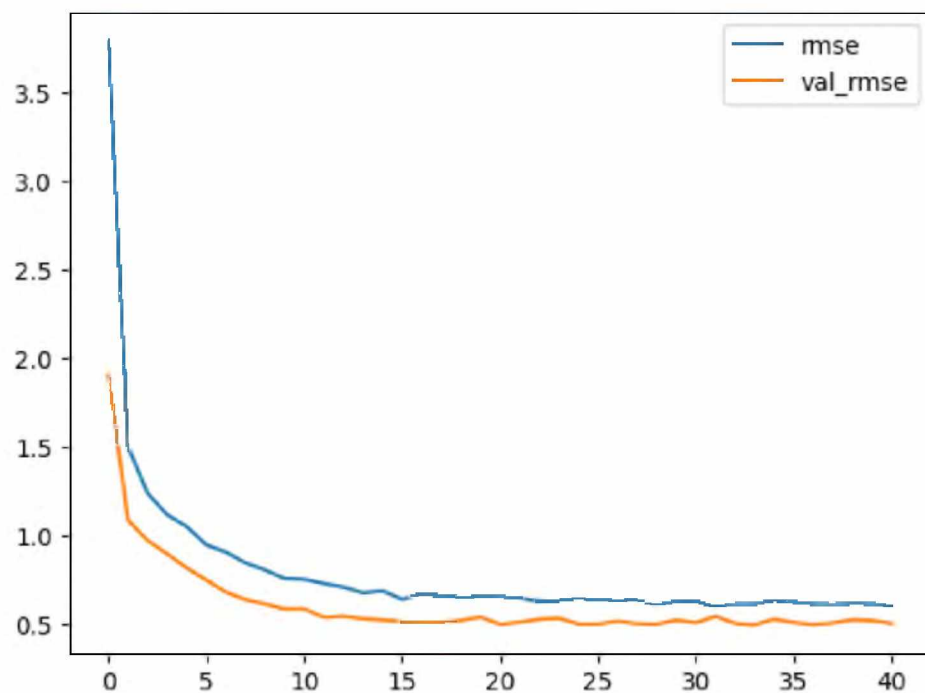


Explicação do Gráfico de Perda

No início do treinamento, tanto a perda de treinamento quanto a de validação diminuem rapidamente, sendo um bom sinal, pois indica que o modelo está aprendendo e ajustando seus parâmetros para se adaptar aos dados, mas após a época 10 as curvas se estabilizam, e a perda mantém um nível relativamente constante, e a curva de validação permanece próxima a de treinamento, o que indica que o modelo não está sofrendo overfitting.

O gráfico indica que o modelo aprendeu e convergiu para uma solução estável, com uma boa generalização entre os dados, indicando que o modelo está equilibrado.

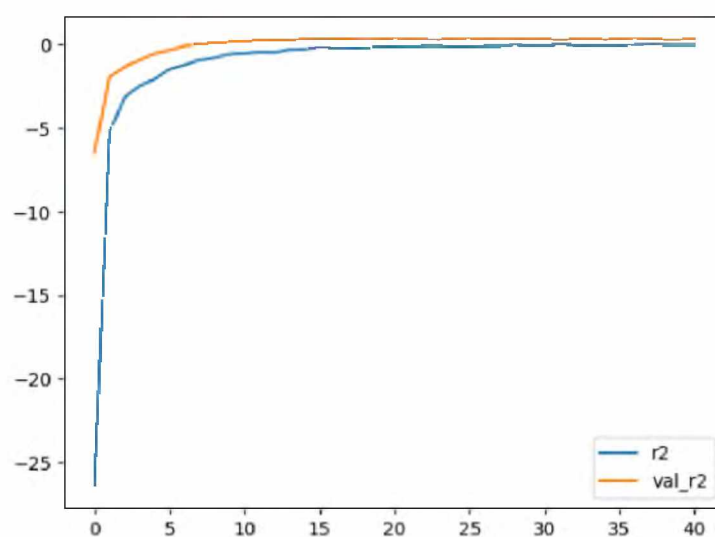
```
plt.plot( r.history["rmse"], label="rmse" )
plt.plot( r.history["val_rmse"], label="val_rmse" )
plt.legend()
```



Explicação do Gráfico de RMSE

Nas primeiras épocas o RMSE diminui rapidamente nos dois conjuntos de dados, mais após da época 10 estabiliza em um ponto razoável, que para ser ótimo teria que chegar mais próximo de zero. E por esta estabilização em várias épocas também indica que um número maior de épocas não iria melhorar o resultado do modelo.

```
plt.plot( r.history["r2"], label="r2" )
plt.plot( r.history["val_r2"], label="val_r2" )
plt.legend()
```



Explicação do Gráfico R²

Percebemos que no início aumenta rapidamente, até passar de valores negativos aproximadamente pela décima época, e que posteriormente estabiliza próximo a zero sem mudanças significativas,

indicando que o modelo não está conseguindo fazer boas previsões em relação a variabilidade dos dados, e também que um aumento do número de épocas não iria melhorar o resultado do mesmo.

Predições

```
# Predição
y_pred = model.predict(x_test).flatten()

# Cálculo das métricas de acurácia: mse, r2 e rmse
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Resultados das métricas de acurácia
print("mse      = ", mse)
print("rmse     = ", rmse)
print("r2       = ", r2)

mse      = 0.4178302620711986
rmse     = 0.6463979131086351
r2       = 0.3966349959373474
```

Explicação das métricas

No geral, os resultados obtidos são razoáveis, com espaço para melhorias, tentamos de diversas formas obter melhor resultado mais este foi o máximo que atingimos, sem fugir muito do modelo utilizado em aula.

****mse:**** 0,41 ainda está alto em se tratando das média dos erros quadráticos.

****rmse:**** 0.64 tbm está elevado por ser o erro quadrático médio elevado a raiz quadrada.

****R²:**** o resultado indica que a variância dos dados é explicada pelo modelo em 39.6%, que ainda é baixo.

3 Sistemas de Recomendação

Prática Sistemas de Recomendação: Criar um sistema de recomendação de filmes

Importação das bibliotecas

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

Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

Importação dos dados

```
df=pd.read_csv("/content/drive/MyDrive/frameworksIA/Base_livros.csv",sep=";",
skipinitialspace=True)
df.head()
```

	ISBN	Titulo	Autor	Ano	Editora	ID_usuario	Notas
0	ISBN	Titulo	Autor	Ano	Editora	ID_usuario	Notas
1	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0
2	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	276726	2

