

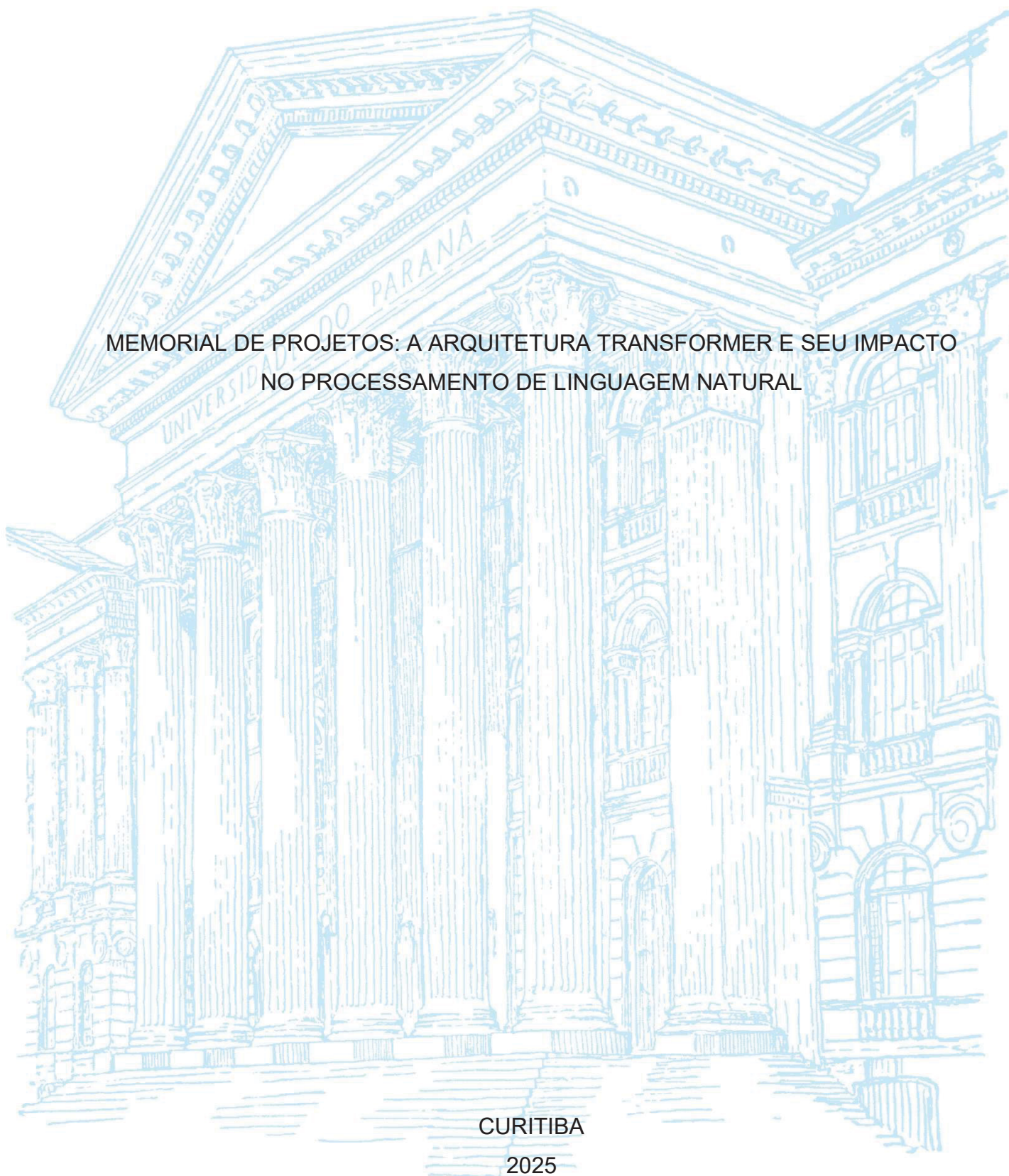
UNIVERSIDADE FEDERAL DO PARANÁ

VALÉRIO VIÉGAS WITTLER

MEMORIAL DE PROJETOS: A ARQUITETURA TRANSFORMER E SEU IMPACTO
NO PROCESSAMENTO DE LINGUAGEM NATURAL

CURITIBA

2025



VALÉRIO VIÉGAS WITTLER

MEMORIAL DE PROJETOS: A ARQUITETURA TRANSFORMER E SEU IMPACTO
NO PROCESSAMENTO DE LINGUAGEM NATURAL

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 VALÉRIO VIÉGAS WITTLER, intitulada: MEMORIAL DE PROJETOS: A ARQUITETURA TRANSFORMER E SEU IMPACTO NO PROCESSAMENTO DE LINGUAGEM NATURAL, 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, 24 de Setembro de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

A arquitetura Transformer, proposta por Vaswani et al. (2017), revolucionou o Processamento de Linguagem Natural (PLN) ao substituir RNNs por mecanismos de autoatenção que geram vetores *Query*, *Key* e *Value* para calcular a *Scaled Dot-Product Attention*. Com atenção multi-cabeças, o modelo captura diversas relações entre tokens em paralelo, enquanto codificações posicionais de seno e cosseno mantêm a ordem sequencial. Blocos de *encoder* e *decoder* incluem subcamadas de autoatenção e redes *feed-forward* com ativação ReLU. Essa estrutura fundamentou o desenvolvimento de BERT, focado em compreensão de texto, e GPT, voltado à geração autorregressiva, redefinindo o estado da arte em PLN.

Palavras-chave: Transformador; autoatenção; atenção multi-cabeças; codificações posicionais; BERT; GPT.

ABSTRACT

The Transformer architecture, proposed by Vaswani et al. (2017), revolutionized Natural Language Processing (NLP) by replacing RNNs with self-attention mechanisms that generate Query, Key, and Value vectors to calculate Scaled Dot-Product Attention. With multi-head attention, the model captures diverse token relationships in parallel, while sine and cosine positional encodings maintain sequential order. Encoder and decoder blocks include self-attention sublayers and feed-forward networks with ReLU activation. This structure laid the foundation for developing BERT, focused on text comprehension, and GPT, aimed at autoregressive generation, redefining the state-of-the-art in NLP.

Keywords: Transformer; self-attention; multi-head attention; positional encodings; BERT; GPT.

SUMÁRIO

1 PARECER TÉCNICO.....	7
1.1 INTRODUÇÃO	7
1.2 A ARQUITETURA TRANSFORMER.....	8
1.3 MECANISMO DE AUTOATENÇÃO (SELF-ATTENTION)	9
1.4 ATENÇÃO MULTI-CABEÇAS (MULTI-HEAD ATTENTION).....	10
1.5 CODIFICAÇÕES POSICIONAIS (POSITIONAL ENCODINGS)	10
1.6 CAMADAS FEED-FORWARD (POSITION-WISE FEED-FORWARD NETWORKS)	11
1.7 IMPACTO E APLICAÇÕES.....	12
1.8 CONCLUSÃO.....	13
REFERÊNCIAS.....	14
APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL	15
APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA	23
APÊNDICE 3 – LINGUAGEM R	51
APÊNDICE 4 – ESTATÍSTICA APLICADA I	57
APÊNDICE 5 – ESTATÍSTICA APLICADA II.....	67
APÊNDICE 6 – ARQUITETURA DE DADOS	79
APÊNDICE 7 – APRENDIZADO DE MÁQUINA.....	111
APÊNDICE 8 – DEEP LEARNING	119
APÊNDICE 9 – BIG DATA.....	136
APÊNDICE 10 – VISÃO COMPUTACIONAL	141
APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	165
APÊNDICE 12 – GESTÃO DE PROJETOS DE IA.....	175
APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL	180
APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING	198
APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL	206

1 PARECER TÉCNICO

1.1 INTRODUÇÃO

A Inteligência Artificial (IA) vem conquistando avanços notáveis nas últimas décadas, especialmente na área do Processamento de Linguagem Natural (PLN), que busca capacitar máquinas a compreender, interpretar e gerar linguagem humana de forma eficaz. Durante muitos anos, o estado-da-arte no PLN estava definido por arquiteturas baseadas em Redes Neurais Recorrentes (*RNNs*), como as *Long Short-Term Memory (LSTM)* (Hochreiter et al., 1997) e *Gated Recurrent Units (GRU)* (Cho et al., 2014). Esses modelos processam os dados de forma sequencial, token a token, o que, embora seja eficaz para capturar relações locais, apresenta limitações importantes para capturar dependências de longo alcance devido a problemas como o desaparecimento ou explosão do gradiente e a dificuldade de paralelização do grande volume de cálculos necessários para o treinamento.

Nesse contexto, em 2017, surgiu inovador artigo "*Attention Is All You Need*" (Vaswani et al., 2017) que redefiniu o paradigma das redes neurais para PLN. Uma nova abordagem foi apresentada e serviu como base para uma prolífica e intensa sucessão de conquistas na área. Sua principal inovação consiste no uso do chamado mecanismo de atenção, que proporciona uma maneira extremamente eficiente para o modelo entender e focar nas partes relevantes da entrada. O mecanismo de atenção compõe a arquitetura *Transformer*, também apresentada no artigo de Vaswani et al. (2017).

Na arquitetura *Transformer*, os autores propuseram dispensar completamente a recorrência e convoluções, que até então eram os mecanismos mais utilizados nessas tarefas (Vaswani et al., 2017). Em seu lugar, o mecanismo de atenção conseguia oferecer uma nova abordagem capaz de superar diversas limitações das *RNNs* e atingir novos patamares de desempenho em diversas tarefas de PLN. A capacidade de processar sequências de qualquer tamanho sem depender de uma memória fixa é uma das principais vantagens da arquitetura *Transformer*, tornando-o mais robusto e adaptável.

1.2 A ARQUITETURA TRANSFORMER

A arquitetura de um *Transformer* é concebida em torno de dois componentes principais que trabalham em conjunto: o *encoder* e o *decoder*. Ambos são compostos por uma pilha de N blocos idênticos, cujos elementos centrais são as camadas de autoatenção e as redes *feed-forward* (Vaswani et al., 2017).

O *encoder* é o primeiro elemento da arquitetura e recebe a sequência de entrada (uma frase, um áudio ou qualquer outro dado sequencial). Ele começa convertendo cada elemento da sequência em uma representação numérica correspondente em forma de vetores, em um processo chamado de tokenização. Esses vetores são processados ao longo de várias camadas, onde está presente o mecanismo de atenção que fica responsável por identificar as relações e a importância de cada parte da entrada. O resultado de todo esse processamento é uma representação mais compacta da sequência original, que captura suas características mais importantes. Essa representação consiste na entrada do próximo componente: o *decoder*.

No *encoder* cada bloco possui duas subcamadas principais: uma de autoatenção multi-cabeças e uma rede *feed-forward*. Conexões residuais (Hinton et al., 2016) são empregadas em torno de cada uma das duas subcamadas, seguidas por normalização de camada (Hinton et al., 2016). A saída de cada subcamada será:

$$\text{LayerNorm}(x + \text{Sublayer}(x))$$

onde:

$\text{Sublayer}(x)$ é a função implementada pela própria subcamada.

O *decoder* tem como tarefa gerar a sequência de saída a partir da representação compacta gerada pelo *encoder*. A construção da resposta acontece de maneira autorregressiva, passo a passo, com a utilização de cada dado gerado como uma nova entrada. Essa realimentação da saída permite que a geração da resposta seja incremental sem perder o contexto, o que garante um resultado final mais coerente e completo.

No *decoder*, além das duas subcamadas presentes nos blocos do *encoder*, há uma terceira subcamada, chamada de "atenção codificador-decodificador", que proporciona ao decodificador a capacidade de focar em partes relevantes da sequência de entrada. Essa subcamada realiza atenção multi-cabeças sobre a saída da pilha do *encoder*. Semelhante ao *encoder*, são aplicadas conexões residuais e normalização de camada a cada subcamada. Já a camada de autoatenção é modificada para evitar que as posições considerem as posições subsequentes, garantindo que a predição de uma posição dependa apenas das saídas anteriores a ela. Esse mecanismo é chamado mascaramento e é feito atribuindo um valor muito baixo aos pesos de atenção das posições futuras e quando esses valores passam pela função *softmax*, eles se transformam em zero.

1.3 MECANISMO DE AUTOATENÇÃO (SELF-ATTENTION)

O componente mais importante da arquitetura *Transformer* é o mecanismo de autoatenção (*self-attention*), pois ele permite ao modelo mensurar a importância de diferentes *tokens* em uma sequência, processando cada palavra individualmente. No lugar do estado oculto, propagado sequencialmente nas RNNs, a autoatenção calcula uma representação para cada token de acordo com o contexto global da sequência de entrada.

Para cada token de entrada, o mecanismo de atenção gera três informações, representadas em vetores:

1. Vetor de Consulta (*Query* - Q): Representa a palavra atual que o modelo está processando. É a "pergunta" que a palavra faz às outras.
2. Vetor de Chave (*Key* - K): Representa as outras palavras na frase. É como uma "etiqueta" ou "código" que as outras palavras usam para identificar seu conteúdo.
3. Vetor de Valor (*Value* - V): Contém a informação real, o "conteúdo" de cada palavra. É o que o modelo vai usar para formar a representação final.

A esses três vetores Q, K e V, estão associadas matrizes de pesos W_Q , W_K , W_V treináveis, ou seja, o modelo ajusta esses pesos durante o treinamento para que a autoatenção seja cada vez melhor em identificação as relações entre os *tokens*.

A operação de atenção é frequentemente descrita como "*Scaled Dot-Product Attention*" e é calculada como:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

onde:

d_k = dimensão dos vetores de chave K e consulta Q

Q = *Query*

K = *Key*

V = *Value*

A divisão por d_k é um fator de escalonamento que previne que os produtos escalares se tornem muito grandes, o que poderia levar a gradientes muito pequenos na função *softmax*, dificultando o aprendizado. O mecanismo de atenção foi inspirado em trabalhos anteriores, como o de Bahdanau et al. (2014), que introduziram a atenção em modelos de tradução automática baseados em RNNs.

1.4 ATENÇÃO MULTI-CABEÇAS (MULTI-HEAD ATTENTION)

Vaswani et al. (2017) descobriram que é benéfico projetar linearmente as *queries*, chaves e valores várias vezes com diferentes projeções aprendidas. Como são projeções independentes é possível paralelizar a execução gerando várias saídas que no final são concatenadas linearmente, resultando na saída final.

Cada cabeça de atenção pode aprender a focar em diferentes aspectos das relações entre os *tokens*, como relações sintáticas ou semânticas distintas, enriquecendo o contexto, pois cada cabeça pode representar diferentes subespaços de representação em diferentes posições trazendo uma maior riqueza contextual ao modelo, segundo Vaswani et al. (2017).

1.5 CODIFICAÇÕES POSICIONAIS (POSITIONAL ENCODINGS)

Enquanto nas arquiteturas das RNNs a ordem dos *tokens* é naturalmente preservada pelo estado oculto, enquanto elas processam um *token* por vez, o mesmo não acontece na arquitetura *Transformer*, pois ela processa os tokens em paralelo. Assim, faz-se necessário representar a ordem nessa arquitetura.

Codificações posicionais (*positional encodings*) são adicionadas a cada token na sequência antes da entrada no primeiro *encoder*.

Como a arquitetura *Transformer* não utiliza recorrência nem convoluções, ela não possui uma maneira inerente de levar em consideração a ordem dos *tokens* na sequência. Para resolver essa limitação, informações sobre a posição relativa ou absoluta dos *tokens* são injetadas no modelo através de codificações posicionais (*positional encodings*). Essas codificações são adicionadas aos vetores de *embeddings* de entrada na base das pilhas do codificador e do decodificador.

Vaswani et al. (2017) propuseram o uso de funções seno e cosseno de diferentes frequências:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

onde:

pos = posição do *token* na sequência

i = índice da dimensão do vetor de *embedding*,

d_{model} = própria dimensão do *embedding*.

As codificações posicionais são uma forma de fornecer memória de posição ao *Transformer*. Elas garantem que o modelo entenda a ordem e a estrutura da frase, mesmo processando as palavras em paralelo.

1.6 CAMADAS FEED-FORWARD (POSITION-WISE FEED-FORWARD NETWORKS)

Após a autoatenção ter gerado uma representação de cada palavra com base no contexto global, o modelo precisa processar essa informação de forma individual em cada posição, processando refinando essa representação.

Cada bloco do *encoder* e *decoder* contém uma rede *feed-forward* totalmente conectada que consiste em duas transformações lineares e é aplicada a cada *token* separadamente. Ela pode ser representada pela seguinte função:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

onde:

x = vetor de entrada representando cada token

W_1 = matriz de pesos da primeira camada linear

b_1 = viés da primeira camada linear.

$\max(0, xW_1 + b_1)$ = função de ativação *ReLU* (*Rectified Linear Unit*).

W_2 = matriz de pesos da segunda camada linear.

b_2 = viés da segunda camada linear.

Essa FFN adiciona capacidade não linear ao modelo, permitindo que ele aprenda interações mais complexas entre as características de cada *token*.

1.7 IMPACTO E APLICAÇÕES

A apresentação da arquitetura *Transformer* serviu de base para uma onda de inovações no PLN. Devido à sua versatilidade, diversos novos modelos e arquiteturas inspiradas por ela alcançaram resultados que redefiniram o estado da arte em diversas tarefas de PLN e se tornaram referência na indústria e na academia.