	ISBN	Titulo	Autor	Ano	Editora	ID_usuario	Notas
3	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6
4	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	276729	1

```
# Exibir os valores únicos do campo 'Notas' apenas para saber a escala e desnormalizar no final
valores_unicos_notas = df['Notas'].unique()
print("Valores únicos em 'Notas':", valores_unicos_notas)
```

```
Valores únicos em 'Notas': ['Notas' '0' '2' ... 'Fantagraphics Books' 'Forge Books' '29888']
```

```
# verificar linhas com valores nulos
print(df[df.isnull().any(axis=1)])
```

```

ISBN                               Titulo \
23      61076031  Mary-Kate & Ashley Switching Goals (Mary-Kate ...
691     1567920047  A Tally of Types: With Additions by Several Ha...
825     2264009306                                CantilÃfÃnes en gelÃfÃe
1285    2070372294                                Le Coq De Bruygre (Folio Ser
1688     785268030                                Starting Your Marriage Right &lt
...
126334   393058557  The Girl with the Long Back: A Harpur & Iles M...
127108    3252477                                A midsummer night's dream
127119   1557132445                                "The Women at the Pump (Sun & Moon Classics
127230   3518065793  Das Glasperlenspiel: Versuch e. Lebensbeschrei...
128122    312307195  The Desperate Remedy: Henry Gresham and the Gu...
```

```

Autor                               Ano           Editora \
23      Mary-Kate & Ashley           NaN           NaN
691      NaN                         NaN           NaN
825      NaN                         NaN           NaN
1285     NaN                         NaN           NaN
1688     NaN                         NaN           NaN
...
126334   Bill James                   2004   W.W. Norton &
127108     NaN                       NaN           NaN
127119    No 115)"  Knut Hamsun           1996
127230     NaN                       NaN           NaN
128122     NaN                       NaN           NaN
```

```

ID_usuario  Notas
23          NaN  NaN
691          NaN  NaN
825          NaN  NaN
1285         NaN  NaN
1688         NaN  NaN
...
126334         NaN  NaN
127108         NaN  NaN
127119  Consortium Book Sales &  NaN
127230         NaN  NaN
128122         NaN  NaN
```

```
[686 rows x 7 columns]
```

```
# Remover linhas com valores NULOS
df = df.dropna()
```

```
# verificar linhas com valores nulos para ver se apagou
print(df[df.isnull().any(axis=1)])
```

```
Empty DataFrame
Columns: [ISBN, Título, Autor, Ano, Editora, ID_usuario, Notas]
Index: []
```

Conversão de ID_usuario e ISBN para categoria

```
df.ID_usuario = pd.Categorical(df.ID_usuario)
df['new_ID_usu'] = df.ID_usuario.cat.codes
```

```
df.ISBN = pd.Categorical(df.ISBN)
df['new_ISBN'] = df.ISBN.cat.codes
```

```
from sklearn.preprocessing import MinMaxScaler
```

```
df['Notas'] = pd.to_numeric(df['Notas'], errors='coerce')
```

```
df = df.dropna(subset=['Notas'])
```

```
#Normalizar notas entre 0 e 1
scaler = MinMaxScaler()
df['notas_norm'] = scaler.fit_transform(df[['Notas']])
```

```
<ipython-input-9-8c34480651d2>:9: 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

```
df['notas_norm'] = scaler.fit_transform(df[['Notas']])
```

```
df.head()
```

ISBN	Título	Autor	Ano	Editora	ID_usuario	Notas	new_ID_usu	new_ISBN	notas_norm
2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlami ngo Canada	276725	0.0	7026	26192	0.000000
60973129	Decision in Normandy	Carlo D'Este	1991	HarperPeren nial	276726	2.0	7027	86632	0.000007
374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6.0	7028	50102	0.000022
393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	276729	1.0	7029	56336	0.000004
399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	276729	9.0	7029	58737	0.000032

```
# Dimensões
N = len(set(df.new_ID_usu))
M = len(set(df.new_ISBN))
```



```
# dimensão do embedding
K = 10
```

Criar o modelo

```
from tensorflow.keras.regularizers import l2
from tensorflow.keras.layers import Dropout

# usuário
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

# livro
m = Input(shape=(1,))
m_emb = Embedding(M, K)(m) # saída : num_samples, 1, K
m_emb = Flatten()(m_emb)   # saída : num_samples, K

x = Concatenate()([u_emb, m_emb])

# Adicionando Dropout e Regularização L2
x = Dense(128, activation="relu", kernel_regularizer=l2(0.01))(x) # Regularização L2
x = Dropout(0.5)(x) # Dropout com taxa de 50%

x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)
```

Compilação do modelo

```
model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.001, momentum=0.9)
    #optimizer=Adam(learning_rate=0.001)
)
```

Separação dos dados e pré-processamento

```
#user_ids, book_ids, ratings = shuffle(df.new_ID_usu, df.new_ISBN, df.Notas)
user_ids, book_ids, ratings = shuffle(df.new_ID_usu, df.new_ISBN, df.notas_norm)

# Separar os dados em 80% para treino e 20% para teste
Ntrain = int(0.80 * len(ratings))

train_user = user_ids[:Ntrain]
train_book = book_ids[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_book = book_ids[Ntrain:]
test_ratings = ratings[Ntrain:]

# Centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating
```

Treinamento do modelo

```
r = model.fit(
    x=[train_user, train_book],
    y=train_ratings,
    epochs=50,
    batch_size=128,
    verbose=2,
    validation_data=([test_user, test_book], test_ratings)
)
```

```

Epoch 40/50
782/782 - 1s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 41/50
782/782 - 1s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 42/50
782/782 - 1s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 43/50
782/782 - 3s - 3ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 44/50
782/782 - 3s - 3ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 45/50
782/782 - 2s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 46/50
782/782 - 2s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 47/50
782/782 - 2s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 48/50
782/782 - 1s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 49/50
782/782 - 1s - 2ms/step - loss: 0.0079 - val_loss: 0.0087
Epoch 50/50
782/782 - 1s - 2ms/step - loss: 0.0079 - val_loss: 0.0087

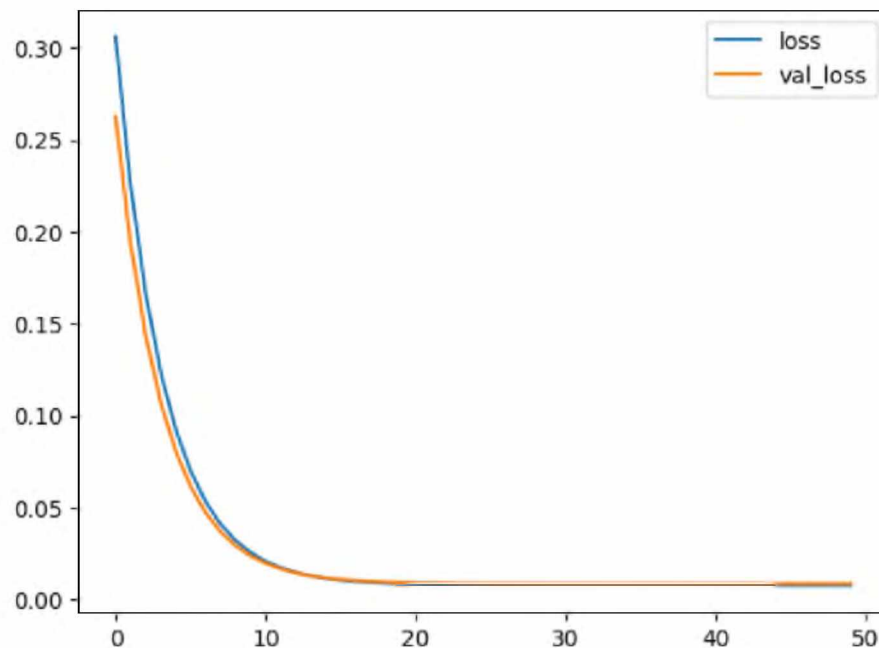
```

Plotar a função de perda

```

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

```



Análise do Gráfico de Perda

Ambas as curvas apresentam uma diminuição constante ao longo das épocas, indicando que o modelo está aprendendo e reduzindo o erro durante o treinamento, e mostra uma certa estabilização a partir da época 25 apesar de ainda estar diminuindo a perda.

As curvas de treinamento e validação permanecem próximas durante todo o processo o que sugere que o modelo não está sofrendo overfitting.

9. Recomendações para o usuário 5360

```

# Gerar o array com o usuário único
# repete a quantidade de livros
input_usuario = np.repeat(a=5360, repeats=M)
book = np.array(list(set(book_ids)))