Um dos exemplos é o BERT (*Bidirectional Encoder Representations from Transformers*) (Devlin et al., 2018). BERT utiliza apenas a porção do *encoder* do Transformer e é pré-treinado em grandes volumes de texto não rotulado usando duas tarefas: Modelagem de Linguagem Mascarada (*Masked Language Modeling - MLM*) e Previsão da Próxima Sentença (*Next Sentence Prediction - NSP*). Segundo Devlin et al. (2018) o BERT é conceitualmente simples e obteve novos resultados estado da arte diversas tarefas de processamento de linguagem natural. Após o pré-treinamento, BERT pode ser ajustado (*fine-tuned*) com uma pequena quantidade de dados rotulados para tarefas específicas, como classificação de texto, resposta a perguntas e reconhecimento de entidades nomeadas.

Outra família de modelos baseada na arquitetura Transformer são os Modelos Generativos Pré-treinados (Generative Pre-trained Transformers - GPT), desenvolvidos pela OpenAI (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020). Diferentemente do BERT, os modelos GPT utilizam apenas a porção do *decoder* do Transformer e são pré-treinados primariamente para a tarefa de modelagem de linguagem, ou seja, prever o próximo *token* em uma sequência. Esses

modelos demonstraram uma capacidade notável de gerar texto coerente e contextualmente relevante, além de realizar tarefas com poucas ou nenhuma amostra de treinamento (*few-shot e zero-shot learning*). O GPT-3 (Brown et al., 2020), com seus 175 bilhões de parâmetros, exemplifica o poder da escala em modelos baseados em *Transformer*.

As aplicações dos modelos baseados na arquitetura *Transformer* são vastas e com aplicação em diversas áreas, incluindo geração e compreensão de texto, análise de sentimentos, tradução automática, entre outros.

1.8 CONCLUSÃO

A arquitetura *Transformer*, proposta por Vaswani et al. (2017), representou um grande avanço no campo do aprendizado de máquina, especialmente no Processamento de Linguagem Natural. Ao substituir a recorrência por mecanismos de atenção, o *Transformer* resolveu problemas de modelos sequenciais anteriores, a dificuldade de paralelização e de captura de dependências de longo alcance

O impacto do *Transformer* é verificado pelo surgimento de modelos como BERT e GPT, que redefiniram o estado da arte em inúmeras tarefas e impulsionaram avanços significativos no PLN. A flexibilidade e escalabilidade do mecanismo de atenção continua a inspirar novas pesquisas e aplicações.

REFERÊNCIAS

- BA, J. L. et al. Layer normalization. **arXiv preprint**. 21 jul. 2016. DOI: arXiv:1607.06450v1.
- BAHDANAU, D. et al. Neural machine translation by jointly learning to align and translate. **arXiv preprint**. 1 set. 2014. DOI: 10.48550/arXiv.1409.0473.
- BROWN, T. B. et al. Language Models are Few-Shot Learners. **arXiv preprint**. 28 mai. 2020. DOI: arXiv:10.48550/arXiv.2005.14165
- CHO, K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. **arXiv preprint**. 3 jun. 2014. DOI:10.48550/arXiv.1406.1078.
- DEVLIN, J. et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **arXiv preprint**. 11 out. 2018. DOI: 10.48550/arXiv.1810.04805
- HOCHREITER, S. et al. "Long Short-Term Memory," in **Neural Computation**, vol. 9, no. 8, pp. 1735-1780, 15 nov. 1997, DOI: 10.1162/neco.1997.9.8.1735.
- RADFORD, A. et al. Improving language understanding by generative pre-training. **OpenAI blog**. 2018. Disponível em: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf. Acesso em: 22 maio 2025.
- RADFORD, A. et al. Language models are unsupervised multitask learners. **OpenAI blog**, v. 1, n. 8, 2019. Disponível em: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Acesso em: 22 maio 2025.
- VASWANI, A. et al. Attention Is All You Need. **arXiv preprint**, 12 jun. 2017. DOI: arXiv:1706.03762.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 CHATGPT

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

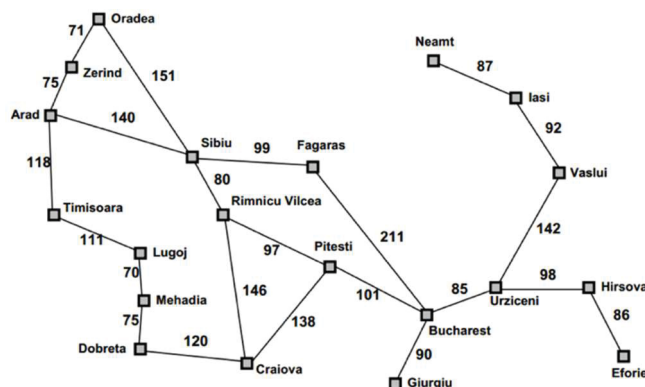
2 BUSCA HEURÍSTICA

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 LÓGICA

Verificar se o argumento lógico é válido.

Se as uvas caem, então a raposa as come

Se a raposa as come, então estão maduras

As uvas estão verdes ou caem

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem

q: a raposa come as uvas

r: as uvas estão maduras

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

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

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

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

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

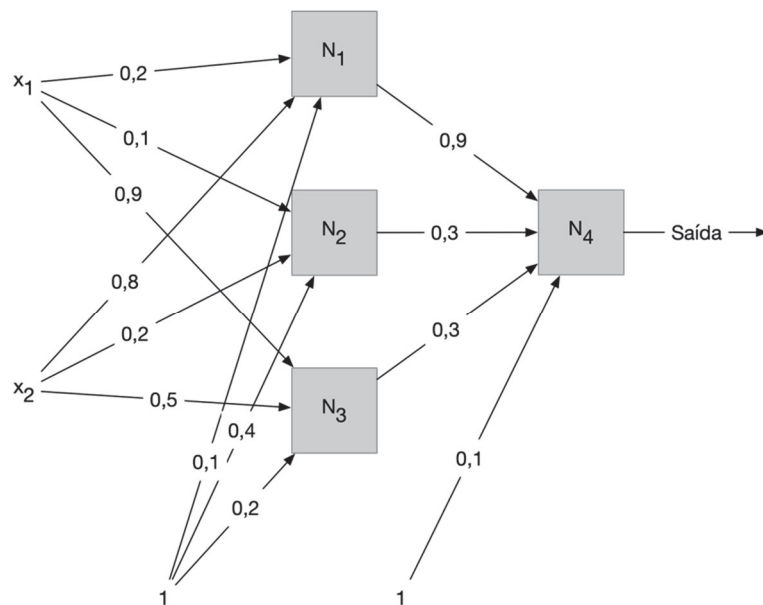
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 REDES NEURAIS ARTIFICIAIS

Seja a RNA da figura abaixo.



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

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

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

B – RESOLUÇÃO

1.a) Segundo o ChatGPT: "Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e algoritmos que podem realizar tarefas que normalmente exigiriam inteligência humana. Isso inclui coisas como reconhecimento de fala, reconhecimento de padrões visuais, tomada de decisões, tradução de idiomas, entre outras.

Os sistemas de IA são capazes de aprender com dados, encontrar padrões e realizar tarefas específicas de forma autônoma, muitas vezes melhorando sua própria performance ao longo do tempo. Existem várias abordagens para implementar IA incluindo machine learning, redes neurais artificiais, algoritmos genéticos, lógica fuzzy, entre outros.

A IA tem aplicações em uma ampla variedade de setores, incluindo medicina, finanças, manufatura, transporte, entretenimento e muito mais. À medida que a tecnologia continua a evoluir, a IA está se tornando cada vez mais integrada em nossas vidas cotidianas, impactando a forma como trabalhamos, nos comunicamos e interagimos com o mundo ao nosso redor. "

1.b) Análise da definição de acordo com as quatro Abordagens da Inteligência Artificial:

1. Agir como Humanos: a definição de IA fornecida se aproxima da abordagem "Agir como Humanos", segundo Russell e Norvig (2022), que busca imitar o comportamento humano de forma convincente. No entanto, a definição apresentada é mais ampla, pois enfatiza a realização de tarefas, autonomia, aprendizado e tomada de decisões, aspectos que não são necessariamente o foco principal da abordagem "Agir como Humanos".

2. Pensar como Humanos: a segunda abordagem, "Pensar como Humanos", concentra-se na reprodução fiel dos processos de pensamentos humanos, focando no "como fazer". A definição fornecida, por outro lado, dá ênfase aos resultados e ao "fazer a coisa certa".

3. Pensar Racionalmente: a abordagem "Pensar Racionalmente" destaca a inferência correta como base da inteligência. No entanto, a inferência correta nem sempre representa a totalidade da racionalidade. Em algumas situações, a inferência correta pode ser impossível, exigindo a tomada de decisões mesmo sem total certeza. Além disso, existem formas de agir racionalmente que não se originam de inferências lógicas, como os atos reflexos humanos.

4. Agir Racionalmente: a definição de IA fornecida se identifica mais com a abordagem "Agir Racionalmente", segundo Russell e Norvig (2022). Essa abordagem define um agente racional como aquele que busca alcançar o melhor resultado ou, em situações de incerteza, o melhor resultado esperado. A definição de IA destaca a capacidade de realizar tarefas, aprender, tomar decisões e agir autonomamente, características que se alinham com a perspectiva do agente racional. Essa abordagem é uma das mais importantes e influentes no campo da IA, sendo fundamental para o desenvolvimento de muitos sistemas de IA bem-sucedidos.

Russell e Norvig (2022) também definem as características esperadas de um agente racional:

- Operar sob controle autônomo;
- Perceber seu ambiente;
- Persistir por um período prolongado;
- Adaptar-se a mudanças;
- Criar e perseguir metas.

A abordagem "Agir Racionalmente" é a mais geral das quatro e tem se mostrado dominante por se basear na ideia de que a inteligência pode ser caracterizada como a capacidade de agir de forma racional para alcançar objetivos.

Referências:

Russell, Stuart, J. e Peter Norvig. Inteligência Artificial: Uma Abordagem Moderna. 4ª edição. Porto Alegre: Grupo GEN, 2022. Disponível em: Minha Biblioteca.

1.c) O ChatGPT é um chatbot, da empresa OpenAI, que consiste em uma ferramenta capaz de realizar várias tarefas como: gerar códigos em linguagens de programação, escrever artigos, resolver problemas, sugerir ações, gerar e interpretar imagens, áudios, vídeos, entre outros. Ele oferece diversos recursos como plug-ins, prompts personalizados, contexto etc. <<OpenAI – ChatGPT>>.

O modelo base do ChatGPT é o GPT (Generative Pre-Trained Transformer), que é uma arquitetura de modelo de rede neural profunda capaz de gerar textos de alta qualidade, mas não se limitando a somente textos, já que é um modelo agnóstico quanto ao domínio, podendo ser aplicado tanto para sequências 1-D quanto para qualquer formato de sequências <<OpenAI – Image GPT>>. A sigla Generative Pre-Trained Transformer faz referência aos modelos generativos, especificamente os que são baseados na arquitetura Transformer desenvolvida pelo Google, que é uma arquitetura de Deep Learning, estruturada sobre o mecanismo de atenção que oferece acesso ao contexto em que os dados estão inseridos <<Attention is All you Need>>. Por Pre-trained entende-se a etapa inicial de treinamento do modelo, que é auto-supervisionada, ou seja, no caso de os dados serem textos são retiradas partes dos textos e a rede é treinada para prever essas partes. No caso dos modelos GPT é utilizada a técnica chamada autoregressive, que aprende a prever o que dá sequência a determinado segmento <<Wikipedia - Large language model>>.

O modelo pré-treinado (treinamento auto-supervisionado) dá origem ao que se costuma chamar de modelo de linguagem propriamente dito. A partir dele, para criar um modelo a ser utilizado em aplicações de conversação, como o ChatGPT, são feitos mais dois tipos de treinamentos: um supervisionado e outro aprendizado reforçado por feedback humano (RLHF). No treinamento supervisionado, geralmente são apresentadas perguntas e respostas e o modelo deve ir ajustando seus pesos para atribuir respostas às perguntas de acordo com os exemplos fornecidos. No treinamento reforçado por feedback humano é utilizada curadoria humana sobre as respostas apresentadas, a partir de um mecanismo de recompensas e punições, com o objetivo de melhorar a qualidade das respostas e de evitar respostas ofensivas ou que não se adequem a determinadas políticas definidas para a ferramenta <<Wikipedia – ChatGPT>>.

Essa combinação de técnicas transformam o ChatGPT em uma ferramenta bastante robusta e com uma alta capacidade de processamento e geração de textos, vídeos, áudios, imagens, entre outros tipos de dados.

Referências:

- OpenAI – ChatGPT: <https://openai.com/chatgpt>
- OpenAI – Image GPT: <https://openai.com/research/image-gpt>
- Wikipedia - ChatGPT: <https://pt.wikipedia.org/wiki/ChatGPT>
- Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia (2017). "Attention is All you Need".
- Wikipedia - Transformer (deep learning architecture): [https://en.wikipedia.org/wiki/Transformer_\(deep_learning_architecture\)](https://en.wikipedia.org/wiki/Transformer_(deep_learning_architecture))

- Wikipedia - Large language model: https://en.wikipedia.org/wiki/Large_language_model
- Wikipedia - Generative pre-trained transformer: https://en.wikipedia.org/wiki/Generative_pre-trained_transformer
- Wikipedia - Generative artificial intelligence: https://en.wikipedia.org/wiki/Generative_artificial_intelligence

1.d)

1. Agir como Humanos: O ChatGPT é capaz de imitar parcialmente o comportamento humano no contexto de um chat, conversando de forma relativamente natural e humana, tomando decisões sobre respostas com base no contexto da conversa e no “conhecimento” adquirido durante o treinamento. Mesmo assim, ele não imita exatamente um humano de forma proposital em várias situações. São alguns exemplos disso: análise de grande quantidade de informações em pouco tempo, geração de código também em tempo inviável para um ser humano e as respostas em que ele se identifica um modelo de IA.

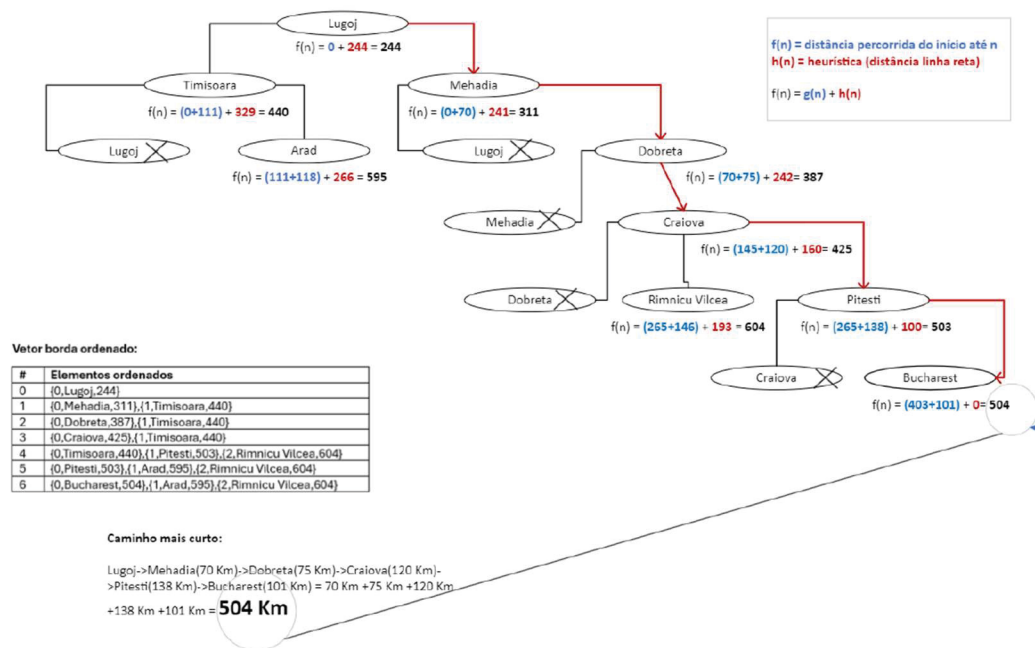
2. Pensar como Humanos: embora o ChatGPT imite, ainda que parcialmente, o comportamento humano, ele não replica exatamente os processos de pensamentos humanos. Por outro lado, ele utiliza redes neurais, que têm alguma inspiração nos processos biológicos de pensamento.

3. Pensar Racionalmente: embora o ChatGPT faça inferências sobre o mundo, elas ainda são limitadas e frequentemente ocorrem erros de lógica.

4. Agir Racionalmente: essa abordagem se encaixa melhor na descrição do funcionamento do ChatGPT. O modelo busca alcançar o melhor resultado possível, tomar decisões sob incerteza e agir de forma autônoma. Por ser uma abordagem mais ampla, acaba abrangendo também algumas características das outras abordagens.

Conclusão: o funcionamento do ChatGPT se encaixa principalmente na abordagem "Agir Racionalmente", mas apresenta características das outras abordagens.

2.a)



3.a)

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

Equivalência Implicação em R1:
R4: $\neg p \vee q$
Equivalência Implicação em R2:
R5: $\neg q \vee r$
Equivalência Implicação em R3:
R6: $r \rightarrow p$
Silogismo Hipotético em R1 e R2:
R7: $p \rightarrow r$
Silogismo Hipotético em R2 e R6:
R8: $q \rightarrow p$
Silogismo Hipotético em R6 e R1:
R9: $r \rightarrow q$
Conjunção em R8 e R1:
R10: $(q \rightarrow p) \wedge (p \rightarrow q)$

Bicondicional em R10:
R11: $(q \leftrightarrow p)$

Temos então a prova de que o argumento lógico é válido, pois R7 é, claramente, equivalente à conclusão **C**.

4.a)

Segundo (<https://www.deeplearningbook.com.br/funcacao-de-ativacao/>, em 23/02/2024), a função de ativação linear é representada pela seguinte fórmula:

$f(x) = ax$, consideraremos $a=1$, então:

$$f(x) = ax$$

$$N1 = f(x) = x, \text{ onde } x = (-3)*0,2 + 1*0,8 + 1*0,1$$

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

4.b)

$$N2 = f(x) = x = (-3)*0,1 + 1*0,2 + 1*0,4$$

$$\mathbf{N2 = 0,3}$$

4.c)

$$N3 = f(x) = x = (-3)*0,9 + 1*0,5 + 1*0,2$$

$$\mathbf{N3 = -2}$$

4.d)

Ainda segundo (<https://www.deeplearningbook.com.br/funcacao-de-ativacao/>, em 23/02/2024), a função de ativação TanH é definida da seguinte maneira: $\text{TanH}(x) = 2 / (1 + e^{(-2x)}) - 1$

Assim:

$$N4 = \tanh(x)$$

$$x = N1*0,9 + N2*0,3 + N3*0,3 + 1*0,1 = 0,3*0,9 + 0,3*0,3 + (-2)*0,3 + 1*0,1 = -0,14$$

$$N4 = \tanh(x) = 2 / (1 + e^{(-2x)}) - 1 = \tanh(-0,14) = -0,13909244787$$

$$\mathbf{N4 = -0,13909244787 \approx -0.14}$$

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 ANÁLISE EXPLORATÓRIA DOS DADOS

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

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

2 VISUALIZAÇÃO DOS DADOS

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

- Gere um gráfico da distribuição da quantidade de carros por marca
- Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro
- 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 R^2
- h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

B - RESOLUÇÃO

1) Análise Exploratória dos dados

1.a) Carregue a base de dados *media_precos_carros_brasil.csv*

```
# Lendo a base de dados
df = pd.read_csv('precos_carros_brasil.zip')
```

1.B) Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes

```
# verificando a quantidade de valores faltantes de cada variável
df.isnull().sum()
```

In [7]:

```
year_of_reference      65245
month_of_reference     65245
fipecode               65245
authentication         65245
brand                  65245
model                  65245
fuel                   65245
gear                   65245
engine_size            65245
year_model             65245
avg_price_brl          65245
dtype: int64
```

Out[7]:

```
# verificando a proporção de valores faltantes de cada variável
df.isnull().mean()
```

In [8]:

```
year_of_reference      0.243868
month_of_reference     0.243868
fipecode               0.243868
authentication         0.243868
brand                  0.243868
model                  0.243868
fuel                   0.243868
gear                   0.243868
engine_size            0.243868
year_model             0.243868
```

Out[8]:

```
avg_price_brl          0.243868
dtype: float64
```

In [9]:

```
# removendo os dados vazios, como todos estão na mesma proporção
provavelmente são linhas vazias
df = df.dropna()
```

In [10]:

```
# verificando novamente a proporção de valores nulos de cada variável e
comprovando que não restam mais dados faltantes
df.isnull().mean()
```

Out[10]:

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

• 1.C) Verifique se há dados duplicados nos dados

In [11]:

```
# comprovando que temos linhas repetidas, utilizando a propriedade shape.
Percebe-se que ao remover os duplicados a quantidade de linhas é reduzida
em 2.
print(f"df.shape: {df.shape}")
print(f"df.drop_duplicates().shape: {df.drop_duplicates().shape}")
df.shape: (202297, 11)
df.drop_duplicates().shape: (202295, 11)
```

In [12]:

```
# Também é possível utilizar o método duplicate() para verificar que
existem 2 registros duplicados:
df.duplicated().sum()
```

Out[12]:

2

In [13]:

```
# Verificando quais são as linhas que estão duplicadas
df[df.duplicated()]
```

Out[13]:

	year_of_refere nce	month_of_refere nce	fipe_co de	authentica tion	brand	model	fuel	gear	engine_si ze	year_mo del	avg_price_ brl
45793	2021.0	June	025232- 8	5rtdwkpqp 5h	Renau lt	DUSTE R OROC H Dyna.	Gasoli ne	manu al	2	2018.0	69893.0

	year_of_refere nce	month_of_refere nce	fipe_co de	authenticati on	brand	model	fuel	gear	engine_si ze	year_mo del	avg_price_ brl
						2.0 Flex 16V Mec.					
18989 6	2022.0	December	003296- 4	3r6c277cnqc b	Ford	Ranger Limited 3.0 PSE 4x4 CD TB Diesel	Diesel	manu al	3	2007.0	64638.0

In [14]:

```
# Removendo as linhas duplicadas e verificando
df = df.drop_duplicates()
df.shape
```

Out[14]:

```
(202295, 11)
```

In [15]:

```
# Verificando por outro método:
df.duplicated().sum() # sem dados duplicados
```

Out[15]:

```
0
```

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

In [16]:

```
# Verificando os tipos:
df.dtypes
```

Out[16]:

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

In [17]:

```
# Criando as duas categorias:
numérica = [col for col in df.columns if df[col].dtype != 'object']
categórica = [col for col in df.columns if df[col].dtype == 'object']
```

In [18]:

```
# Estatística descritiva das variáveis numéricas
df[numérica].describe()
```

Out[18]:

	year_of_reference	year_model	avg_price_brl
count	202295.000000	202295.000000	202295.000000
mean	2021.564695	2011.271514	52756.765713

	year_of_reference	year_model	avg_price_brl
std	0.571904	6.376241	51628.912116
min	2021.000000	2000.000000	6647.000000
25%	2021.000000	2006.000000	22855.000000
50%	2022.000000	2012.000000	38027.000000
75%	2022.000000	2016.000000	64064.000000
max	2023.000000	2023.000000	979358.000000

In [19]:

```
# Estatística descritiva das variáveis categóricas
df[categorica].describe()
```

Out[19]:

	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size
count	202295	202295	202295	202295	202295	202295	202295	202295
unique	12	2091	202295	6	2112	3	2	29
top	January	003281-6	cfzltzfwrcp	Fiat	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6
freq	24260	425	1	44962	425	168684	161883	47420

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

In [20]:

```
df.columns
```

Out[20]:

```
Index(['year_of_reference', 'month_of_reference', 'fipe_code',
      'authentication', 'brand', 'model', 'fuel', 'gear', 'engine_size',
      'year_model', 'avg_price_brl'],
      dtype='object')
```

In [21]:

```
# Contagem de valores por modelo
pd.DataFrame(df['model'].value_counts())
```

Out[21]:

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

2112 rows × 1 columns

In [22]:

```
# Contagem de valores por marca
```

```
pd.DataFrame(df['brand'].value_counts())
```

Out[22]:

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

In [23]:

```
# Contagem de valores por modelo e marca
pd.DataFrame(df[['model', 'brand']].value_counts())
```

Out[23]:

		count
model	brand	
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Fiat	425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	Ford	425
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	VW - VolksWagen	400
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	Ford	400
Doblo Adv/Adv TRYON/LOCKER 1.8 Flex	Fiat	375
...
Saveiro Robust 1.6 Total Flex 16V CD	VW - VolksWagen	2
Polo Track 1.0 Flex 12V 5p	VW - VolksWagen	2
STEPWAY Zen Flex 1.0 12V Mec.	Renault	2
Gol Last Edition 1.0 Flex 12V 5p	VW - VolksWagen	2
KICKS Active 1.6 16V Flex Aut.	Nissan	2

2112 rows × 1 columns

In [24]:

```
# Contagem de valores por modelo e marca no ano de 2021
pd.DataFrame(df[df['year_of_reference']==2021][['model', 'brand']].value_counts())
```

Out[24]:

		count
model	brand	
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Fiat	204
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	Ford	204
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	VW - VolksWagen	192
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	Ford	192
Golf 2.0/ 2.0 Mi Flex Aut/Tiptronic.	VW - VolksWagen	180
...
PULSE DRIVE 1.0 Turbo 200 Flex Aut.	Fiat	4
PULSE DRIVE 1.3 8V Flex Aut.	Fiat	4
SANDERO S Edition Flex 1.0 12V 5p Mec.	Renault	2
Clio Hi-Flex/ Expres. Hi-Flex 1.6 16V 3p	Renault	2
S10 Pick-Up Z71 2.8 TDI 4x4 CD Dies. Aut	GM - Chevrolet	2

2062 rows × 1 columns

In [25]:

```
# Contagem de valores por modelo e marca no ano de 2022
pd.DataFrame(df[df['year_of_reference']==2022][['model', 'brand']].value_counts())
```

Out[25]:

		count
model	brand	
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Fiat	204
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	Ford	204
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	Ford	192
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	VW - VolksWagen	192
Corvette 5.7/ 6.0, 6.2 Targa/Stingray	GM - Chevrolet	180
...
Polo Highline TSI 1.0 Flex 12V Aut.	VW - VolksWagen	6
Fastback Audace 1.0 200 T. Flex Aut	Fiat	6
Fastback Impetus 1.0 200 T. Flex Aut	Fiat	6
Fastback Limited Ed.1.3 270 T.Flex Aut.	Fiat	6
Polo Comfortline TSI 1.0 Flex 12V Aut.	VW - VolksWagen	6

2090 rows × 1 columns

In [26]:

```
# Contagem de valores por modelo e marca no ano de 2023
pd.DataFrame(df[df['year_of_reference']==2023][['model', 'brand']].value_counts())
```

Out[26]:

		count
model	brand	
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	Ford	17
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Fiat	17
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	VW - VolksWagen	16
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	Ford	16
Golf 2.0/ 2.0 Mi Flex Aut/Tiptronic.	VW - VolksWagen	15
...
Corsa Super 1.0 MPFI 16V 3p	GM - Chevrolet	1
COBALT ADVANTAGE 1.4 MPFI 8V F.Power 4p	GM - Chevrolet	1
KICKS SV Limited 1.6 16V Flex 5p Aut.	Nissan	1
COBALT ADVANTAGE 1.8 8V Eco.Flex 4p Aut.	GM - Chevrolet	1
up! track 1.0 Total Flex 12V 5p	VW - VolksWagen	1

2090 rows × 1 columns

In [27]:

```
# Contagem de valores por modelo e marca - Outro método.
df.groupby(['model', 'brand']).agg({'avg_price_brl': 'sum',
'year_of_reference': 'size'})\
.rename(columns={'avg_price_brl': 'Soma de Precos', 'year_of_reference':
'Contagem'}).sort_values(by='Contagem', ascending=False)
```

Out[27]:

		Soma de Precos	Contagem
model	brand		
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Fiat	16805703.0	425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	Ford	15718023.0	425
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	VW - VolksWagen	9732477.0	400
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	Ford	15726703.0	400
Doblo Adv/Adv TRYON/LOCKER 1.8 Flex	Fiat	18866956.0	375
...
Saveiro Robust 1.6 Total Flex 16V CD	VW - VolksWagen	210866.0	2
Polo Track 1.0 Flex 12V 5p	VW - VolksWagen	155303.0	2
STEPWAY Zen Flex 1.0 12V Mec.	Renault	152592.0	2
Gol Last Edition 1.0 Flex 12V 5p	VW - VolksWagen	183825.0	2
KICKS Active 1.6 16V Flex Aut.	Nissan	221512.0	2

2112 rows × 2 columns

- **1.F) Dê uma breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados**

Resposta:

- O dataset original possui 267.542 registros e 11 colunas. Foram removidos 65.245 registros que apresentavam valores faltantes em todas as colunas, resultando em 202.297 registros. Destes havia 2 registros duplicados que quando removidos resultaram em 202.295 registros.
- 'year_of_reference', 'year_model', 'avg_price_brl' são explicitamente numéricas. 'month_of_reference' também poderia ser transformada em numérica para facilitar o ordenamento em algumas análises. 'engine_size' também parece ser uma boa candidata a ser numérica, pois possui somente valores numéricos e que possuem ordem entre si. As demais colunas são categóricas.
- Os dados referem-se aos anos de referência entre 2021 e 2023, com anos dos modelos entre 2000 e 2023, sendo que metade deles estão entre 2000 e 2012 e outra metade até 2023. "Fiat Palio Week. Adv/Adv TRYON 1.8 mpi Flex" e "Ford Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p" são os modelos que contêm mais registros ao longo do período de referência presente no dataset.
- Com relação aos preços (em Reais), o valor médio é de 52.756,91 e um desvio padrão de 51.628,68, que distribui-se ao longo do valor mínimo de 6.647,00 e máximo de 979.358,00.

2) Visualização dos dados

2.A) Gere um gráfico da distribuição da quantidade de carros por marca

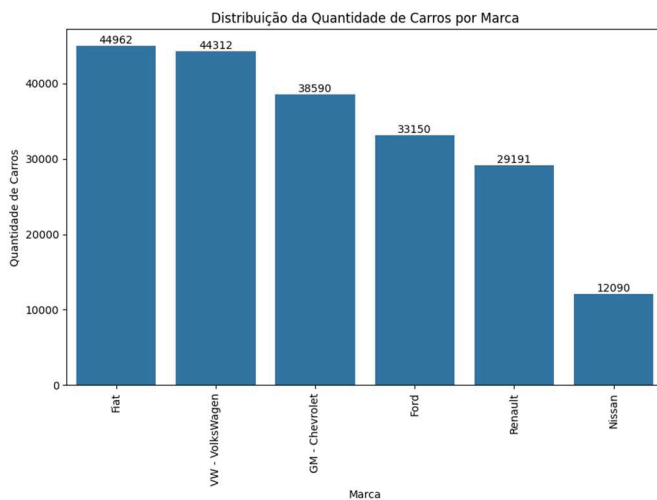
In [28]:

```
# Contar a quantidade de carros por marca
contagem_por_marca = df['brand'].value_counts()

# Criar um gráfico de barras
plt.figure(figsize=(10, 6))
sns.barplot(x=contagem_por_marca.index, y=contagem_por_marca.values)
```

```
# Adicionar os números nas barras
for i in range(len(contagem_por_marca)):
    plt.annotate(str(contagem_por_marca.values[i]), xy=(i,
contagem_por_marca.values[i]), ha='center', va='bottom')

plt.xticks(rotation=90) # Rotacionar os rótulos do eixo x para melhor
visualização
plt.xlabel('Marca')
plt.ylabel('Quantidade de Carros')
plt.title('Distribuição da Quantidade de Carros por Marca')
plt.show()
```



- **2.B) Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro**

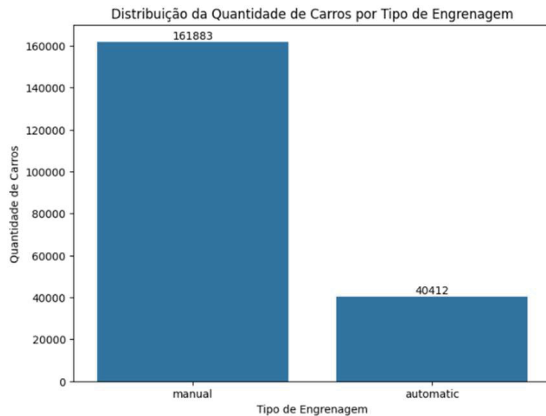
In [29]:

```
# Contar a quantidade de carros por tipo de engrenagem
contagem_por_engrenagem = df['gear'].value_counts()

# Criar um gráfico de barras
plt.figure(figsize=(8, 6))
sns.barplot(x=contagem_por_engrenagem.index,
y=contagem_por_engrenagem.values)

# Adicionar os números nas barras
for i in range(len(contagem_por_engrenagem)):
    plt.annotate(str(contagem_por_engrenagem.values[i]), xy=(i,
contagem_por_engrenagem.values[i]), ha='center', va='bottom')

plt.xlabel('Tipo de Engrenagem')
plt.ylabel('Quantidade de Carros')
plt.title('Distribuição da Quantidade de Carros por Tipo de Engrenagem')
plt.show()
```

- **2.C) Gere um gráfico da evolução da média de preço dos carros ao longo dos meses de 2022 (variável de tempo no eixo X)**

In [30]:
df.columns

Out[30]:
Index(['year_of_reference', 'month_of_reference', 'fiipe_code',
 'authentication', 'brand', 'model', 'fuel', 'gear', 'engine_size',
 'year_model', 'avg_price_brl'],
 dtype='object')

In [31]:
pegando a base de 2022
df_2022 = df[df['year_of_reference'] == 2022]

Definir a função que mapeia o nome do mês para o número do mês
def mes_para_numero(mes):

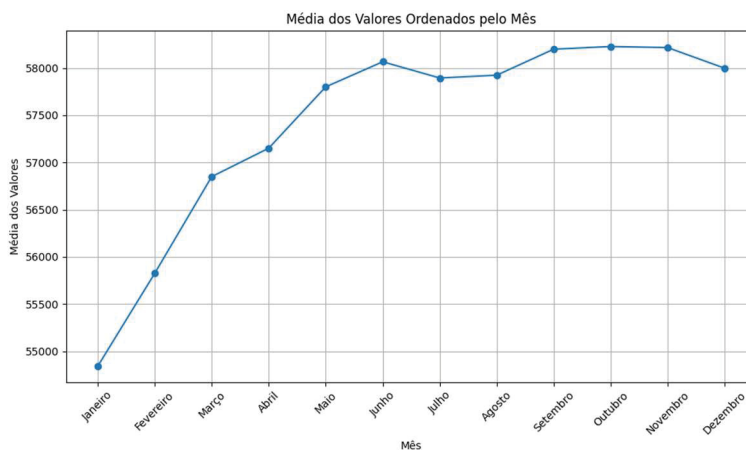
```
    meses = {
        'January': 1,
        'February': 2,
        'March': 3,
        'April': 4,
        'May': 5,
        'June': 6,
        'July': 7,
        'August': 8,
        'September': 9,
        'October': 10,
        'November': 11,
        'December': 12
    }
    return meses.get(mes)
```