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

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

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

print("Recomendação:")
print("Livro - ", book[idx], " / ", rat[idx] , "*")
print("Dados originais do Livro:")
isbn_recomendado = book[idx]
# Filtrar o DataFrame para obter as informações do livro recomendado
livro_recomendado = df[df['new_ISBN'] == isbn_recomendado].iloc[0]
print("ISBN :", livro_recomendado['ISBN'])
print("Nota :", livro_recomendado['Notas'])
print("Título:", livro_recomendado['Título'])
print("Autor:", livro_recomendado['Autor'])
print("Ano:", livro_recomendado['Ano'])
print("Editora:", livro_recomendado['Editora'])

3908/3908 _____ 11s 3ms/step
Recomendação:
Livro - 90758 / 0.01694195 *
Dados originais do Livro:
ISBN : 671034553
Nota : 2.0
Título: A Place Among the Stars : The Women of Star Trek: Voyager
Autor: Pocket Books
Ano: 1998
Editora: Star Trek

# Gerar o array com o usuário único e repetir pela quantidade de livros
input_usuario = np.repeat(a=5360, repeats=M)
book = np.array(list(set(book_ids)))

# Fazer a predição de notas para o usuário
preds = model.predict([input_usuario, book])

# Descentralizar e escalonar as predições para o intervalo de notas originais
rat_normalizada = preds.flatten() + avg_rating
# Escalando de volta para o intervalo original (por exemplo, 1 a 5)
rat = rat_normalizada * 10

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

# Exibir a recomendação principal
print("=== Recomendação de Livro Personalizada ===")
print(f"Com base no histórico do usuário, recomendamos o livro com a maior previsão de avaliação.")
print("Livro Recomendado:", book[idx])
print("Nota Prevista:", round(rat[idx], 2), "*") # Arredondar para 2 casas decimais

# Obter os dados originais do livro recomendado
isbn_recomendado = book[idx]
livro_recomendado = df[df['new_ISBN'] == isbn_recomendado].iloc[0]

# Exibir mais informações detalhadas do livro
print("\n=== Informações do Livro Recomendado ===")

```

```

print("ISBN          :", livro_recomendado['ISBN'])
print("Nota Média    :", livro_recomendado['Notas'])
print("Título       :", livro_recomendado['Título'])
print("Autor        :", livro_recomendado['Autor'])
print("Ano          :", livro_recomendado['Ano'])
print("Editora      :", livro_recomendado['Editora'])

```

Observação adicional sobre as recomendações

```

print("\nNota: A recomendação é baseada na maior previsão de avaliação com base no modelo de rede neural.")

```

```

3908/3908 ----- 6s 2ms/step
=== Recomendação de Livro Personalizada ===
Com base no histórico do usuário, recomendamos o livro com a maior previsão de
avaliação.
Livro Recomendado: 90758
Nota Prevista: 0.17 *

=== Informações do Livro Recomendado ===
ISBN          : 671034553
Nota Média    : 2.0
Título       : A Place Among the Stars : The Women of Star Trek: Voyager
Autor        : Pocket Books
Ano          : 1998
Editora      : Star Trek

```

Nota: A recomendação é baseada na maior previsão de avaliação com base no modelo de rede neural.

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.

Importação da imagem

```

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:
href=https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-
cat_on_snow.jpg">Von.grzanka</a>'))

```

<a

Downloading data from

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

2125399/2125399 ————— 0s 0us/step



Preparar o modelo de extração de recursos

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

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5

87910968/87910968 ————— 0s 0us/step

```
# 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

```
def calc_loss(img, model):
    # 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)
```

Subida de gradiente

```
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

img = img + gradients*step_size
img = tf.clip_by_value(img, -1, 1)

return loss, img
```

```
deepdream = DeepDream(dream_model)
```

Circuito principal (*Main Loop*)

```
def run_deep_dream_simple(img, steps=100, step_size=0.01):

    img = tf.keras.applications.inception_v3.preprocess_input(img)
    img = tf.convert_to_tensor(img)
    step_size = tf.convert_to_tensor(step_size)
    steps_remaining = steps
    step = 0
    while steps_remaining:
        if steps_remaining>100:
            run_steps = tf.constant(100)
        else:
            run_steps = tf.constant(steps_remaining)
        steps_remaining -= run_steps
        step += run_steps

        loss, img = deepdream(img, run_steps, tf.constant(step_size))

        display.clear_output(wait=True)
        show(deprocess(img))
        print ("Step {}, loss {}".format(step, loss))

    result = deprocess(img)
    display.clear_output(wait=True)
    show(result)
    return result

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



Explicação da imagem

A imagem é de baixa resolução e os padrões estão praticamente na mesma granularidade, tendo em vista que o processamento em escala única a imagem é processada apenas em sua escala original, assim é mais difícil para o modelo capturar detalhes que existem em diferentes tamanhos ou escalas, como pequenos padrões ou texturas.

Levando o modelo até uma oitava

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

OCTAVE_SCALE = 1.3

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=100, step_size=0.005)

end = time.time()
end-start
```



Explicação da imagem

Agora a imagem fica mais detalhada e complexa, com maior mistura de texturas de padrões.

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

B - RESOLUÇÃO

Visualização de Dados e Storytelling

Dados Brutos

Exemplo de dados gerados a partir dos dispositivos distribuídos pela cidade em painéis de LED, totens digitais de publicidade. De tempos em tempos estes dados são enviados a um servidor central de controle de publicidade digital.

Agenda	ID ponto	Duracao Midia	Data View	Hora View	GPSLatitude	GPSLongitude
193	239VxWL9gq7NBpxr0	20	2025-01-09	11:47:01	-19.937206294876045	-43.93569054749388
193	239VxWL9gq7NBpxr0	20	2025-01-09	11:47:31	-19.937206294876045	-43.93569054749388
193	239VxWL9gq7NBpxr0	20	2025-01-09	11:48:11	-19.937206294876045	-43.93569054749388
193	3b1b21e85b10173e	10	2025-01-09	11:48:52	-19.91977401780811	-43.93907536347547
193	3b1b21e85b10173e	10	2025-01-09	11:49:32	-19.91977401780811	-43.93907536347547
193	3b1b21e85b10173e	10	2025-01-09	11:50:12	-19.91977401780811	-43.93907536347547
193	3b1b21e85b10173e	10	2025-01-09	11:50:52	-19.91977401780811	-43.93907536347547
193	cd7bef1659857728	20	2025-01-09	11:51:22	-19.924539150514665	-43.92729109503373
193	cd7bef1659857728	20	2025-01-09	11:52:03	-19.924539150514665	-43.92729109503373
193	cd7bef1659857728	20	2025-01-09	11:52:43	-19.924539150514665	-43.92729109503373

Como a publicidade DOOH conecta Belo Horizonte

PeraAI é um jovem empreendedor que acaba de lançar sua primeira marca de roupas inteligentes em Belo Horizonte. Para garantir o sucesso de seu novo empreendimento, ele sabe que precisa alcançar o público certo, no lugar certo e no momento certo. Mas como fazer isso em uma cidade tão dinâmica? A resposta está nas ruas, nos prédios e nos pontos estratégicos onde milhares de olhares se voltam todos os dias: **a publicidade DOOH (Digital Out-of-Home)**.

Ao analisar os dados de impacto da mídia DOOH na cidade de seu interesse, PeraAI descobre uma dinâmica fascinante na movimentação urbana. Alguns bairros se destacam como verdadeiros epicentros de atenção, enquanto outros representam nichos estratégicos de público. Cada edifício no gráfico representa o volume médio de visualizações diárias de publicidade digital, dando vida aos números e transformando estatísticas em oportunidades de negócio. PeraAI percebe um **circuito** interessante para imprimir sua marca, atingindo uma gama de pessoas residentes ou de passagem.