Criar uma nova coluna com o número do mês
df_2022['numero_do_mes'] = df_2022['month_of_reference'].apply(mes_para_numero)

ordenando pela variavel mes
df_2022 = df_2022.sort_values(by='numero_do_mes')

```
# Calcular a média dos valores para cada mês
df_2022 = df_2022.groupby('numero_do_mes')['avg_price_brl'].mean()

# Plotar o gráfico da média dos valores ordenados pelo mês
plt.figure(figsize=(10, 6))
plt.plot(df_2022.index, df_2022.values, marker='o', linestyle='-')
plt.xlabel('Mês')
plt.ylabel('Média dos Valores')
plt.title('Média dos Valores Ordenados pelo Mês')
plt.xticks(range(1, 13), ['Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio',
                           'Junho', 'Julho', 'Agosto', 'Setembro', 'Outubro', 'Novembro', 'Dezembro'],
            rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



- **2.D) Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem**

```
df.columns
```

In [32]:

```
Index(['year_of_reference', 'month_of_reference', 'fiipe_code',
      'authentication', 'brand', 'model', 'fuel', 'gear', 'engine_size',
      'year_model', 'avg_price_brl'],
      dtype='object')
```

Out[32]:

```
# Calcular a média de preço dos carros por marca e tipo de engrenagem
media_preco = df.groupby(['brand',
                           'gear'])['avg_price_brl'].mean().unstack()
```

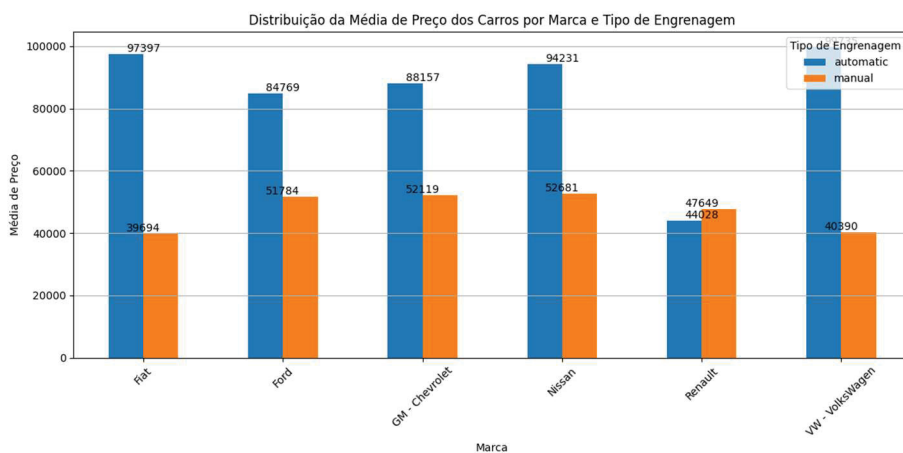
In [33]:

```
# Plotar o gráfico de barras agrupado
ax = media_preco.plot(kind='bar', figsize=(12, 6))
```

```
# Adicionar os valores das médias de preço sobre as barras
for i in range(len(media_preco)):
    for j in range(len(media_preco.columns)):
```

```
plt.annotate(str(round(media_preco.values[i][j], )),
             xy=(i, media_preco.values[i][j]),
             ha='center', va='bottom')

plt.xlabel('Marca')
plt.ylabel('Média de Preço')
plt.title('Distribuição da Média de Preço dos Carros por Marca e Tipo de Engrenagem')
plt.xticks(rotation=45)
plt.legend(title='Tipo de Engrenagem')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



• **2.E) Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item d**

- Geralmente carros automáticos são vendidos com um preço médio superior ao dos carros manuais, podendo chegar ao dobro do valor, que é o caso de algumas marcas como a Fiat e VW - Volkswagen.
- A marca Renault é a única em que o preço médio de carro automático é menor que manual.
- A marca Ford é que tem o menor preço médio para automáticos: R\$ 84.769,00.
- A marca VW-Volkswagem é que tem o maior preço médio para automáticos: R\$99.735,00.

• **2.F) Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de combustível**

In [34]:

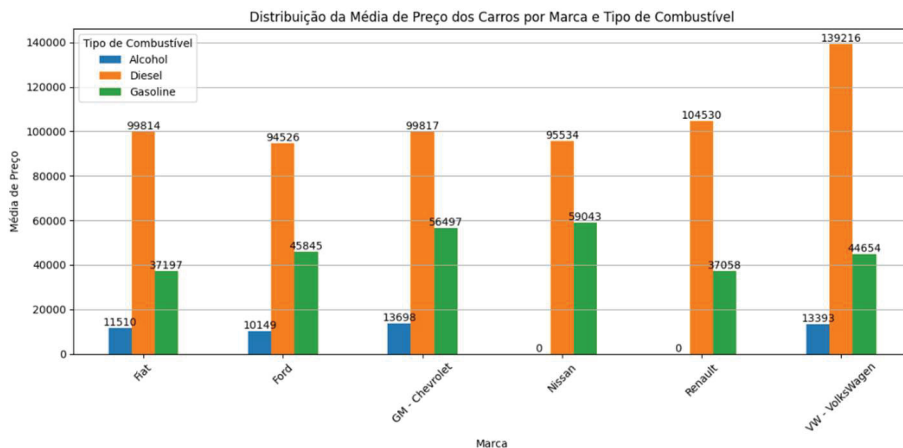
```
# Calcular a média de preço dos carros por marca e tipo de combustível
media_preco = df.groupby(['brand',
                          'fuel'])['avg_price_brl'].mean().unstack()

# Plotar o gráfico de barras agrupado
ax = media_preco.plot(kind='bar', figsize=(12, 6))
plt.xlabel('Marca')
plt.ylabel('Média de Preço')
plt.title('Distribuição da Média de Preço dos Carros por Marca e Tipo de Combustível')
plt.xticks(rotation=45)
```

```
plt.legend(title='Tipo de Combustível')
plt.grid(axis='y')

# Adicionar os valores de média de preço em cima de cada barra
for p in ax.patches:
    ax.annotate(str(round(p.get_height(), )), (p.get_x() + p.get_width() /
2., p.get_height()), ha='center', va='bottom')

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



• **2.G) Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item f**

- Pode-se perceber que em todas as marcas os carros se tornam mais caros quando são movidos à diesel, depois gasolina e por último álcool. Renault e Nissan não possuem carros à álcool no dataset analisado.
- Os carros à diesel têm um preço médio que corresponde a aproximadamente o dobro que os carros à gasolina. Já os carros à álcool têm um preço médio muito menor.
- Os preços médios dos carros à diesel é extremamente elevado em todas as marcas, sendo a mais cara a VW chegando a um preço médio de R\$139.216,00.
- Os preços dos carros movidos à álcool não varia tanto entre as marcas. Renault e Nissan não possuem carros à álcool no dataset analisado.

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

In [35]:
df.columns

Out[35]:
Index(['year_of_reference', 'month_of_reference', 'fiipe_code',
 'authentication', 'brand', 'model', 'fuel', 'gear', 'engine_size',
 'year_model', 'avg_price_brl'],
 dtype='object')

In [36]:
categorica

Out[36]:
['month_of_reference',
 'fiipe_code',
 'authentication',
 'brand',

```
'model',
'fuel',
'gear',
'engine_size']
```

- **3.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**

In [37]:

```
# Variáveis a serem removidas. fiipe_code e authentication não contém
informações relevantes para o modelo.
# fiipe_code já está contido nas informações de modelos e marcas e
authentication é um código de validação que não contém informações
relevantes no contexto dos modelos
remover = ['fiipe_code', 'authentication']

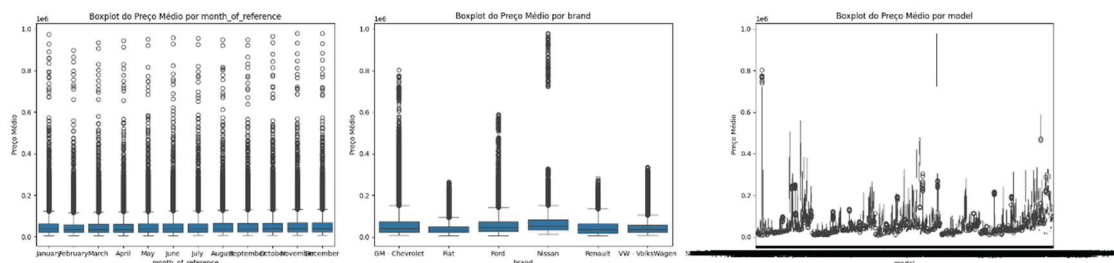
# Remover as variáveis da lista usando list comprehension
categorica = [var for var in categorica if var not in remover]
categorica
```

Out[37]:

```
['month_of_reference', 'brand', 'model', 'fuel', 'gear', 'engine_size']
```

In [38]:

```
# Analisando quais variáveis categóricas poderiam fazer sentido para o
modelo
for nomes in categorica:
    # Plotar o boxplot
    print(nomes, ' :')
    print('')
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=nomes, y='avg_price_brl', data=df)
    plt.xlabel(nomes)
    plt.ylabel('Preço Médio')
    plt.title(f'Boxplot do Preço Médio por {nomes}')
    plt.show()
month_of_reference :
```



In [39]:

df

Out[39]:

202295 rows × 11 columns

In []:

In [40]:

```
# Fazendo uma cópia do dataframe para aplicar nos modelos
df_ml = df.copy()
# removendo variáveis que parecem não fazer sentido para o modelo
#df.drop(['month_of_reference', 'model', 'fipe_code', 'authentication'], axis
= 1, inplace = True)
df_ml.drop(remover, axis = 1, inplace = True)
```

In [43]:

```
# Transformando variável engine_size em numérica
df_ml['engine_size'] = pd.to_numeric( df_ml['engine_size'].map(lambda x:
x.replace(',','.'))) )
```

In [44]:

```
# Vamos converter os meses de month_of_reference para valores numéricos
# Mapeia valores dos meses (january, february, etc..) para números de 0 a
11 em ordem.
# LabelEncoder não garante a ordem e ela pode ser importante para os
algoritmos
# Vamos usar outra técnica para garantir a ordem, que pode ser importante.
meses = df_ml['month_of_reference'].unique().tolist()
mapa_meses = {mes: str(i+1).zfill(2) for i, mes in enumerate(meses)}
df_ml['month_of_reference'] = df_ml['month_of_reference'].map(mapa_meses)
print(df_ml['month_of_reference'].unique().tolist())
['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12']
```

In [45]:

```
# Feature engineering:
# Agora vamos concatenar 'year_of_reference' e 'month_of_reference',
# criando uma nova coluna chamada 'year_month_reference',
# pois eles fazem mais sentido juntos
# e ao criar essa variável poupamos o modelo de ter que fazer mais essa
descoberta.
# A ordem year+month é importante por manter a ordem.
# Também é importante que os meses estejam formatados com 2 dígitos,
mantendo os zeros à esquerda.
```

```

df_ml['year_month_of_reference'] = df_ml['year_of_reference'].apply(lambda
x: str(x).replace('.0',''))+df_ml['month_of_reference']
df_ml['year_month_of_reference']=
df_ml['year_month_of_reference'].apply(int)

# Feature engineering: criando uma nova variável "tempo de mercado",
# que apresenta o a diferença entre o ano de referência da série temporal e
o ano do modelo do automóvel.
df_ml['tempo_mercado'] = df_ml.year_of_reference - df_ml.year_model

# Agora removemos a coluna year_of_reference e month_of_reference, pois
temos a nova coluna year_month_of_reference
df_ml.drop(['year_of_reference', 'month_of_reference'], axis=1,
inplace=True)

#Verificando os tipos
df_ml.dtypes

```

In [47]:

```

brand                object
model                object
fuel                 object
gear                 object
engine_size          float64
year_model           float64
avg_price_brl        float64
year_month_of_reference  int64
tempo_mercado        float64
dtype: object

```

Out[48]:

```

# atualizando as colunas categóricas
categorica = [col for col in df_ml.columns if df_ml[col].dtype == 'object']

# atualizando as colunas numéricas
numerica = [col for col in df_ml.columns if df_ml[col].dtype != 'object']

print("categorica", categorica)
print("numerica", numerica)
categorica ['brand', 'model', 'fuel', 'gear']
numerica ['engine_size', 'year_model', 'avg_price_brl',
'year_month_of_reference', 'tempo_mercado']

# Aplicando o método LabelEncoding nas outras variáveis categóricas.
# Talvez fosse mais interessante utilizar one-hot-encoding,
# pois o LabelEnconding traz uma relação de ordem que pode ser
inapropriada.
# Mas como não vimos na disciplina, vamos simplificar e usar o
LabelEncoding mesmo
for nomes in categorica:
    df_ml[nomes] = LabelEncoder().fit_transform(df_ml[nomes])
# Verificando as colunas

```

In [49]:

In [50]:

```
list(df_ml.columns)
```

Out[51]:

```
['brand',
 'model',
 'fuel',
 'gear',
 'engine_size',
 'year_model',
 'avg_price_brl',
 'year_month_of_reference',
 'tempo_mercado']
```

```
# Verificando os tipos se estão todos com tipos numéricos agora.
df_ml.dtypes
```

In [52]:

Out[52]:

```
brand          int64
model          int64
fuel           int64
gear           int64
engine_size    float64
year_model     float64
avg_price_brl  float64
year_month_of_reference  int64
tempo_mercado  float64
dtype: object
```

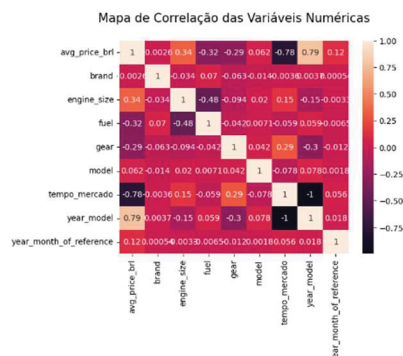
In [53]:

```
# atualizando novamente as colunas categóricas/numéricas
categorica = [col for col in df_ml.columns if df_ml[col].dtype == 'object']
numerica = [col for col in df_ml.columns if df_ml[col].dtype != 'object']
```

```
print("categorica", categorica)
print("numerica", numerica)
categorica []
numerica ['brand', 'model', 'fuel', 'gear', 'engine_size', 'year_model',
 'avg_price_brl', 'year_month_of_reference', 'tempo_mercado']
```

In [54]:

```
# Mapa de correlação entre as variáveis numéricas (incluindo target)
sns.heatmap(df_ml[sorted(numerica)].corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```



In []:

In []:

In [55]:

```
# Para analisar a correlação das features com a variável target,
# obtemos os índices de correlação e filtramos pela target,
# aplicamos abs (para analisar somente pela magnitude da correlação e não
considerar a direção (direta ou inversa))
# e, por fim, ordenados em ordem decrescente
df_corr =
abs(df_ml.corr(method='spearman')[['avg_price_brl']]).sort_values(by="avg_p
rice_brl", ascending=False)
df_corr
```

Out[55]:

	avg_price_brl
avg_price_brl	1.000000
year_model	0.787232
tempo_mercado	0.776231
engine_size	0.335524
fuel	0.315170
gear	0.292713
year_month_of_reference	0.121565
model	0.061857
brand	0.002561

In [56]:

```
# Temos que year_model é fortemente correlacionada (diretamente) e
tempo_mercado é fortemente correlacionada (inversamente).
# Logo depois, temos engine_size com uma correlação positiva moderada
# e fuel e gear com correlações negativas aproximadamente na mesma
magnitude de engine_size
# year_month_of_reference tem uma correlação fraca
# model e brand, especialmente brand possuem correlações bem fracas com a
target.
```

- **3.B) Crie partições contendo 75% dos dados para treino e 25% para teste**

In [57]:

```
# Variáveis explicativas/features
X = df_ml.drop(['avg_price_brl'],axis = 1)
X.head()
```

```
# Variável resposta/target
Y = df_ml['avg_price_brl']
```

```
# Divisão: 25% dos dados são de teste e 75% de treinamento
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
random_state = 24)
```

In [58]:

```
# Verificando Y
Y.head()
```

Out[59]:

```

0      9162.0
1      8832.0
2      8388.0
3      8453.0
4     12525.0
Name: avg_price_brl, dtype: float64

```

In [60]:

```

# Verificando os formatos e se a proporção entre treino e teste está
correta
print(df_ml.shape, X_train.shape, Y_test.shape,
round(Y_test.shape[0]/df.shape[0], 2))
(202295, 9) (151721, 8) (50574,) 0.25

```

- **3.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**

In [61]:

```

# Função auxiliar para treinar, predizer e calcular métricas relevantes
def treinar_modelo(desc, modelo, X_train, Y_train, X_test, Y_test):
    modelo_treinado = modelo.fit(X_train, Y_train)
    predicted = modelo_treinado.predict(X_test)
    return { 'desc' : desc,
            'modelo': modelo_treinado,
            'predicted': predicted,
            'feature_importances': pd.DataFrame(modelo.feature_importances_, index =
X_train.columns, columns=['importance']).sort_values('importance',
ascending = False),
            'mse': mean_squared_error(Y_test, predicted),
            'mae': mean_absolute_error(Y_test, predicted),
            'r2_score': r2_score(Y_test, predicted)
        }

```

In [62]:

```

# Treinamos 3 RFG e 2 XGBR com parâmetros diferentes.
# Os parâmetros de RFG foram escolhidos de acordo com o que foi descoberto
através de uma busca randômica com cross validation, realizada a parte
pelos integrantes do grupo
modelos = [RandomForestRegressor(),
            RandomForestRegressor(max_depth=10, min_samples_leaf=40,
min_samples_split=2000, n_estimators=350),
            RandomForestRegressor(n_estimators=200, min_samples_leaf= 5,
max_depth=25),
            XGBRegressor(),
            XGBRegressor(n_estimators=200, min_samples_leaf=35,
max_depth=5, learning_rate=0.3)
        ]
resultados = []
for modelo in modelos:

```

```
resultados.append(treinar_modelo("primeiro dataset",modelo, X_train,
Y_train, X_test, Y_test))
```

In []:

In [63]:

```
# Verificando resultados ordenados por R2 Score
pd.DataFrame(resultados).sort_values(by="r2_score",
ascending=False).style.format({'mse': '{:.5f}', 'mae': '{:.5f}', 'r2_score':
'{:.5f}'})
```

Out[63]:

	desc	modelo	predicted	feature_importances	mse	mae	r2_score
0	primeiro dataset	RandomForestRegressor()	[38883.16 39434.43 30754.61 ... 235097.26 68738.82 88976.11]	importance engine_size 0.449644 year_model 0.388835 model 0.056630 gear 0.033980 fuel 0.033758 year_month_of_reference 0.018617 brand 0.017223 tempo_mercado 0.001313	5094775.21852	1079.45244	0.99808
2	primeiro dataset	RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)	[38551.0045764 40189.57700563 29887.59927214 ... 237462.6957077 69244.56544732 87849.81251029]	importance engine_size 0.452532 year_model 0.390376 model 0.054063 fuel 0.034079 gear 0.033345 year_month_of_reference 0.017565 brand 0.016943 tempo_mercado 0.001097	11342023.08055	1729.44427	0.99572
4	primeiro dataset	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=0.3, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=5, max_leaves=None, min_child_weight=None, min_samples_leaf=35, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=200, n_jobs=None, num_parallel_tree=None, ...)	[40368.797 42872.89 28003.34 ... 243703.7 63282.902 88174.84]	importance engine_size 0.364650 fuel 0.241932 year_model 0.203306 gear 0.125106 brand 0.024754 model 0.022727 year_month_of_reference 0.015224 tempo_mercado 0.002301	26832565.58925	3081.67472	0.98986
3	primeiro dataset	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)	[41557.4 44044.99 27453.285 ... 235962.89 64311.69 82776.67]	importance engine_size 0.426685 year_model 0.220761 fuel 0.172464 gear 0.101514 brand 0.044566 model 0.018511 year_month_of_reference 0.013735 tempo_mercado 0.001765	27124030.48182	3110.92747	0.98975
1	primeiro dataset	RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=2000, n_estimators=350)	[30830.28139818 68886.75070984 25004.90578024 ... 215643.65480023 58719.24908819 78096.32034198]	importance engine_size 0.513285 year_model 0.419505 fuel 0.038545 gear 0.018823 model 0.005336 brand 0.002672 year_month_of_reference 0.001806 tempo_mercado 0.000027	455463309.23569	9562.45289	0.82796

In [64]:

```
# Agora vamos remover year_month_of_reference, model e brand
# que mostraram correlação absoluta fraca (<0.121565) na análise anterior.
# É esperado que as métricas fiquem menos favoráveis.
baixa_corr = ['year_month_of_reference', 'model', 'brand']
desc = "removendo features com baixa correlação"
for modelo in modelos:
    resultados.append(treinar_modelo(desc,modelo, X_train.drop(baixa_corr,
axis=1), Y_train, X_test.drop(baixa_corr, axis=1), Y_test))
```

In [65]:

```
#pd.DataFrame(resultados)[['desc','modelo','mse','mae','r2_score']].style.fo
rmat({'mse': '{:.5f}','mae': '{:.5f}','r2_score':
'{:.5f}'}).to_excel('resultados.xlsx', index=False)
```

In [66]:

```
# Mesmo com baixa correlação, as features removidas são importantes,
# pois percebemos um score R2 significativamente inferior ao do obtido com
todas as features em praticamente todos os modelos testados.
pd.DataFrame(resultados).sort_values(by="r2_score",
ascending=False).style.format({'mse': '{:.5f}','mae': '{:.5f}','r2_score':
'{:.5f}'})
```

Out[66]:

	desc	modelo	predicted	feature importances	mse	mae	r2_score
0	primeiro dataset	RandomForestRegressor()	[38883.16 39434.43 30754.61 ... 235097.26 68738.82 88976.11]	importance engine_size 0.449644 year_model 0.388835 model 0.056630 gear 0.033980 fuel 0.033758 year_month_of_reference 0.018617 brand 0.017223 tempo_mercado 0.001313	5094775.21852	1079.45244	0.99808
2	primeiro dataset	RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)	[38551.0045764 40189.57700563 29887.59927214 ... 237462.6957077 69244.56544732 87849.81251029]	importance engine_size 0.452532 year_model 0.390376 model 0.054063 fuel 0.034079 gear 0.033345 year_month_of_reference 0.017565 brand 0.016943 tempo_mercado 0.001097	11342023.08055	1729.44427	0.99572
4	primeiro dataset	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=0.3, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=5, max_leaves=None, min_child_weight=None, min_samples_leaf=35, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=200, n_jobs=None, num_parallel_tree=None, ...)	[40368.797 42872.89 28003.34 ... 243703.7 63282.902 88174.84]	importance engine_size 0.364650 fuel 0.241932 year_model 0.203306 gear 0.125106 brand 0.024754 model 0.022727 year_month_of_reference 0.015224 tempo_mercado 0.002301	26832565.58925	3081.67472	0.98986
3	primeiro dataset	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)	[41557.4 44044.99 27453.285 ... 235962.89 64311.69 82776.67]	importance engine_size 0.426685 year_model 0.220761 fuel 0.172464 gear 0.101514 brand 0.044566 model 0.018511 year_month_of_reference 0.013735 tempo_mercado 0.001765	27124030.48182	3110.92747	0.98975
7	removendo features com baixa correlação	RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)	[39094.19787054 41702.15346034 25099.63847269 ... 260501.39291887 51202.07765153 72344.63836105]	importance engine_size 0.491515 year_model 0.418055 gear 0.042979 fuel 0.036750 tempo_mercado 0.010701	169850208.50182	7115.05771	0.93584
9	removendo features com baixa correlação	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None,	[40354.387 41899.543 25089.46 ... 260741.69 50944.707 71880.31]	importance year_model 0.396500 engine_size 0.308275 fuel 0.157290 gear 0.130243 tempo_mercado 0.007691	170005186.96815	7133.72144	0.93578

	desc	modelo	predicted	feature_importances	mse	mae	r2_score
		feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=0.3, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=5, max_leaves=None, min_child_weight=None, min_samples_leaf=35, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=200, n_jobs=None, num_parallel_tree=None, ...) XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)					
8	removendo features com baixa correlação		[39486.047 41719.08 25021.5 ... 260255.05 50864.312 72336.08]	importance year_model 0.419754 engine_size 0.312843 gear 0.136833 fuel 0.122437 tempo_mercado 0.008134	170136981.15636	7134.30625	0.93573
5	removendo features com baixa correlação	RandomForestRegressor()	[39046.40454431 41580.57455413 25100.85181219 ... 261131.46865592 51316.07985214 72546.39687138]	importance engine_size 0.493420 year_model 0.417019 gear 0.041314 fuel 0.036997 tempo_mercado 0.011250	170138595.30602	7096.97366	0.93573
1	primeiro dataset	RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=2000, n_estimators=350)	[30830.28139818 68886.75070984 25004.90578024 ... 215643.65480023 58719.24908819 78096.32034198]	importance engine_size 0.513285 year_model 0.419505 fuel 0.038545 gear 0.018823 model 0.005336 brand 0.002672 year_month_of_reference 0.001806 tempo_mercado 0.000027	455463309.23569	9562.45289	0.82796
6	removendo features com baixa correlação	RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=2000, n_estimators=350)	[33447.97884641 71884.89692838 25019.42012059 ... 215444.08398827 60459.15723852 78077.60654301]	importance engine_size 0.517260 year_model 0.423539 fuel 0.040040 gear 0.019011 tempo_mercado 0.000150	471668582.35265	10012.92058	0.82183

3.D) Grave os valores preditos em variáveis criadas

In [67]:

```
# Para cada modelo testado (e cada conjunto de datasets) foi criado um
dicionário contendo descrição, modelo, predicted (valores preditos),
feature importances e métricas;
# Exemplo:
print(resultados[0])
print("predicted:", resultados[0]['predicted'])
{'desc': 'primeiro dataset', 'modelo': RandomForestRegressor(),
'predicted': array([ 38883.16, 39434.43, 30754.61, ..., 235097.26,
68738.82,
88976.11]), 'feature_importances':
importance
engine_size          0.449644
year_model           0.388835
model                0.056630
gear                 0.033980
fuel                 0.033758
```

```

year_month_of_reference    0.018617
brand                      0.017223
tempo_mercado              0.001313, 'mse': 5094775.21852061, 'mae':
1079.4524404756412, 'r2_score': 0.9980755293605337}
predicted: [ 38883.16  39434.43  30754.61 ... 235097.26  68738.82
88976.11]

```

3.E) Realize a análise de importância das variáveis para estimar a variável target, para cada modelo treinado

In [68]:

```

# Iterando sobre um dataframe com os resultados e acessando
feature_importances_ já armazenado anteriormente.
df_resultados = pd.DataFrame(resultados).sort_values(by="r2_score",
ascending=False)
for i, r in df_resultados.iterrows():
    desc = r['desc']
    model = r['modelo']
    feat_importances = r['feature_importances']
    print(f"{desc} - {model}")
    print(f"{feat_importances}")
    print("_"*120)

```

primeiro dataset - RandomForestRegressor()

	importance
engine_size	0.449644
year_model	0.388835
model	0.056630
gear	0.033980
fuel	0.033758
year_month_of_reference	0.018617
brand	0.017223
tempo_mercado	0.001313

primeiro dataset - RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)

	importance
engine_size	0.452532
year_model	0.390376
model	0.054063
fuel	0.034079
gear	0.033345
year_month_of_reference	0.017565
brand	0.016943
tempo_mercado	0.001097

primeiro dataset - XGBRegressor(...)

	importance
engine_size	0.364650
fuel	0.241932
year_model	0.203306
gear	0.125106
brand	0.024754
model	0.022727
year_month_of_reference	0.015224
tempo_mercado	0.002301

primeiro dataset - XGBRegressor(...)

	importance
engine_size	0.426685
year_model	0.220761

fuel	0.172464
gear	0.101514
brand	0.044566
model	0.018511
year_month_of_reference	0.013735
tempo_mercado	0.001765

removendo features com baixa correlação - RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)

	importance
engine_size	0.491515
year_model	0.418055
gear	0.042979
fuel	0.036750
tempo_mercado	0.010701

removendo features com baixa correlação - XGBRegressor(...)

	importance
year_model	0.396500
engine_size	0.308275
fuel	0.157290
gear	0.130243
tempo_mercado	0.007691

removendo features com baixa correlação - XGBRegressor (...)

	importance
year_model	0.419754
engine_size	0.312843
gear	0.136833
fuel	0.122437
tempo_mercado	0.008134

removendo features com baixa correlação - RandomForestRegressor()

	importance
engine_size	0.493420
year_model	0.417019
gear	0.041314
fuel	0.036997
tempo_mercado	0.011250

primeiro dataset - RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=2000, n_estimators=350)

	importance
engine_size	0.513285
year_model	0.419505
fuel	0.038545
gear	0.018823
model	0.005336
brand	0.002672
year_month_of_reference	0.001806
tempo_mercado	0.000027

removendo features com baixa correlação - RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=2000,

	importance
engine_size	0.517260
year_model	0.423539
fuel	0.040040
gear	0.019011
tempo_mercado	0.000150

Vamos restringir a análise aos melhores modelos de cada algoritmo (índices 0 e 4)

In [69]:

```
print("RandomForestRegressor()", df_resultados.loc[0].feature_importances)
print("_"*120)
print("XGBRegressor(base_score=None, booster=None, callbacks=None,...",
df_resultados.loc[4].feature_importances)
RandomForestRegressor()           importance
engine_size           0.449644
year_model            0.388835
model                 0.056630
gear                  0.033980
fuel                  0.033758
year_month_of_reference 0.018617
brand                 0.017223
tempo_mercado         0.001313
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,...           importance
engine_size           0.364650
fuel                  0.241932
year_model            0.203306
gear                  0.125106
brand                 0.024754
model                 0.022727
year_month_of_reference 0.015224
tempo_mercado         0.002301
```

3.F) Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis

- Analisando os melhores modelos (pelo critério da métrica R2) dos dois algoritmos (Random Forest Regressor e XGBRegressor), percebe-se que ambos os modelos foram baseados no dataset com o conjunto de features mais completo. O que leva a crer que as features apontadas como tendo menor índice de correlação com a variável target ainda são importantes para o bom desempenho dos modelos. Poderíamos analisar multicolinearidade entre features ou ainda algum tipo de overfitting, mas entendemos que esse tipo de análise está fora do escopo desta disciplina.
- engine_size foi a variável explicativa mais importante em ambos os modelos, o que "contraria" um pouco a análise de correlação com a variável target que tinha entre as maiores correlações as variáveis de tempos (year_model e tempo_mercado) e que provavelmente possuem alta correlação entre si, já que uma deriva de outra.
- year_model ainda é importante nos dois modelos, embora tenha ficado em 2º lugar no RandomForest e em 3ª no XGBoost. Mas nos dois modelos ela tem uma importância muito relevante.
- tempo_mercado acabou tendo uma importância menor, mas provavelmente porque year_model (que compõe tempo_mercado) dominou. Ao remover year_model devemos perceber um aumento da importância de tempo_mercado, embora deva levar a um R2 menor porque aparentemente year_model ainda é mais significativa.

3.G) Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R²

Com base nas métricas utilizadas, o melhor modelo para aplicar nos dados foi o Random Forest Regressor com parâmetros default [RandomForestRegressor()], pois obteve menores valores de MSE e MAE e o maior valor de R2.

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

Com base nas métricas utilizadas, o melhor modelo para aplicar nos dados foi o Random Forest Regressor com parâmetros default [RandomForestRegressor()], pois obteve menores valores de MSE e MAE e o maior valor de R2.

Logo depois temos outro Random Forest, agora com parâmetros configurados de acordo com uma busca randômica por parâmetros feito à parte. [RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)]

Em terceiro lugar temos o XGBoostRegressor, com parâmetros configurados e encontrados na busca randômica por parâmetros feito à parte. [XGBoostRegressor(n_estimators=200, min_samples_leaf=35, max_depth=5, learning_rate=0.3)]

Uma diferença muito pequena entre o R2 score dos 3, que mostraram, respectivamente, os valores: 0.99804, 0.99572 e 0.98986.

As demais métricas também acompanham a métrica R2, pois tendem a diminuir enquanto a R2 aumenta.

In [70]:

```
pd.DataFrame(resultados).sort_values(by="r2_score",
ascending=False).style.format({'mse': '{:.5f}', 'mae': '{:.5f}', 'r2_score':
'{:.5f}' })
```

Out[70]:

	desc	modelo	predicted	feature_importances	mse	mae	r2_score
0	primeiro dataset	RandomForestRegressor()	[38883.16 39434.43 30754.61 ... 235097.26 68738.82 88976.11]	importance engine_size 0.449644 year_model 0.388835 model 0.056630 gear 0.033980 fuel 0.033758 year_month_of_reference 0.018617 brand 0.017223 tempo_mercado 0.001313	5094775.21852	1079.45244	0.99808
2	primeiro dataset	RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)	[38551.0045764 40189.57700563 29887.59927214 ... 237462.6957077 69244.56544732 87849.81251029]	importance engine_size 0.452532 year_model 0.390376 model 0.054063 fuel 0.034079 gear 0.033345 year_month_of_reference 0.017565 brand 0.016943 tempo_mercado 0.001097	11342023.08055	1729.44427	0.99572
4	primeiro dataset	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=0.3, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=5, max_leaves=None, min_child_weight=None, min_samples_leaf=35, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=200, n_jobs=None, num_parallel_tree=None, ...) XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)	[40368.797 42872.89 28003.34 ... 243703.7 63282.902 88174.84]	importance engine_size 0.364650 fuel 0.241932 year_model 0.203306 gear 0.125106 brand 0.024754 model 0.022727 year_month_of_reference 0.015224 tempo_mercado 0.002301	26832565.58925	3081.67472	0.98986
3	primeiro dataset	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)	[41557.4 44044.99 27453.285 ... 235962.89 64311.69 82776.67]	importance engine_size 0.426685 year_model 0.220761 fuel 0.172464 gear 0.101514 brand 0.044566 model 0.018511 year_month_of_reference 0.013735 tempo_mercado 0.001765	27124030.48182	3110.92747	0.98975
7	removendo features com baixa correlação	RandomForestRegressor(max_depth=25, min_samples_leaf=5, n_estimators=200)	[39094.19787054 41702.15346034 25099.63847269 ... 260501.39291887	importance engine_size 0.491515 year_model 0.418055 gear 0.042979	169850208.50182	7115.05771	0.93584

desc	modelo	predicted	feature_importances	mse	mae	r2_score	
		51202.07765153 72344.63836105]	fuel 0.036750 tempo_mercado 0.010701				
9	removendo features com baixa correlação	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=0.3, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=5, max_leaves=None, min_child_weight=None, min_samples_leaf=35, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=200, n_jobs=None, num_parallel_tree=None, ...) XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)	[40354.387 41899.543 25089.46 ... 260741.69 50944.707 71880.31]	importance year_model 0.396500 engine_size 0.308275 fuel 0.157290 gear 0.130243 tempo_mercado 0.007691	170005186.96815	7133.72144	0.93578
8	removendo features com baixa correlação	XGBRegressor(base_score=None, booster=None, callbacks=None, colsample_bylevel=None, colsample_bynode=None, colsample_bytree=None, device=None, early_stopping_rounds=None, enable_categorical=False, eval_metric=None, feature_types=None, gamma=None, grow_policy=None, importance_type=None, interaction_constraints=None, learning_rate=None, max_bin=None, max_cat_threshold=None, max_cat_to_onehot=None, max_delta_step=None, max_depth=None, max_leaves=None, min_child_weight=None, missing=nan, monotone_constraints=None, multi_strategy=None, n_estimators=None, n_jobs=None, num_parallel_tree=None, random_state=None, ...)	[39486.047 41719.08 25021.5 ... 260255.05 50864.312 72336.08]	importance year_model 0.419754 engine_size 0.312843 gear 0.136833 fuel 0.122437 tempo_mercado 0.008134	170136981.15636	7134.30625	0.93573
5	removendo features com baixa correlação	RandomForestRegressor()	[39046.40454431 41580.57455413 25100.85181219 ... 261131.46865592 51316.07985214 72546.39687138]	importance engine_size 0.493420 year_model 0.417019 gear 0.041314 fuel 0.036997 tempo_mercado 0.011250	170138595.30602	7096.97366	0.93573
1	primeiro dataset	RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=2000, n_estimators=350)	[30830.28139818 68886.75070984 25004.90578024 ... 215643.65480023 58719.24908819 78096.32034198]	importance engine_size 0.513285 year_model 0.419505 fuel 0.038545 gear 0.018823 model 0.005336 brand 0.002672 year_month_of_reference 0.001806 tempo_mercado 0.000027	455463309.23569	9562.45289	0.82796
6	removendo features com baixa correlação	RandomForestRegressor(max_depth=10, min_samples_leaf=40, min_samples_split=2000, n_estimators=350)	[33447.97884641 71884.89692838 25019.42012059 ... 215444.08398827 60459.15723852 78077.60654301]	importance engine_size 0.517260 year_model 0.423539 fuel 0.040040 gear 0.019011 tempo_mercado 0.000150	471668582.35265	10012.92058	0.82183