Figura 1: Quantidade média de views de publicidade DOOH - Região central BH

Centro: O Coração Pulsante da Cidade

Com **6.378 visualizações por dia**, o **Centro** de Belo Horizonte é o grande palco onde a cidade acontece. Milhares de pessoas transitam diariamente por suas avenidas, shoppings e terminais de ônibus, tornando essa região um dos pontos mais valiosos para exibição de publicidade. PeraAI percebe que um anúncio aqui significa atingir profissionais, estudantes e turistas que cruzam as ruas sem perder de vista os painéis publicitários com mídias digitais.

Lourdes: O Bairro da Sofisticação

Logo ao lado, o bairro **Lourdes** apresenta **5.230 visualizações diárias**, ultrapassando até mesmo o Centro. Com seus cafés refinados, boutiques de luxo e restaurantes badalados, a região é um ponto nobre onde o público tem alto poder aquisitivo. Para PeraAI, essa é a chance perfeita de posicionar sua marca entre consumidores exigentes e atentos às tendências da e de inovação.

Funcionários: Sofisticado e Tradicional

A alguns quilômetros dali, o bairro **Funcionário (5.200 visualizações diárias)** é um dos mais tradicionais da cidade. Reúne gente com poder aquisitivo, interessado em produtos novos. PeraAI sabe que uma boa impressão de seu empreendimento nestes bairros pode impactar positivamente muito a marca. Este bairro é estratégico pois está próximo ao Centro e Savassi.

São Pedro: O Público Local que Gera Conversão

O **São Pedro** é um bairro localizado na Zona Sul de Belo Horizonte. Apesar de predominantemente residencial, o bairro também é conhecido por sua intensa vida noturna, ao lado de seus vizinhos Santo Antônio de Savassi. Com em média **4.550 visualizações diárias** PeraAI vê oportunidades nestes pontos onde existem coexistência de culturas, onde as pessoas gostam de se mostrar.

Serra: Mobilidade e Movimento

O bairro **Serra (4.300 visualizações diárias)** é um dos mais antigos da cidade, onde moradores fizeram sua via e ainda moram no bairro. Apesar da tradição, concentra alto poder aquisitivo. É vizinho do bairro Mangabeiras, o mais elitizado da cidade. PeraAI vê oportunidade pois muitos moradores são bom frequentadores de shoppings e gostam de inovação.

Santa Efigênia: Predominante Serviços Médicos

Próximo ao centro e ao bairro Funcionários, o bairro **Santa Efigênia (3.760 visualizações diárias)** possui intensa movimentação nas ruas por conta de serviços médicos e rede hospitalar. PeraAI observa que a movimentação urbana não se concentra apenas no centro. São áreas que conectam importantes vias da cidade, atraindo motoristas, pedestres e ciclistas que, no trânsito ou nas calçadas, mantêm contato diário com a publicidade DOOH.

A publicidade que move Belo Horizonte

Com os números em mãos, PeraAI percebe que a publicidade DOOH não é apenas sobre exibir um anúncio, mas sim sobre **se conectar e interagir com a cidade**. Cada bairro tem seu próprio perfil, e os dados mostram como cada espaço pode ser aproveitado para atingir o público certo.

Ele decide lançar sua campanha com anúncios espalhados estrategicamente, aproveitando os locais de maior visibilidade e presença do seu público-alvo. Agora, ao caminhar pela cidade, PeraAI não apenas vê os painéis publicitários – ele entende a **história** que cada um deles conta.

E você? Já pensou onde sua marca pode estar sendo vista hoje?

APÊNDICE 15 - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java.

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida).

Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

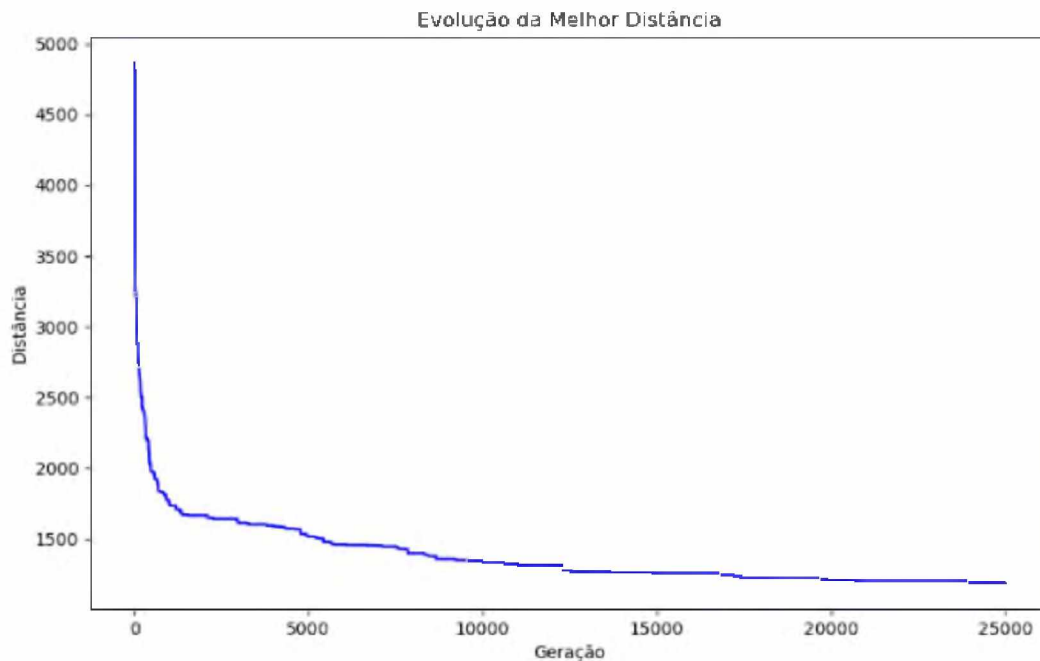
Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels.

O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

- (1) considere o cromossomo formado pelas cidades, onde a cidade de início (escolhida aleatoriamente) deverá estar na posição 0 e 100 e a ordem das cidades visitadas nas posições de 1 a 99 deverão ser definidas pelo algoritmo genético.
- (2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
- (3) Utilize no mínimo uma população com 100 indivíduos;
- (4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;
- (5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);
- (6) Preserve sempre a melhor solução de uma geração para outra.

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.



LOG GERAÇÕES:

Sucesso, 100 cidades (GENS) foram geradas. Tentativas: 575.

Geração 0: Melhor Distância = 4865.216266052416

...