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;

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1 Pesquisa com Dados de Satélite (Satellite)

```
library("caret")
library(mlbench)

# Carregando dataset
data(Satellite)

# Verificando o dataset
head(Satellite)

# df_sattelite vai ser o dataset principal
df_satellite <- Satellite

# Separando os dados em treinamento (80%) e teste (20%)
set.seed(24) # A equipe 24 elegeu o número 24 para podermos comparar os resultados de cada integrante de um
a forma mais determinística

indices <- createDataPartition(df_satellite$classes, p=0.80, list=FALSE)

df_treino <- df_satellite[indices,]
df_teste <- df_satellite[-indices,]

# Função auxiliar para treinar, predizer e fazer matriz de confusão
treinar <- function(df_treino, df_teste, algo) {
  set.seed(24) # A equipe 24 elegeu o número 24 para podermos comparar os resultados de cada integrante de
  uma forma mais determinística

  model = caret::train(classes~., data=df_treino, method=algo, trace=FALSE)

  predictions <- predict(model, df_teste)

  confusion_matrix <- caret::confusionMatrix(predictions, df_teste$classes)

  return(list(model = model, predictions = predictions, confusion_matrix = confusion_matrix) )
}

# Treinando, predizendo e fazendo matriz de confusão para rf, svm e nnet com o dataset original (todas as f
eatures):

rf <- treinar(df_treino, df_teste, "rf")

svm <- treinar(df_treino, df_teste, "svmRadial")
```

```

rna <- treinar(df_treino, df_teste, "nnet")

# Criando dataframe com os resultados (acurácia para cada modelo)

df_resultados <- data.frame(desc="todas features", rf=rf$confusion_matrix$overall["Accuracy"], svm=svm$confusion_matrix$overall["Accuracy"], rna=rna$confusion_matrix$overall["Accuracy"])

# Selecionando apenas os 4 atributos importantes, segundo enunciado do exercício

df_treino <- df_treino[, c("x.17", "x.18", "x.19", "x.20", "classes")]

df_teste <- df_teste[, c("x.17", "x.18", "x.19", "x.20", "classes")]

# Treinando, predizendo e fazendo matriz de confusão para rf, svm e nnet com o dataset com apenas 4 features:

rf <- treinar(df_treino, df_teste, "rf")

svm <- treinar(df_treino, df_teste, "svmRadial")

rna <- treinar(df_treino, df_teste, "nnet")

# Adicionando nova linha no dataframe com os novos resultados

df_resultados <- rbind(df_resultados, data.frame(desc="4 features", rf=rf$confusion_matrix$overall["Accuracy"], svm=svm$confusion_matrix$overall["Accuracy"], rna=rna$confusion_matrix$overall["Accuracy"]))

df_resultados

```

Considerando todas as features, o melhor modelo (pelo critério da acurácia) foi Random Forest com 0.90 de acurácia, seguido pelo SVM e Rede Neural. Se considerarmos somente as features x.17, x.18, x.19 e x.20 conforme o enunciado, obtemos a melhor acurácia no SVM com 0.865, seguido pelo Random Forest com 0.856 (praticamente um empate técnico). Interessante notar que a Rede Neural aumentou bastante a sua acurácia nesse dataset que utiliza apenas 4 features, chegando a 0.820 (enquanto que com o dataset completo obteve 0.563).

2 Estimativa de Volumes de Árvores

```

# Carregando bibliotecas

library(caret)

# Carregando o arquivo csv em um dataframe

df_volumes <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv")

# Verificando uma pequena amostra dos dados

head(df_volumes)

# Selecionando apenas os atributos importantes (descartando o NR)

df_volumes <- df_volumes[, -1]

# Verificando novamente uma amostra dos dados após seleção

head(df_volumes)

# Separando datasets de treinamento e testes

set.seed(24)

indices <- createDataPartition(df_volumes$VOL, p=0.80, list=FALSE)

df_treino <- df_volumes[indices,]

df_teste <- df_volumes[-indices,]

#### Funções auxiliares para calcular R2, Syx e Syxp

# Métrica R2

r2 <- function(y, yhat) {

  return ( 1 - (sum((y-yhat)^2) / sum((y - mean(y))^2)) )

}

# Métrica Syx

```

```

Syx <- function(y, yhat) {
  n = length(y)
  return (sqrt( sum((y-yhat)^2) / (n-2) ))
}

#Métrica Syxp
Syxp <- function(y, yhat) {
  return ( Syx(y,yhat)/mean(y) *100)
}

# Treinando cada um dos modelos
set.seed(24)

# Random Forest
rf <- caret::train(VOL~., data=df_treino, method="rf")
set.seed(24)

# SVM
svm <- caret::train(VOL~., data=df_treino, method="svmRadial")
set.seed(24)

# Rede Neural
rna <- caret::train(VOL~., data=df_treino, method="nnet", trace=FALSE)

# Modelo alométrico de SPURR
set.seed(24)
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, df_treino, start=list(b0=0.5, b1=0.5))

### Predições

# Random forest
pred_rf <- predict(rf, df_teste)

# SVM
pred_svm <- predict(svm, df_teste)

# RNA
pred_rna <- predict(rna, df_teste)

# modelo alométrico de SPURR
pred_alom <- predict(alom, df_teste)

## Métricas

#R2
r2_rf = r2(df_teste$VOL, pred_rf)
r2_svm = r2(df_teste$VOL, pred_svm)
r2_rna = r2(df_teste$VOL, pred_rna)
r2_alom = r2(df_teste$VOL, pred_alom)

#Syx
Syx_rf = Syx(df_teste$VOL, pred_rf)
Syx_svm = Syx(df_teste$VOL, pred_svm)
Syx_rna = Syx(df_teste$VOL, pred_rna)
Syx_alom = Syx(df_teste$VOL, pred_alom)

#Syx%
Syxp_rf = Syxp(df_teste$VOL, pred_rf)
Syxp_svm = Syxp(df_teste$VOL, pred_svm)
Syxp_rna = Syxp(df_teste$VOL, pred_rna)

```

```
Sysxp_alom = Syxp(df_teste$VOL, pred_alom)

# Criando um dataframe com os resultados para facilitar a comparação

df_resultados <- data.frame(

  R2 = c(r2_rf, r2_svm, r2_rna, r2_alom),

  Sysx = c(Sysx_rf, Sysx_svm, Sysx_rna, Sysx_alom),

  Sysxp = c(Sysxp_rf, Sysxp_svm, Sysxp_rna, Sysxp_alom)

)

# Define os nomes das linhas.

rownames(df_resultados) <- c("RF", "SVM", "RNA", "Alom")

# Verificando os resultados

df_resultados
```

O maior R2 foi obtido no modelo alométrico de Spurr, assim como os menores erros padrões de estimativa. Esse, portanto é o modelo escolhido. Logo depois vem o Random Forest, seguindo por SVM e RNA.

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 mostrá-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

Bibliotecas:

```
library(dplyr)
library(onewaytests)
library(stats)
library(carData)
library(datasets)
library(BSDA)
library(nortest)
library(rcompanion)
library(tigerstats)
```

```
library(misty)
library(ggpubr)
library(fdth)
```

Lendo a base de dados

```
load("./salarios.RData")
head(salarios,10)
```

##	husage	husunion	husearns	huseduc	husblk	hushisp	hushrs	kidge6	earns	age
## 1	42	0	568	14	0	0	40	1	290	43
## 2	26	0	600	14	0	0	0	0	654	26
## 3	56	0	1500	14	0	0	40	1	100	49
## 4	35	NA	0	12	0	0	40	0	0	35
## 5	42	0	450	11	0	0	45	1	0	43
## 6	55	0	465	13	0	0	50	0	0	58
## 7	68	NA	0	16	0	0	0	0	360	56
## 8	48	0	492	10	0	1	48	0	0	48
## 9	38	NA	0	14	0	0	65	1	0	37
## 10	48	1	1923	12	0	0	21	0	0	39

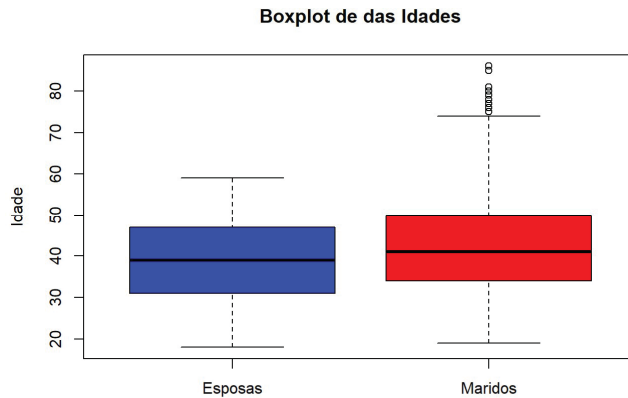
##	black	educ	hispanic	union	faminc	husexp	exper	kidlt6	hours	nwifeinc	inlf
## 1	0	14	0	0	45000	22	23	0	0	29.920	0
## 2	0	14	0	0	45000	6	6	0	0	10.992	0
## 3	0	12	0	0	55000	36	31	0	15	49.800	1
## 4	0	10	0	NA	55000	17	19	1	0	55.000	0
## 5	0	13	0	NA	27500	25	24	0	0	27.500	0
## 6	0	12	0	NA	55000	36	40	0	0	55.000	0
## 7	0	12	0	0	22500	46	38	0	45	3.780	1
## 8	0	12	1	NA	45000	32	30	0	0	45.000	0
## 9	0	12	0	NA	37500	18	19	0	0	37.500	0
## 10	0	14	0	NA	112500	30	19	1	0	112.500	0

##	hrwage	lwage
## 1	NA	NA
## 2	NA	NA
## 3	6.666667	1.897120
## 4	NA	NA
## 5	NA	NA
## 6	NA	NA
## 7	8.000000	2.079442
## 8	NA	NA
## 9	NA	NA
## 10	NA	NA

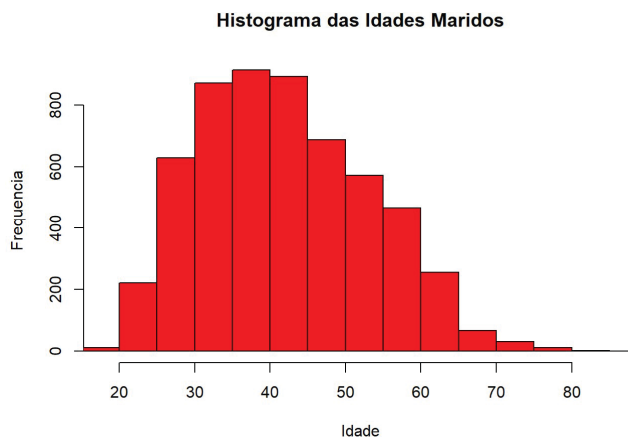
1 Gráficos e Tabelas

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

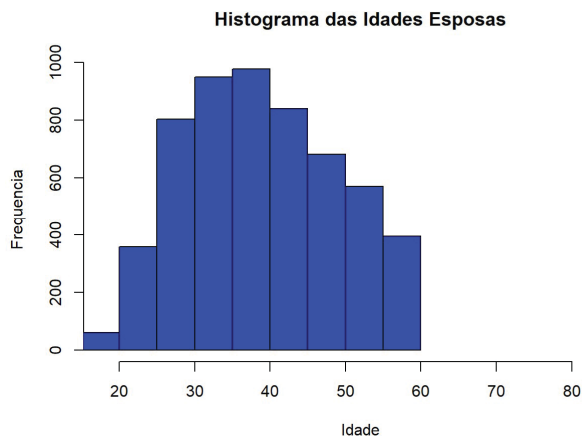
```
# Criando os boxplots para as duas variáveis e plotando-os no mesmo local
boxplot(salarios$age, salarios$husage, col = c("blue", "red"),
        names = c("Esposas", "Maridos"),
        main = "Boxplot de das Idades",
        xlab = "", ylab = "Idade")
```



```
# Criando os histogramas para as duas variáveis e plotando-os no mesmo local.
# Definindo máximos e mínimos para configurar gráfico.
max_age = max(salarios$age, salarios$husage)
min_age = min(salarios$age, salarios$husage)
# Histograma das Idades das Esposas
hist(salarios$husage, col = "red", main = "Histograma das Idades Maridos",
     xlab = "Idade", ylab = "Frequencia", xlim = c(min_age, max_age))
```



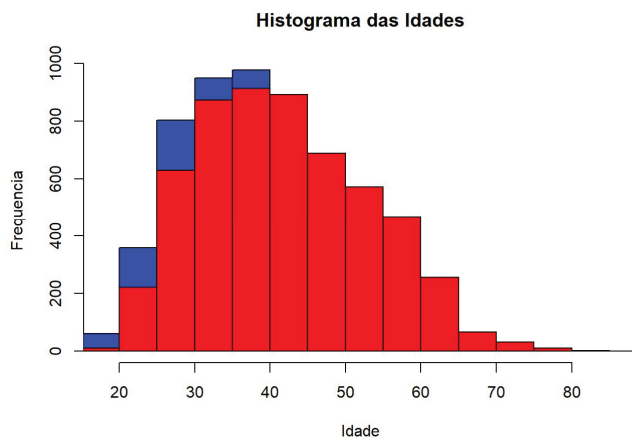
```
# Histograma das Idades dos Maridos
hist(salarios$age, col = "blue", main = "Histograma das Idades Esposas",
     xlab = "Idade", ylab = "Frequencia", xlim = c(min_age, max_age) )
```



```
# Agora os dois histogramas sobrepostos

# Histograma das Idades dos Esposas
hist(salarios$age, col = "blue", main = "Histograma das Idades",
      xlab = "Idade", ylab = "Frequencia", xlim = c(min_age, max_age) )

# Histograma das Idades das Maridos
hist(salarios$husage, col = "red", add=TRUE)
```



Resposta/Análise:

Ao analisar os box plots e os histogramas podemos perceber que:

- A idade mínima entre maridos e esposas é muito semelhante.
- Já a idade máxima é diferente. Enquanto as esposas possuem idade máxima de 60 anos os maridos têm uma idade máxima um pouco maior, em torno de 70 anos, com alguns outliers acima de 70 anos.
- A mediana da idade é semelhante, embora os maridos possuam uma mediana um pouco maior. E o mesmo acontece com os quartis que se situam em idades um pouco superiores nos maridos.
- As idades dos maridos são um pouco mais distribuídas, especialmente na cauda direita, onde no histograma das esposas a idade para abruptamente nos 60 anos enquanto a dos maridos continua sendo observada até mais de 70 anos. # * Até aproximadamente 40 anos as esposas têm uma frequência maior dos que os maridos. A partir dos 40 ocorre o inverso e os maridos têm uma frequência maior.

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

```
freq_table_age = fdt(salarios$age)
print("Tabela de frequência Esposas")

## [1] "Tabela de frequência Esposas"

print(freq_table_age)

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

freq_table_husage = fdt(salarios$husage)
print("Tabela de frequência Maridos")

## [1] "Tabela de frequência Maridos"

print(freq_table_husage)

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

print(table(salarios$age))
```

```
##
## 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
## 12 18 31 47 47 67 84 114 114 174 161 170 184 191 187 180 204 187 205 217
## 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
## 202 171 181 158 172 185 185 140 164 128 131 125 133 117 138 103 109 104 91 104
## 58 59
## 104 95

print(table(salarios$husage))

##
## 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
## 5 6 25 31 35 71 59 115 113 108 137 157 172 174 169 196 161 174 195 169
## 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58
## 200 175 192 172 178 201 149 149 130 147 135 127 112 128 102 104 126 103 106 86
## 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78
## 91 81 74 62 45 45 30 18 14 14 13 8 7 9 6 2 7 4 2 2
## 79 80 81 85 86
## 2 1 1 1 1
```

Resposta/Análise:

- Pode-se observar o mesmo comportamento já observado nos boxplots e histogramas, onde a idade dos maridos é um pouco maior que a das esposas, com uma maior concentração de valores próximos da média e mediana.
- Ao analisar a idade do marido, observa-se que apresenta valores superiores, que destoam um pouco do restante dos dados (outliers), chegando até os 86 anos de idade.
- A idade das esposas é mais homogênea, com valores mais próximos da média e mediana, e com uma distribuição mais uniforme.
- A idade dos maridos é mais dispersa e mais assimétrica à direita, por conta dos outliers.

2 Medidas de posição e dispersão

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

```
# Calculando a média, mediana e moda para a variável "age" (esposas)
media_age <- mean(salarios$age)
mediana_age <- median(salarios$age)
moda_age <- as.numeric(names(which.max(table(salarios$age))))

# Calculando a média, mediana e moda para a variável "husage" (maridos)
media_husage <- mean(salarios$husage)
mediana_husage <- median(salarios$husage)
moda_husage <- as.numeric(names(which.max(table(salarios$husage))))

# Criando um dataframe para comparação
comparacao <- data.frame(Variavel = c("age", "husage"),
                          Media = c(media_age, media_husage),
                          Mediana = c(mediana_age, mediana_husage),
```

```

        Moda = c(moda_age, moda_husage),
        Diferenca_Media_Mediana = c(media_age-mediana_age, media_husage-media
na_husage))

# Exibindo o dataframe de comparação
print(comparacao)

```

##	Variavel	Media	Mediana	Moda	Diferenca_Media_Mediana
## 1	age	39.42758	39	37	0.4275825
## 2	husage	42.45296	41	44	1.4529641

Resposta/Análise:

- Como observado anteriormente, tanto a média quanto a mediana da idade do marido é maior, sendo a idade que mais se repete é 44 para o marido e 37 para a esposa.
- A diferença entre média e mediana é maior para a idade do marido, o que confirma o que já havia sido observado sobre a maior assimétrica para a direita, com valores mais altos puxando a média para cima.

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

```

# Calculando a variância, desvio padrão e coeficiente de variação para a variável "age" (espos
as)
var_age <- var(salarios$age)
dp_age <- sd(salarios$age)
cv_age <- dp_age/media_age

# Calculando a variância, desvio padrão e coeficiente de variação para a variável "husage" (ma
ridos)
var_husage <- var(salarios$husage)
dp_husage <- sd(salarios$husage)
cv_husage <- dp_husage/media_husage

# Criando dataframe
comparacao2 <- data.frame(Variavel = c("age", "husage"),
                          Variancia = c(var_age, var_husage),
                          Desvio_Padrao = c(dp_age, dp_husage),
                          Coeficiente_de_Variacao = c(cv_age, cv_husage))

print(comparacao2)

```

##	Variavel	Variancia	Desvio_Padrao	Coeficiente_de_Variacao
## 1	age	99.75234	9.98761	0.2533153
## 2	husage	126.07173	11.22817	0.2644849

Resposta/Análise:

- A variância, desvio padrão e coeficiente de variação da idade do marido são maiores, o que confirma a maior dispersão dos dados e a maior assimetria à direita.

- O coeficiente de variação da idade do marido é maior, o que indica uma maior dispersão dos dados em relação à média.

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.

```
# Ajustando o formato da base de dados
# Criando um dataframe para as esposas
id_esposa <- data.frame(idade = salarios$age, grupo = "esposa")
# Criando um dataframe para os maridos
id_marido <- data.frame(idade = salarios$husage, grupo = "marido")
# Combinar os dataframes em um único dataframe
idades <- rbind(id_esposa, id_marido)
rm(id_esposa)
rm(id_marido)
rm(salarios)
```

Premissa 1: As duas amostras são independentes?

Sim, pois os grupos de marido e esposa não estão relacionados. Não se trata de uma amostra ou grupos emparelhados.

Premissa 2: Os dados de cada amostra/grupo possuem distribuição normal?

Vamos usar o teste de normalidade Shapiro-Wilk com o seguinte teste de hipóteses: * H0: os dados são normalmente distribuídos * Ha: os dados não são normalmente distribuídos

```
# Primeiro vamos fazer o teste de normalidade Kolmogorov-Smirnov
# Teste de Kolmogorov-Smirnov (ESPOSA)
ks.test(idades$idade[idades$grupo == "esposa"], "pnorm", mean = mean(idades$idade[idades$grupo == "esposa"]), sd = sd(idades$idade[idades$grupo == "esposa"]))

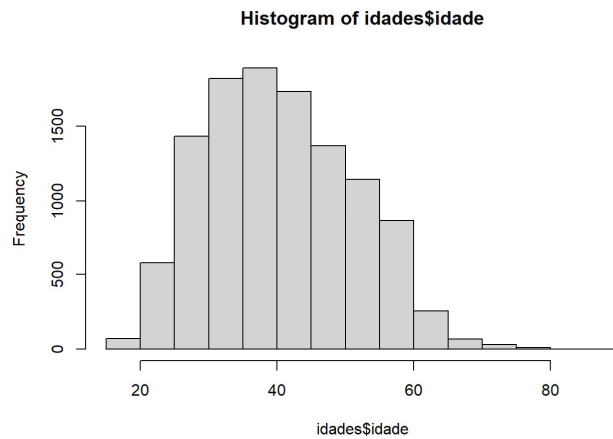
## ## Asymptotic one-sample Kolmogorov-Smirnov test
## ## data: idades$idade[idades$grupo == "esposa"]
## D = 0.058909, p-value < 2.2e-16
## alternative hypothesis: two-sided

# p-value < 2.2e-16, logo o grupo das esposas não possui distribuição normal
# Teste de Kolmogorov-Smirnov (ESPOSA)
ks.test(idades$idade[idades$grupo == "marido"], "pnorm", mean = mean(idades$idade[idades$grupo == "marido"]), sd = sd(idades$idade[idades$grupo == "marido"]))

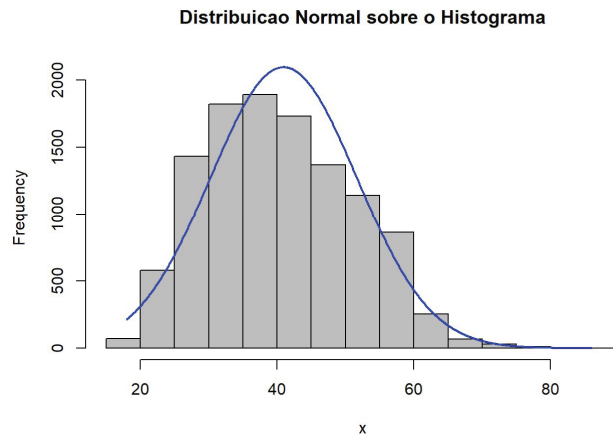
##
## Asymptotic one-sample Kolmogorov-Smirnov test
##
## data: idades$idade[idades$grupo == "marido"]
## D = 0.059662, p-value < 2.2e-16
```



```
## alternative hypothesis: two-sided
# p-value < 2.2e-16, logo o grupo dos maridos não possui distribuição normal
# Vamos gerar um histograma para confirmar visualmente:
hist(idades$idade) # Histograma de todas as idades
```



```
# Vamos plotar a curva normal sobre o histograma para visualizar a diferença:
plotNormalHistogram(idades$idade, prob = FALSE,
                    main = "Distribuicao Normal sobre o Histograma",
                    length = 1000 )
```



```
lillie.test(idades$idade) # toda a amostra nao segue um distribuição normal
## ## Lilliefors (Kolmogorov-Smirnov) normality test
## ## data: idades$idade
## D = 0.060083, p-value < 2.2e-16
```

Como nao passa no teste de normalidade, deveremos usar o teste não paramétrico

Teste de Mann-Whitney:

- H0: A idade mediana do marido e igual estatisticamente a idade mediana das esposas
- H1: A idade mediana do marido nao e igual estatisticamente a idade mediano das esposas

```
wilcox.test(idade ~ grupo, data = idades, exact = FALSE, conf.int = TRUE)

##
## Wilcoxon rank sum test with continuity correction
##
## data: idade by grupo
## W = 13619912, p-value < 2.2e-16
## 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
```

p-value = $< 2.2e-16$, logo rejeita-se H_0 . Portanto, a idade mediana dos maridos não é estatisticamente igual ao das esposas

Resposta/Análise:

- Como a distribuição dos dados não é normal, foi utilizado o teste não paramétrico de Mann-Whitney.
- O teste de Mann-Whitney rejeita a hipótese nula de que as medianas das idades dos maridos e esposas são iguais.
- O intervalo de confiança para a diferença das medianas não inclui o zero, o que reforça a conclusão de que as medianas não são iguais.
- A idade mediana dos maridos não é estatisticamente igual ao das esposas.

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

1) 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

Bibliotecas:

```
library(plyr)
library(readr)
library(dplyr)
library(caret)
```

```
library(ggplot2)
library(repr)
library(glmnet)
```

Lendo a base de dados

```
load("./trabalhosalarios.RData")
df <- trabalhosalarios
head(df,10)
```

##	husage	husunion	husearns	huseduc	husblk	hushisp	hushrs	kidge6	earns	age
## 3	56	0	1500	14	0	0	40	1	100	49
## 13	31	0	800	17	0	0	40	0	480	29
## 20	33	0	950	13	0	0	60	0	455	30
## 21	34	0	1000	14	0	0	50	1	102	31
## 22	42	0	730	14	0	0	40	1	300	41
## 25	45	0	1154	16	0	0	38	1	425	45
## 26	33	0	1350	16	0	0	40	0	770	32
## 27	31	0	769	18	0	0	55	0	125	27
## 29	31	0	340	12	0	0	40	0	245	30
## 31	44	0	750	12	0	0	40	1	539	42

##	black	educ	hispanic	union	exper	kidlt6	lwage
## 3	0	12	0	0	31	0	1.897120
## 13	0	14	0	0	9	0	2.484907
## 20	0	12	0	0	12	1	2.431418
## 21	0	12	0	0	13	0	1.629241
## 22	0	12	0	0	23	0	2.302585
## 25	0	18	0	0	21	0	2.496741
## 26	0	12	0	0	14	0	2.957511
## 27	0	14	0	0	7	1	1.783791
## 29	0	15	0	0	9	1	1.945910
## 31	0	12	0	0	24	0	2.600836

```
rm(trabalhosalarios )
```

1 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 R2) e concluir qual o melhor modelo entre os três, e mostrar o resultado da predição com intervalos de confiança para os seguintes valores:

- husage = 40 (anos – idade do marido)

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

1.1) Pré-processamento

```
# removendo uma coluna que nao irei utilizar
df$earns <- NULL

set.seed(24)

# Criar índices para a partição dos dados
indices <- createDataPartition(df$lwage, p = 0.8, list = FALSE)

# Criar conjuntos de treinamento e teste
df_train <- df[indices, ]
df_test <- df[-indices, ]

cols = c("husage", "husearns", "huseduc", "hushrs", "age", "educ", "exper", "lwage")

# Padronizando a base de treinamento e teste
pre_proc_val <- preProcess(df_train[,cols],
                           method = c("center", "scale"))

df_train[,cols] = predict(pre_proc_val, df_train[,cols])
df_test[,cols] = predict(pre_proc_val, df_test[,cols])

cols_reg = c('husage', 'husunion', 'husearns', 'huseduc', 'husblck', 'hushisp', 'hushrs', 'kidge6', 'age', 'black', 'educ', 'hispanic', 'union', 'exper', 'kidlt6', 'lwage')

# Estamos interessados em estimar o salario-hora esposa em log (hrwage)
dummies <- dummyVars(lwage~., data = df[,cols_reg])
train_dummies = predict(dummies, newdata = df_train[,cols_reg])
test_dummies = predict(dummies, newdata = df_test[,cols_reg])
print(dim(train_dummies)); print(dim(test_dummies))

## [1] 2061 15
## [1] 513 15

# variaveis explicativas para o modelo em um objeto
x = as.matrix(train_dummies)

# variavel dependente para o modelo em um objeto
y_train = df_train$lwage

# explicativas para o modelo em um objeto chamado
x_test = as.matrix(test_dummies)

# dependente para o modelo em um objeto chamado "y_test"
```

```

y_test = df_test$lwage

# Vamos testar os lambdas de 10^-3 ate 10^2, a cada 0.1
lambdas <- 10^seq(2, -3, by = -.1)

# Vamos construir uma matriz de dados para a predicao
our_pred = as.matrix(data.frame(husage = scale(40, center = pre_proc_val[["mean"]][["husage"]],
                                scale = pre_proc_val[["std"]][["husage"]]),
                        husunion = 0,
                        husearns = scale(600, center = pre_proc_val[["mean"]][["husearns"]],
                                scale = pre_proc_val[["std"]][["husearns"]]),
                        huseduc = scale(13, center = pre_proc_val[["mean"]][["huseduc"]],
                                scale = pre_proc_val[["std"]][["huseduc"]]),
                        husblk = 1,
                        hushisp = 0,
                        hushrs = scale(40, center = pre_proc_val[["mean"]][["hushrs"]],
                                scale = pre_proc_val[["std"]][["hushrs"]]),
                        kidge6 = 1,
                        age = scale(38, center = pre_proc_val[["mean"]][["age"]],
                                scale = pre_proc_val[["std"]][["age"]]),
                        black = 0,
                        educ = scale(13, center = pre_proc_val[["mean"]][["educ"]],
                                scale = pre_proc_val[["std"]][["educ"]]),
                        hispanic = 1,
                        union = 0,
                        exper = scale(18, center = pre_proc_val[["mean"]][["exper"]],
                                scale = pre_proc_val[["std"]][["exper"]]),
                        kidlt6 = 1))

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

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

```

1.2) Regressão Ridge

```

# Calculando o lambda:
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0,
                        lambda = lambdas)

```

```

# Vamos ver qual o lambda otimo
best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge

## [1] 0.07943282

# Para contar o tempo de processamento do modelo usamos:
start <- Sys.time()

# Estimando o modelo Ridge
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
                  family = 'gaussian',
                  lambda = best_lambda_ridge)

end <- Sys.time()
difftime(end, start, units="secs")

## Time difference of 0.003528118 secs

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

## 15 x 1 sparse Matrix of class "dgCMatrix"
##
##          s0
## husage    0.02656736
## husunion -0.07467661
## husearns  0.22604883
## huseduc   0.07046174
## husblck   0.07217368
## hushisp   0.11734775
## hushrs    -0.06812692
## kidge6    -0.10185700
## age       0.04429239
## black     -0.04380116
## educ      0.28069839
## hispanic  -0.07534998
## union     0.42258527
## exper     -0.02820302
## kidlt6    -0.04704322

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

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

##          RMSE    Rsquare
## 1 0.8491541 0.2785873

# Predicao e avaliacao nos dados de teste:

```

```

predictions_test <- predict(ridge_reg,
                           s = best_lambda_ridge,
                           newx = x_test)

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

##           RMSE    Rsquare
## 1 0.8239824 0.3019891

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

# O resultado da predicao eh:
predict_our_ridge

##           s1
## [1,] -0.2049218

# Despadronizando para ter o valor do salario original
wage_pred_ridge=(predict_our_ridge*
                  pre_proc_val[["std"]][["lwage"]])+
                  pre_proc_val[["mean"]][["lwage"]]

# O resultado eh:
wage_pred_ridge

##           s1
## [1,] 2.086919

exp(wage_pred_ridge) # valor real

##           s1
## [1,] 8.06004

# Este eh o valor predito do salario por hora (US$),
# segundo as caracteristicas que atribuimos

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

# Criar um dataframe para armazenar as médias
Ridge_padronizadas <- data.frame(

```



```

Model = 'Ridge Padronizada',
Lambda = best_lambda_ridge,
RMSE_treino = eval_results(y_train, predictions_train, df_train)[1],
Rsquare_treino = eval_results(y_train, predictions_train, df_train)[2],
RMSE_teste = eval_results(y_test, predictions_test, df_test)[1],
Rsquare_teste = eval_results(y_test, predictions_test, df_test)[2],
Valor_predito = exp(wage_pred_ridge),
int.conf1.inf = exp(CI1wr_ridge), # esta com o valor padronizado, teria que ajustar aqui
int.conf1.up = exp(CI1upr_ridge), # esta com o valor padronizado, teria que ajustar aqui
stringsAsFactors = FALSE
)

# Atribuir nomes às colunas
names(Ridge_padronizadas) <- c("Model", "Lambda", "RMSE_treino", "Rsquare_treino", "RMSE_teste",
, "Rsquare_teste", "Valor_predito", "int.conf1.inf", "int.conf1.up")

```

1.3) Regressão de Lasso :

```

# Vamos atribuir alpha = 1 para implementar a regressao
# lasso
lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
                        lambda = lambdas,
                        standardize = TRUE, nfolds = 5)

# Vamos guardar o lambda "otimo" em um objeto chamado
# best_lambda_lasso
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso

## [1] 0.01258925

# Vamos estimar o modelo Lasso
lasso_model <- glmnet(x, y_train, alpha = 1,
                      lambda = best_lambda_lasso,
                      standardize = TRUE)

# Vamos visualizar os coeficientes estimados
lasso_model[["beta"]]

## 15 x 1 sparse Matrix of class "dgCMatrix"

##              s0
## husage      0.015648453
## husunion -0.050283412
## husearns  0.235385985
## huseduc   0.043459439
## husblck   .
## hushisp   .