Geração 24900: Melhor Distância = 1189.9014775389585

Algoritmo completo:

```
import random
import math
import matplotlib.pyplot as plt

# Considerando distância mínima entre cidades (pontos) para evitar sobreposição ou muito
próximos
def gerar_pontos_cidades(num_gen_pontos_cidade=100, max_x=100, max_y=100, min_distance=8):
    gen_pontos_cidades = []
    tentativa = 0
    max_attempts = 2000 # Número máximo de tentativas para encontrar 100 cidades
    while len(gen_pontos_cidades) < num_gen_pontos_cidade and tentativa < max_attempts:
        new_city = (random.randint(0, max_x), random.randint(0, max_y))
        valid = True
        # Verificar se a nova cidade está longe o suficiente das cidades existentes
        for existing_city in gen_pontos_cidades:
            distance = math.sqrt((new_city[0] - existing_city[0])**2 +
(new_city[1] - existing_city[1])**2)
            if distance < min_distance: # Distância mínima pre-estabelecida entre
as cidades
                valid = False
                break
        if valid:
```

```

        gen_pontos_cidades.append(new_city)
        tentativa += 1
    if len(gen_pontos_cidades) < num_gen_pontos_cidade:
        print(f"Aviso: Apenas {len(gen_pontos_cidades)} cidades foram geradas devido
à restrição de distância mínima.")
    else:
        print(f"Sucesso, {len(gen_pontos_cidades)} cidades (GENS) foram geradas.
Tentativas: {tentativa}.")
    return gen_pontos_cidades

# Função para calcular a distância total de um percurso
def calculate_distance(pontos_cidade, cromossomo):
    distance = 0
    for i in range(len(cromossomo) - 1):
        x1, y1 = pontos_cidade[cromossomo[i]]
        x2, y2 = pontos_cidade[cromossomo[i + 1]]
        distance += math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    # Retornar à cidade inicial
    x1, y1 = pontos_cidade[cromossomo[-1]]
    x2, y2 = pontos_cidade[cromossomo[0]]
    distance += math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return distance

# Função de fitness (minimizar a distância)
def fitness(pontos_cidade, cromossomo):
    return 1 / calculate_distance(pontos_cidade, cromossomo)

# Operador de cruzamento (Order Crossover - OX)
def crossover_ox(parent1, parent2):
    size = len(parent1)
    start, end = sorted(random.sample(range(1, size - 1), 2)) # Evitar incluir a cidade
inicial/final
    child = [None] * size
    child[0] = parent1[0] # Manter a cidade inicial
    child[-1] = parent1[-1] # Manter a cidade final
    child[start:end] = parent1[start:end]

    # Preencher os valores restantes com base no parent2
    pointer = end
    for city in parent2:
        if city not in child:
            if pointer >= size - 1:
                pointer = 1
            child[pointer] = city
            pointer += 1
    return child

```

```

# Operador de mutação (troca duas cidades)
def mutacao(cromossomo, probabilidade_mutacao=0.10):
    if random.random() < probabilidade_mutacao:
        idx1, idx2 = random.sample(range(1, len(cromossomo) - 1), 2) # Não trocar a
        cidade inicial/final
        cromossomo[idx1], cromossomo[idx2] = cromossomo[idx2], cromossomo[idx1]
    return cromossomo

# Seleção por torneio
def tournament_selection(populacao, fitness_values, k=3):
    selected = random.sample(list(zip(populacao, fitness_values)), k)
    return max(selected, key=lambda x: x[1])[0]

# Gerar população inicial
def gerar_populacao(pontos_cidade, tamanho_populacao_dados=100):
    start_city = random.randint(0, len(pontos_cidade) - 1) # Escolhe a cidade inicial
    aleatoriamente
    populacao = []
    for _ in range(tamanho_populacao_dados): # tamanho_populacao_dados -> Quantidade de
    cromossomos
        # Gera uma lista de cidades excluindo a cidade inicial
        other_cities = list(range(len(pontos_cidade)))
        other_cities.remove(start_city)
        random.shuffle(other_cities) # Embaralha as outras cidades
        # Cria o cromossomo com a cidade inicial no início e no fim
        cromossomo = [start_city] + other_cities + [start_city]
        populacao.append(cromossomo)
    return populacao

# Algoritmo Genético (taxa mutação 1%)
def rotina_algoritmo_genetico(pontos_cidade, tamanho_populacao_dados=100, geracoes=25000,
    taxa_mutacao=0.01):
    populacao = gerar_populacao(pontos_cidade, tamanho_populacao_dados)
    melhor_cromossomo = None
    best_fitness = -1
    best_distances = []
    # Quantidades mínimas garantidas por método
    qtd_crossover = int(tamanho_populacao_dados * 0.90)
    qtd_mutacao = max(int(tamanho_populacao_dados * taxa_mutacao), 1) # mínimo 1
    individuo
    # restante = tamanho_populacao_dados - (1 + qtd_crossover + qtd_mutacao) # 1 é para
    o elitismo (Sem
    uso por conta do loop while )
    for gen in range(geracoes):
        fitness_values = [fitness(pontos_cidade, chrom) for chrom in populacao]
        current_best_fitness = max(fitness_values)
        if current_best_fitness > best_fitness:

```



```

        best_fitness = current_best_fitness
        melhor_cromossomo =
populacao[fitness_values.index(current_best_fitness)]
        best_distances.append(1 / best_fitness)
        nova_populacao = [melhor_cromossomo] # Elitismo
        # Garantir mínimo 90% crossovers
        for _ in range(qtd_crossover):
            parent1 = tournament_selection(populacao, fitness_values)
            parent2 = tournament_selection(populacao, fitness_values)
            child = crossover_ox(parent1, parent2)
            nova_populacao.append(child)
        # Garantir mínimo 1% mutação direta
        for _ in range(qtd_mutacao):
            individuo = tournament_selection(populacao, fitness_values)
            mutante = mutacao(individuo[:], probabilidade_mutacao=1.0)
            # forçar mutação-não representa a % da população, e sim a
            probabilidade individual daquele cromossomo sofrer mutação
            nova_populacao.append(mutante)

        # Completar restante com crossover + mutação tradicional
        while len(nova_populacao) < tamanho_populacao_dados:
            parent1 = tournament_selection(populacao, fitness_values)
            parent2 = tournament_selection(populacao, fitness_values)
            child = crossover_ox(parent1, parent2)
            child = mutacao(child, taxa_mutacao)
            nova_populacao.append(child)
        populacao = nova_populacao
        # Primeiro pior caso/persusso
        if gen / 1 == 1:
            plot_route(pontos_cidade, melhor_cromossomo, f"Início (pior) percurso
entre {len(pontos_cidade)} cidades (Geração: {gen} - Distância = {1 / best_fitness})")
        # Log trace distancias
        if gen % 100 == 0:
            print(f"Geração {gen}: Melhor Distância = {1 / best_fitness}")
        return melhor_cromossomo, 1 / best_fitness, best_distances

# Função para plotar o percurso
def plot_route(pontos_cidade, cromossomo, title):
    # Coordenadas das cidades no percurso
    x = [pontos_cidade[i][0] for i in cromossomo]
    y = [pontos_cidade[i][1] for i in cromossomo]
    plt.figure(figsize=(10, 6))
    # Plotar cidade inicial/final (a primeira cidade, que é igual à última)
    plt.scatter(x[0], y[0], color='red', marker='o', s=100, label='I / F')
    plt.scatter(x[1:-1], y[1:-1], color='green')
    # Traçar percurso
    plt.plot(x, y, color='purple')

```

```

# Título e legenda
plt.title(title)
plt.legend()
plt.show()

# Execução principal
if __name__ == "__main__":
    # Gerar cidades
    pontos_cidades = gerar_pontos_cidades()
    # Executar o algoritmo genético principal
    best_route,best_distance,best_distances=rotina_algoritmo_genetico(pontos_cidades)
    # Plotar a evolução da distância
    plt.figure(figsize=(10, 6))
    plt.plot(best_distances, color='blue')
    plt.title("Evolução da Melhor Distância")
    plt.xlabel("Geração")
    plt.ylabel("Distância")
    plt.show()
    # Plotar a melhor solução
    plot_route(pontos_cidades,best_route,f"Melhor percurso entre {len(pontos_cidades)}
    cidades (Distância = {best_distance})")

```

2) Compare a representação de dois modelos vetoriais

Texto:

Lenda sobre a Gralha Azul:

A lenda da Gralha-Azul teve origem no estado do Paraná, na região sul do Brasil. Conta-se que uma gralha recebeu da Mãe Natureza um pinhão para matar sua fome. A ave, que ficou muito feliz e satisfeita com o alimento, comeu metade do pinhão e enterrou a outra para se alimentar depois, mas se esqueceu onde havia escondido o restante do fruto.

Algum tempo depois a gralha percebeu que um lindo pinheiro começou a brotar na floresta e decidiu cuidar dele. Conforme foi crescendo, o pinheiro começou a dar frutos com os quais a ave continuou a se alimentar para saciar sua fome. Assim como da primeira vez em que recebeu um pinhão da Mãe Natureza, a gralha comia uma parte do fruto e enterrava o restante, sempre se esquecendo de onde havia escondido o alimento.

Dessa forma, a gralha plantou uma floresta inteira de araucárias pela região sul. Como recompensa por sua bela atitude, a Mãe Natureza decidiu modificar a cor do pássaro que antes era pardo, deixando-o com a coloração azul. Com isso, a Gralha-Azul se tornou uma belíssima ave que se diferencia de todas as outras da floresta.

```

sentencas = [
"A lenda da Gralha-Azul teve origem no estado do Paraná, na região sul do Brasil.", #S1
"Conta-se que uma gralha azul recebeu da Mãe Natureza um pinhão para matar sua fome.", #
S2
"A ave, que ficou muito feliz e satisfeita com o alimento, comeu metade do pinhão e
enterrou a outra para se alimentar", # S3
"mas se esqueceu onde havia escondido o restante do fruto", # S4
"Algum tempo depois a gralha percebeu que um lindo pinheiro começou a brotar na floresta e
decidiu cuidar dele", # S5
"Conforme foi crescendo, o pinheiro começou a dar frutos com os quais a ave continuou a se
alimentar para saciar sua fome", # S6

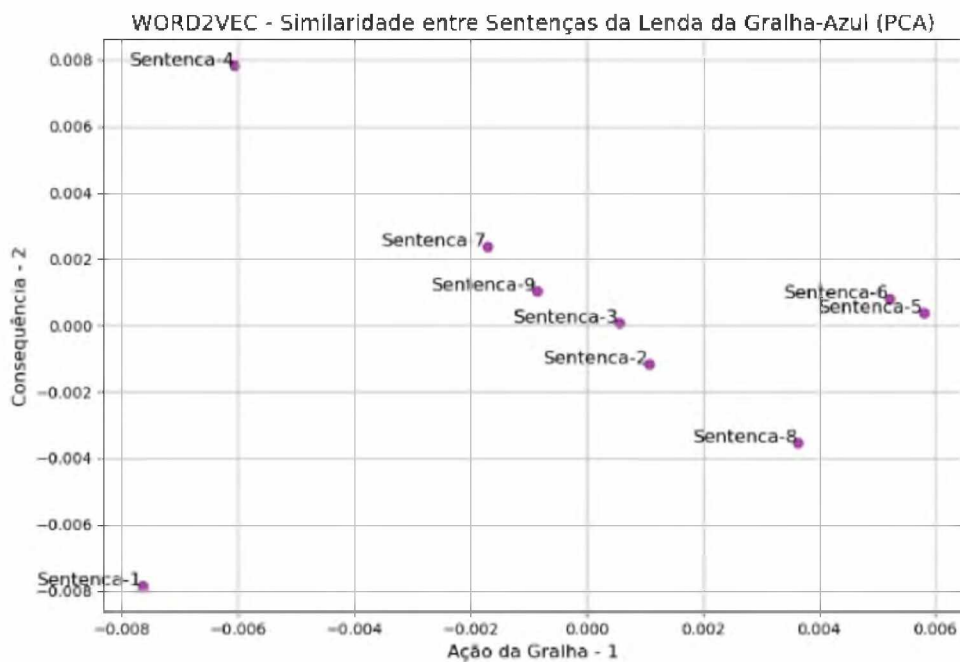
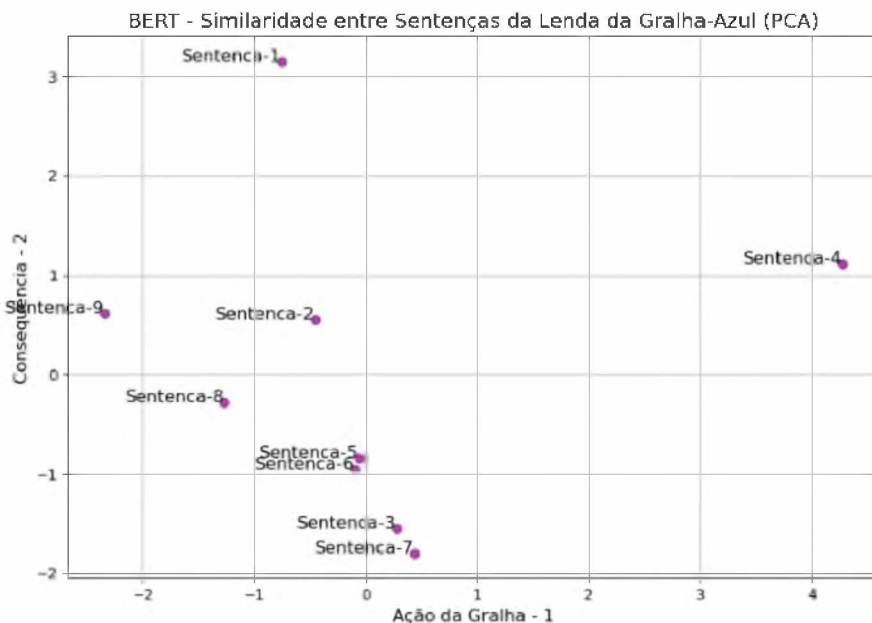
```

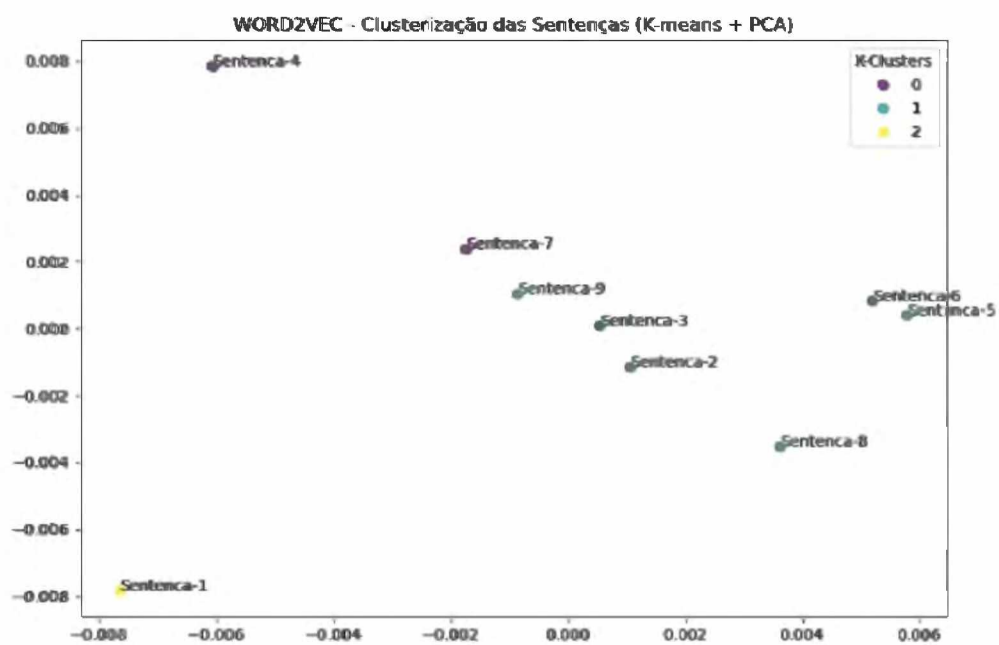
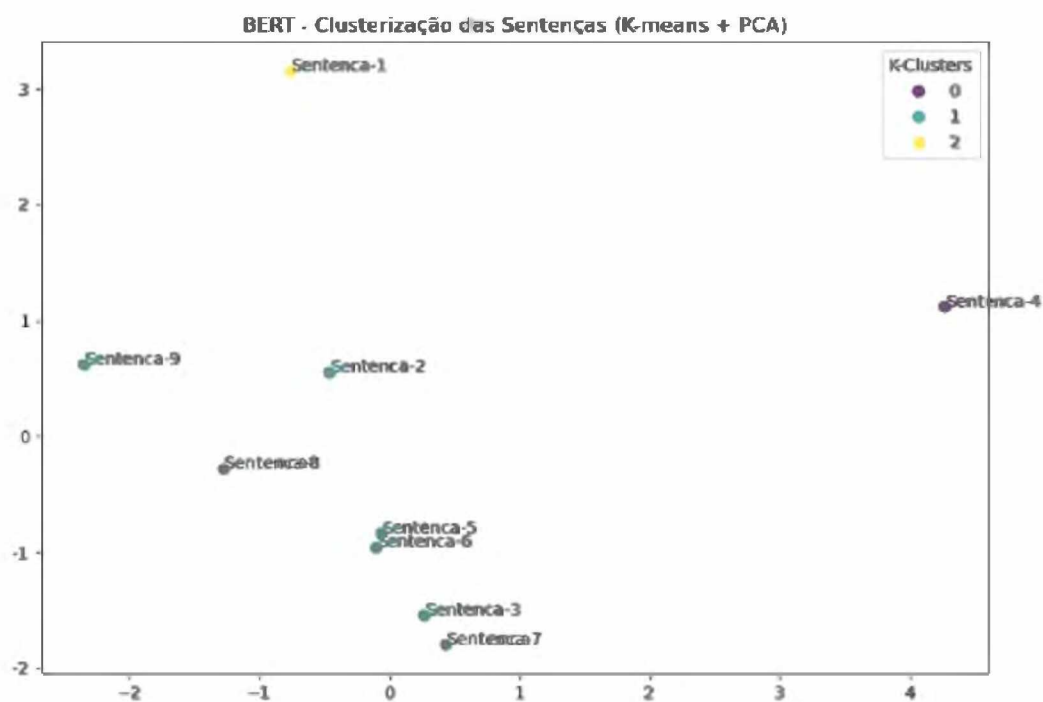
"Assim como da primeira vez em que recebeu um pinhão da Mãe Natureza, a gralha comia uma parte do fruto e enterrava o restante, sempre se esquecendo de onde havia escondido o alimento.", # S7

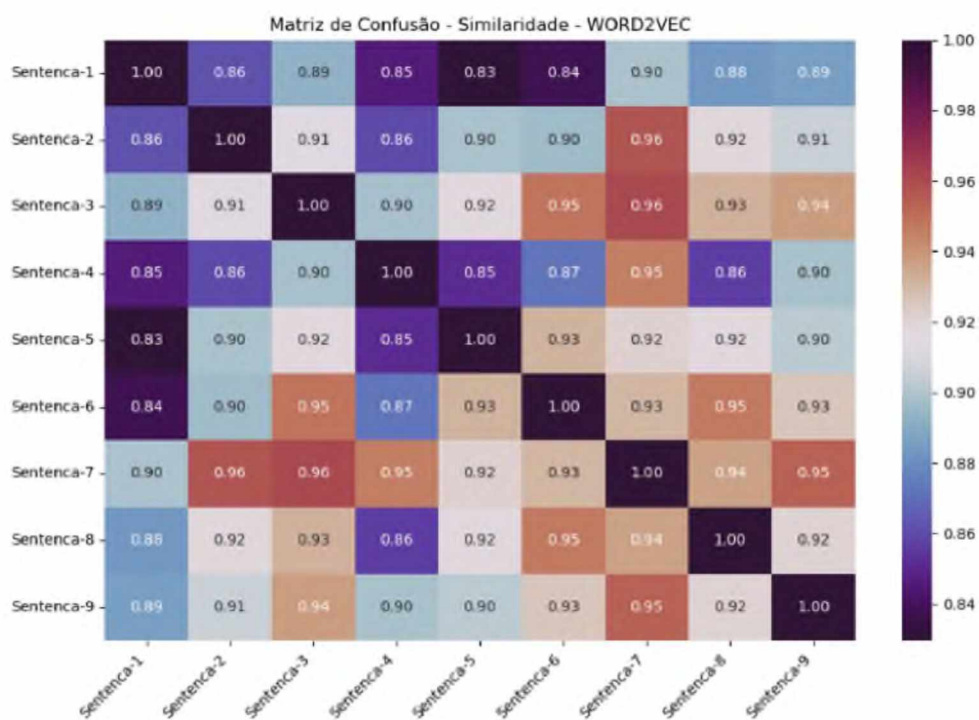
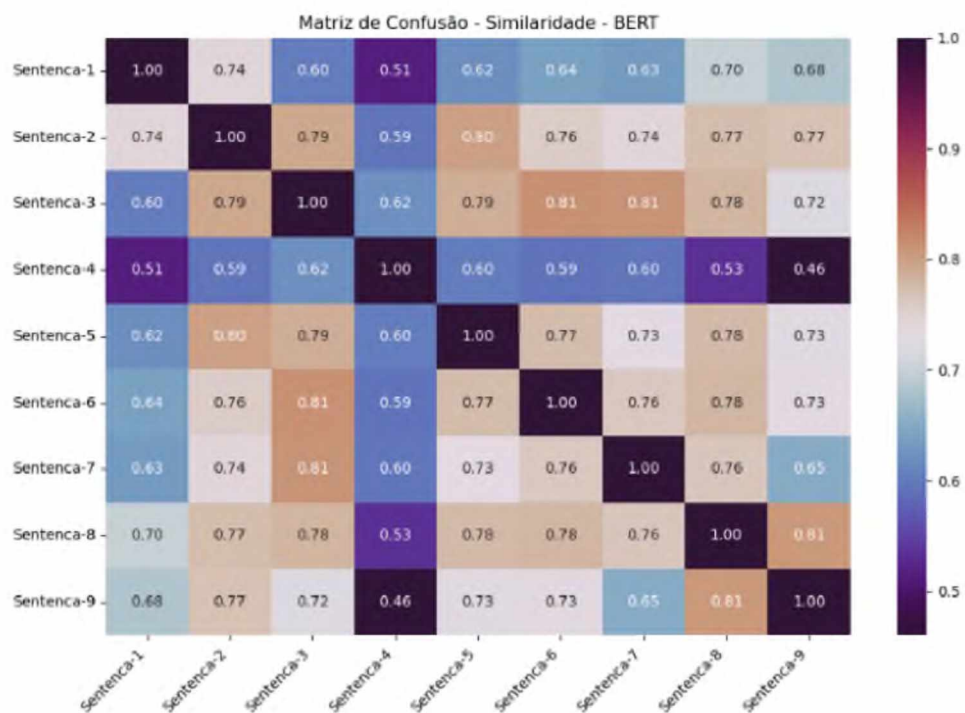
"Como recompensa por sua bela atitude, a Mãe Natureza decidiu modificar a cor do pássaro que antes era pardo, deixando-o com a coloração azul.", # S8

"Com isso, a Galha-Azul se tornou uma belíssima ave que se diferencia de todas as outras da floresta." #S9

]







```
from sentence_transformers import SentenceTransformer
from sklearn.decomposition import PCA
```

```

from transformers import AutoModel, AutoTokenizer
# Mais preciso para o idioma português
import spacy
from gensim.models import Word2Vec
import matplotlib.pyplot as plt
import numpy as np
import os
import torch
import string
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.cluster import KMeans
import seaborn as sns
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
device = "cuda" if torch.cuda.is_available() else "cpu"

sentencas = [
    "A lenda da Gralha-Azul teve origem no estado do Paraná, na região sul do Brasil.", #S1
    "Conta-se que uma gralha azul recebeu da Mãe Natureza um pinhão para matar sua fome.", #
    S2
    "A ave, que ficou muito feliz e satisfeita com o alimento, comeu metade do pinhão e
    enterrou a outra para se alimentar", # S3
    "mas se esqueceu onde havia escondido o restante do fruto", # S4
    "Algum tempo depois a gralha percebeu que um lindo pinheiro começou a brotar na floresta e
    decidiu cuidar dele", # S5
    "Conforme foi crescendo, o pinheiro começou a dar frutos com os quais a ave continuou a se
    alimentar para saciar sua fome", # S6
    "Assim como da primeira vez em que recebeu um pinhão da Mãe Natureza, a gralha comia uma
    parte do fruto e enterrava o restante, sempre se esquecendo de onde havia escondido o
    alimento.", # S7
    "Como recompensa por sua bela atitude, a Mãe Natureza decidiu modificar a cor do pássaro
    que antes era pardo, deixando-o com a coloração azul.", # S8
    "Com isso, a Gralha-Azul se tornou uma belíssima ave que se diferencia de todas as outras
    da floresta." #S9
]

# Dimensão do Embedding 768
model_name = 'neuralmind/bert-base-portuguese-cased'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name).to(device)

# Dimensão do Embedding 768
# model = SentenceTransformer('neuralmind/bert-base-portuguese-cased', device=device,
# local_files_only=True)

# Dimensão do Embedding 384
# model = SentenceTransformer("sentence-transformers/all-MiniLM-L6-v2", device=device,
# local_files_only=True)

# BERT embedding

```

```

def get_embeddings(texts):
    inputs=tokenizer(texts,padding=True,truncation=True,return_tensors='pt').to(device)
    with torch.no_grad():
        outputs = model(**inputs)
        embeddings = outputs.last_hidden_state.mean(dim=1) # Pooling
    return embeddings.cpu().numpy()
bert_embeddings = get_embeddings(sentencas)

# Gerar embeddings (vetores de 768 dimensões)
# embeddings = model.encode(sentencas)
# Word2Vec embedding
# pipeline de processamento de texto que inclui várias etapas automáticas
# Tokenização (dividir o texto em palavras/símbolos)
# Tagging (classificar palavras em verbos, substantivos, etc.)
# Lemmatização (reduzir palavras à sua forma base)
# Reconhecimento de entidades nomeadas (identificar nomes, lugares, etc.)
nlp = spacy.load("pt_core_news_sm") #nlp = Natural Language Processing
# print(type(nlp))

def tokenize_with_spacy(text):
    return [token.text for token in nlp(text)]
tokenized_texts = [tokenize_with_spacy(t) for t in sentencas]
# sg - Escolhe o algoritmo Skip-gram (em vez de CBOW).
# Skip-gram (sg=1):
# Melhor para pequenos datasets ou palavras raras. Prevê palavras vizinhas a partir de uma palavra-alvo.
# CBOW (sg=0):
# Melhor para grandes datasets. Prevê uma palavra-alvo a partir do contexto.
w2v_model = Word2Vec(vector_size=300, window=7, min_count=2, workers=4, sg=1, alpha=0.02, min_alpha=0.0001)
w2v_model.build_vocab(tokenized_texts)
w2v_model.train(tokenized_texts, total_examples=len(tokenized_texts), epochs=50)

# print(w2v_model.wv["gralha"]) # Exibe o vetor para a palavra 'gralha'
def get_word2vec_vector(text):
    words = tokenize_with_spacy(text)
    vectors = [w2v_model.wv[w] for w in words if w in w2v_model.wv]
    return np.mean(vectors, axis=0) if vectors else np.zeros(w2v_model.vector_size)

word2vec_vec = [get_word2vec_vector(sentenca) for sentenca in sentencas]
print(word2vec_vec)

def plot_bert_w2v(tensorvecs, title):
    pca = PCA(n_components=2)

```

```

embeddings_2d = pca.fit_transform(tensorvecs)
plt.figure(figsize=(10, 8))
plt.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1], color='purple', alpha=0.7)

# Adicionar rótulos das sentenças
for i, txt in enumerate(range(len(sentencas))):
    plt.annotate(f"Sentenca-{i+1}", (embeddings_2d[i, 0], embeddings_2d[i, 1]),
                fontsize=12, ha='right')

plt.title(title, fontsize=14)
plt.xlabel("Ação da Galha - 1", fontsize=12)
plt.ylabel("Consequência - 2", fontsize=12)
plt.grid(True)
plt.show()

plot_bert_w2v(bert_embeddings, "BERT - Similaridade entre Sentenças da Lenda da
Galha-Azul (PCA)")
plot_bert_w2v(word2vec_vec, "WORD2VEC - Similaridade entre Sentenças da Lenda da Galha-
Azul (PCA)")

# Matriz de similaridade entre todas as sentenças
# BERT
# similarity_matrix = cosine_similarity(bert_embeddings)
# Exibir os pares mais similares
# for i in range(len(similarity_matrix)):
# for j in range(i+1, len(similarity_matrix)):
# print(f"Sentenca-{i+1} <::> Sentenca-{j+1}: {similarity_matrix[i][j]:.2f}")

# K-Means - BERT e Word2Vec
def plot_kmeans_bert_w2v(tensorvecs, title):
    pca = PCA(n_components=2)
    embeddings_2d = pca.fit_transform(tensorvecs)
    # Definir número de clusters (ajuste conforme necessidade)
    n_clusters = 3
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    clusters = kmeans.fit_predict(tensorvecs)

    # Adicionar rótulos ao gráfico PCA
    plt.figure(figsize=(10, 8))

scatter=plt.scatter(embeddings_2d[:,0],embeddings_2d[:,1],c=clusters,cmap='viridis',alpha=
0.7)

for i, txt in enumerate(range(len(sentencas))):

```



```

plt.annotate(f"Sentenca-{i+1}", (embeddings_2d[i, 0], embeddings_2d[i, 1]),
            fontsize=10)

plt.legend(*scatter.legend_elements(), title="K-Clusters")
plt.title(title)
plt.show()

plot_kmeans_bert_w2v(bert_embeddings, "BERT-Clusterização das Sentenças (K-means + PCA)")
plot_kmeans_bert_w2v(word2vec_vec, "WORD2VEC-Clusterização das Sentenças (K-means + PCA)")

# Gerar matriz de confusão
def plot_heatmap_similarity(emb_vector, title):
    similarity_matrix = cosine_similarity(emb_vector)

    # Plotar heatmap
    plt.figure(figsize=(10, 8))
    heatmap = sns.heatmap(similarity_matrix,
                          annot=True, fmt=".2f",
                          cmap="twilight_shifted",
                          xticklabels=[f"Sentenca-{i+1}" for i in range(len(sentencas))],
                          yticklabels=[f"Sentenca-{i+1}" for i in range(len(sentencas))],
                          )

    heatmap.set_xticklabels(
        heatmap.get_xticklabels(),
        rotation=45,
        ha='right',
        fontsize=10
    )
    plt.title(title)
    plt.tight_layout()
    plt.show()

plot_heatmap_similarity(bert_embeddings, "Matriz de Confusão - Similaridade - BERT")
plot_heatmap_similarity(word2vec_vec, "Matriz de Confusão - Similaridade - WORD2VEC")

```