```

```

## hushrs    -0.063395131
## kidge6    -0.068338201
## age       0.022140728
## black     .
## educ      0.307925589
## hispanic  .
## union     0.412547221
## exper     .
## kidlt6    -0.009068894

# Vamos fazer as predicoes na base de treinamento e
# avaliar a regressao Lasso
predictions_train <- predict(lasso_model,
                             s = best_lambda_lasso,
                             newx = x)

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

##          RMSE    Rsquare
## 1 0.849695 0.2776679

# Vamos fazer as predicoes na base de teste
predictions_test <- predict(lasso_model,
                            s = best_lambda_lasso,
                            newx = x_test)

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

##          RMSE    Rsquare
## 1 0.8239465 0.3020499

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

predict_our_lasso

##          s1
## [1,] -0.1542719

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

```

```
##          s1
## [1,] 2.113432

# Portanto, o salario-hora da esposa eh US$10.36
# Vamos criar o intervalo de confianca para o nosso
# exemplo
n <- nrow(df_train)
m <- wage_pred_lasso
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
Cilwr_lasso <- m + (qnorm(0.025))*dam
CIupr_lasso <- m - (qnorm(0.025))*dam

# Criar um dataframe para armazenar as médias
lasso_padronizadas <- data.frame(
  Model = 'Lasso Padronizada',
  Lambda = best_lambda_lasso,
  RMSE_treino = eval_results(y_train, predictions_train, df_train)[1],
  Rsquare_treino = eval_results(y_train, predictions_train, df_train)[2],
  RMSE_teste = eval_results(y_test, predictions_test, df_test)[1],
  Rsquare_teste = eval_results(y_test, predictions_test, df_test)[2],
  Valor_predito = exp(wage_pred_lasso),
  int.conf.inf = exp(Cilwr_lasso), # esta com o valor padronizado, teria que ajustar aqui
  int.conf.up = exp(CIupr_lasso), # esta com o valor padronizado, teria que ajustar aqui
  stringsAsFactors = FALSE
)

# Atribuir nomes às colunas
names(lasso_padronizadas) <- c("Model", "Lambda", "RMSE_treino", "Rsquare_treino", "RMSE_teste",
  "Rsquare_teste", "Valor_predito", "int.conf.inf", "int.conf.up")

df_final = rbind(Ridge_padronizadas, lasso_padronizadas)
```

1.4) Regressão ElasticNet

```
# Vamos configurar o treinamento do modelo por
# cross validation, com 10 folders, 5 repeticoes
# e busca aleatoria dos componentes das amostras
# de treinamento, o "verboseIter" eh soh para
# mostrar o processamento.

train_cont <- trainControl(method = "repeatedcv",
  number = 15,
  repeats = 8,
  search = "random",
  verboseIter = TRUE)
```

```

# Vamos treinar o modelo
elastic_reg <- train(lwage ~ .,
                     data = df_train,
                     method = "glmnet",
                     tuneLength = 10,
                     trControl = train_cont)

## + Fold01.Rep1: alpha=0.5527, lambda=0.686609
## - Fold01.Rep1: alpha=0.5527, lambda=0.686609
(...)
## - Fold15.Rep8: alpha=0.4182, lambda=0.002922
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0.979, lambda = 0.01 on full training set

# O melhor parametro alpha escolhido eh:
elastic_reg$bestTune[2]

##          lambda
## 10 0.01003417

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

## 15 x 74 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 74 column names 's0', 's1', 's2' ... ]]
##
## husage      . .      .      .      .      .
## husunion . .      .      .      .      .
## husearns . .      .      0.0001595961 0.0222535 0.04239022 0.06075354
(...)
## exper      .      .      .      .      .
## kidlt6     -0.05320073 -0.05341459 -0.05361361 -0.05379533 -0.05393864

# Vamos fazer as predições e avaliar a performance do
# modelo
# Vamos fazer as predições no modelo de treinamento:
predictions_train <- predict(elastic_reg, x)
# As metricas de performance na base de treinamento
# sao:
eval_results(y_train, predictions_train, df_train)

##          RMSE  Rsquare
## 1 0.8493444 0.278264

# Vamos fazer as predições na base de teste
predictions_test <- predict(elastic_reg, x_test)
# As metricas de performance na base de teste sao:
eval_results(y_test, predictions_test, df_test)

```

```
##          RMSE    Rsquare
## 1 0.8236324 0.3025818

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

## [1] -0.1682915

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

## [1] 2.106093

# Então o salário-hora médio da esposa predito com base
# nas características informadas é US$9.28
# Vamos criar o intervalo de confiança para o nosso
# exemplo
n <- nrow(df_train)
m <- wage_pred_elastic
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_elastic <- m + (qnorm(0.025))*dam
CIupr_elastic <- m - (qnorm(0.025))*dam

# Criar um dataframe para armazenar as médias
eladticnet_padronizadas <- data.frame(
  Model = 'ElasticNet Padronizada',
  Lambda = elastic_reg$bestTune[2],
  RMSE_treino = eval_results(y_train, predictions_train, df_train)[1],
  Rsquare_treino = eval_results(y_train, predictions_train, df_train)[2],
  RMSE_teste = eval_results(y_test, predictions_test,df_test)[1],
  Rsquare_teste = eval_results(y_test, predictions_test,df_test)[2],
  Valor_predito = exp(wage_pred_elastic),
  int.conf.inf = exp(CIlwr_elastic), # esta com o valor padronizado, teria que ajeitar aqui
  int.conf.up = exp(CIupr_elastic), # está com o valor padronizado, teria que ajeitar aqui
  stringsAsFactors = FALSE
)

# Atribuir nomes às colunas
names(eladticnet_padronizadas) <- c("Model", "Lambda", "RMSE_treino", "Rsquare_treino", "RMSE_t
este", "Rsquare_teste", "Valor_predito", "int.conf.inf", "int.conf.up")
```

```
df_final = rbind(df_final, eladticnet_padronizadas)
```

1.4) Conclusão

df_final					
##	Model	Lambda	RMSE_treino	Rsquare_treino	RMSE_teste
## 1	Ridge Padronizada	0.07943282	0.8491541	0.2785873	0.8239824
## 2	Lasso Padronizada	0.01258925	0.8496950	0.2776679	0.8239465
## 10	ElasticNet Padronizada	0.01003417	0.8493444	0.2782640	0.8236324
##	Rsquare_teste	Valor_predito	int.conf1.inf	int.conf1.up	
## 1	0.3019891	8.060040	7.879930	8.244267	
## 2	0.3020499	8.276599	8.091650	8.465776	
## 10	0.3025818	8.216081	8.032484	8.403875	

R: Todos os modelos ficaram bem próximos um do outro em relação ao RMSE e R2. Logo, a escolha final será do modelo Ridge Padronizada pois apresentou os melhores valores de RMSE e R2, além de ser o modelo que rodou mais rápido.

APÊNDICE 6 – ARQUITETURA DE DADOS

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) Identificador automático de idioma

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

Entrada: "texto qualquer"

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

O processo de Reconhecimento de Padrões

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

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

#

amostras de texto em diferentes línguas

#

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

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

portugues = ["Estou indo para o trabalho agora.", "Adoro passar tempo com minha família.", "Preciso comprar leite e pão.", "Vamos ao cinema no sábado.", "Gosto de praticar esportes ao ar livre.", "O trânsito está terrível hoje.", "A comida estava deliciosa!", "Você já visitou o Rio de Janeiro?", "Tenho uma reunião importante amanhã.", "A festa começa às 20h.", "Estou cansado depois de um longo dia de trabalho.", "Vamos fazer um churrasco no final de semana.", "O livro que estou lendo é muito interessante.", "Estou aprendendo a cozinhar pratos novos.", "Preciso fazer exercícios físicos regularmente.", "Vou viajar para o exterior nas férias.", "Você gosta de dançar?", "Hoje é meu aniversário!", "Gosto de ouvir música clássica.", "Estou estudando para o vestibular.", "Meu time de futebol favorito ganhou o jogo.", "Quero aprender a tocar violão.", "Vamos fazer uma viagem de carro.", "O parque fica cheio aos finais de semana.", "O filme que assisti ontem foi ótimo.", "Preciso resolver esse problema o mais rápido possível.", "Adoro explorar novos lugares.", "Vou visitar meus avós no domingo.", "Estou ansioso para as férias de verão.", "Gosto de fazer caminhadas na natureza.", "O restaurante tem uma vista incrível.", "Vamos sair para jantar no sábado."]

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

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

import random

pre_padroes = []

for frase **in** ingles:

pre_padroes.append([frase, 'inglês'])

for frase **in** espanhol:

pre_padroes.append([frase, 'espanhol'])

for frase in portugueses:

```
pre_padroes.append([frase, 'português'])
```

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

```
#print(pre_padroes)
```

```
[['I enjoy playing soccer.', 'inglês'], ['Este es mi libro favorito.', 'espanhol'], ['¿Dónde está el restaurante más cercano?', 'espanhol'], ['Let's go for a walk.', 'inglês'], ['Estou indo para o trabalho agora.', 'português'], ['Have a great day!', 'inglês'], ['Quero aprender a tocar violão.', 'português'], ['Estoy emocionado por el concierto.', 'espanhol'], ['Estou ansioso para as férias de verão.', 'português'], ['I want to learn French.', 'inglês'], ['Vou visitar meus avós no domingo.', 'português'], ['Ella canta hermosamente.', 'espanhol'], ['I love to read books.', 'inglês'], ['¿Cuál es tu comida favorita?', 'espanhol'], ['I like to cook dinner at home.', 'inglês'], ['¿Que tengas un buen día!', 'espanhol'], ['Preciso fazer exercícios físicos regularmente.', 'português'], ['I enjoy going to the beach.', 'inglês'], ['O trânsito está terrível hoje.', 'português'], ['A comida estava deliciosa!', 'português'], ['Tengo una reunión a las 2 PM.', 'espanhol'], ['Voy al parque todos los días.', 'espanhol'], ['Where can I find a taxi?', 'inglês'], ['I'm learning to play the guitar.', 'inglês'], ['Preciso comprar leite e pão.', 'português'], ['¿Puedes ayudarme con esto?', 'espanhol'], ['Necesito comprar algunas frutas.', 'espanhol'], ['Lamento las molestias.', 'espanhol'], ['I'm planning a vacation.', 'inglês'], ['Where is the nearest restaurant?', 'inglês'], ['Me gustaría ir de vacaciones.', 'espanhol'], ['¿Dónde puedo encontrar un taxi?', 'espanhol'], ['Você já visitou o Rio de Janeiro?', 'português'], ['Preciso resolver esse problema o mais rápido possível.', 'português'], ['O livro que estou lendo é muito interessante.', 'português'], ['¿Hablas español?', 'espanhol'], ['Gosto de praticar esportes ao ar livre.', 'português'], ['Estoy estudiando para mis exámenes.', 'espanhol'], ['O restaurante tem uma vista incrível.', 'português'], ['Estou estudando para o vestibular.', 'português'], ['¿Me pasas la sal, por favor?', 'espanhol'], ['The weather is nice today.', 'inglês'], ['A festa começa às 20h.', 'português'], ['Me gusta cocinar la cena en casa.', 'espanhol'], ['Estou aprendendo a cozinhar pratos novos.', 'português'], ['Quiero aprender italiano.', 'espanhol'], ['¿Tienes alguna recomendación de restaurantes?', 'espanhol'], ['Meu time de futebol favorito ganhou o jogo.', 'português'], ['I'm going to the movies tonight.', 'inglês'], ['Do you have any recommendations for restaurants?', 'inglês'], ['This is a beautiful place.', 'inglês'], ['Gosto de ouvir música clássica.', 'português'], ['I like listening to music.', 'inglês'], ['Do you speak English?', 'inglês'], ['Hello, how are you?', 'inglês'], ['Tenho uma reunião importante amanhã.', 'português'], ['Could you pass me the salt, please?', 'inglês'], ['She sings beautifully.', 'inglês'], ['Hoje é meu aniversário!', 'português'], ['Me gusta bailar salsa.', 'espanhol'], ['How was your weekend?', 'inglês'], ['What time is it?', 'inglês'], ['Vamos ao cinema no sábado.', 'português'], ['Me encanta leer libros.', 'espanhol'], ['O filme que assisti ontem foi ótimo.', 'português'], ['I'm excited for the concert.', 'inglês'], ['I have a meeting at 2 PM.', 'inglês'], ['I'm sorry for the inconvenience.', 'inglês'], ['Adoro explorar novos lugares.', 'português'], ['I'm studying for my exams.', 'inglês'], ['Disfruto ir a la playa.', 'espanhol'], ['Estoy aprendiendo a tocar el piano.', 'espanhol'], ['¿Cómo estuvo tu fin de semana?', 'espanhol'], ['El perro está jugando.', 'espanhol'], ['The cat is sleeping.', 'inglês'], ['Estou cansado depois de um longo dia de trabalho.', 'português'], ['El clima está agradable hoy.', 'espanhol'], ['Can you help me with this?', 'inglês'], ['Hola, ¿cómo estás?', 'espanhol'], ['What is your favorite color?', 'inglês'], ['Vou viajar para o exterior nas férias.', 'português'], ['I need to buy some groceries.', 'inglês'], ['Vamos a dar un paseo.', 'espanhol'], ['Vamos fazer uma viagem de carro.', 'português'], ['Vamos fazer um churrasco no final de semana.', 'português'], ['Gosto de fazer caminhadas na natureza.', 'português'], ['Adoro passar tempo com minha família.', 'português'], ['Você gosta de dançar?', 'português'], ['¿Qué hora es?', 'espanhol'], ['Estoy planeando unas vacaciones.', 'espanhol'], ['Vamos sair para jantar no sábado.', 'português'], ['O parque fica cheio aos finais de semana.', 'português']]
```

O DataFrame do pandas facilita a visualização.

```
import pandas as pd
```

```
nomes_colunas = ['frase', 'idioma']
```

```
dados = pd.DataFrame(pre_padroes, columns=nomes_colunas)
```

```
dados
```

	frase	idioma
0	I enjoy playing soccer.	inglês
1	Este es mi libro favorito.	espanhol
...
91	O parque fica cheio aos finais de semana.	português

92 rows × 2 columns

Construção dos atributos

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

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

Função para verificar se a frase contém "ing"

```
def contains_ing(text):
    if "ing" in text:
        return 1
    else:
        return 0
```

Aplicando a função ao DataFrame

```
dados['Contem_ing'] = dados['frase'].apply(contains_ing)
```

Função para verificar se a frase contém "oy"

```
def contains_oy(text):
    if "oy" in text:
        return 1
    else:
        return 0
```

Aplicando a função ao DataFrame

```
dados['Contem_oy'] = dados['frase'].apply(contains_oy)
```

Função para verificar se a frase contém "oy"

```
def contains_carc_especial(text):
    if "i" in text or "ç" in text:
        return 1
    else:
        return 0
```

Aplicando a função ao DataFrame

```
dados['Contem_carc_especial'] = dados['frase'].apply(contains_carc_especial)
```

```
dados.head(2)
```

	frase	idioma	Contem_ing	Contem_oy	Contem_carc_especial
0	I enjoy playing soccer.	inglês	1	1	0
1	Este es mi libro favorito.	espanhol	0	0	0

Listas de palavras comuns em cada idioma

```
ingles = set(["the", "is", "in", "and", "to", "a", "it", "you", "of", "for", "on", "with", "this", "that", "by", "an",
"be", "are", "was", "as"])
portugues = set(["o", "é", "em", "e", "para", "um", "ele", "você", "de", "por", "com", "isso", "que", "um", "ser",
"são", "foi", "como"])
espanhol = set(["el", "al", "la", "es", "en", "y", "para", "un", "él", "tú", "de", "por", "con", "esto", "que", "un",
"ser", "son", "fue", "como"])
```

```
def count_palavras(texto, palavras):
    words = texto.lower().split()
    count = sum(1 for word in words if word in palavras)
    return count
def detectar_lingua(text):
    ingles_count = count_palavras(text, ingles)
    portugues_count = count_palavras(text, portugues)
```

```

espanhol_count = count_palavras(text, espanhol)

if ingles_count > portugues_count and ingles_count > espanhol_count:
    return 1 #'Ingles'
elif portugues_count > ingles_count and portugues_count > espanhol_count:
    return 2 #'Portuqgues'
elif espanhol_count > ingles_count and espanhol_count > portugues_count:
    return 3 #'Espanhol'
else:
    return 2 # se nao achar, escolhe portugues como default

dados['detector_lingua'] = dados['frase'].apply(detectar_lingua)

dados2 = dados.copy()
# a entrada é o vetor pré padrões e a saída desse passo deverá ser "padrões"
import re
def tamanhoMedioFrases(texto):
    palavras = re.split("\s", texto)
    #print(palavras)
    tamanhos = [len(s) for s in palavras if len(s)>0]
    #print(tamanhos)
    soma = 0
    for t in tamanhos:
        soma=soma+t
    return soma / len(tamanhos)
def funcao2(frase):
    # faz alguma coisa, mede algo
    return 0
def extraiCaracteristicas(frase):
    # frase é um vetor [ 'texto', 'lingua' ]
    texto = frase[0]
    pattern_regex = re.compile('[^\w+]', re.UNICODE)
    texto = re.sub(pattern_regex, '', texto)
    #print(texto)
    caracteristica1=tamanhoMedioFrases(texto)
    caracteristica2=funcao2(texto)
    caracteristica3=funcao2(texto)
    # acrescenta as suas funcoes no vetor padrao
    padrao = [caracteristica1, caracteristica2, caracteristica3, frase[1] ]
    return padrao
def geraPadroes(frases):
    padroes = []
    for frase in frases:
        padrao = extraiCaracteristicas(frase)
        padroes.append(padrao)
    return padroes

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

dados = pd.DataFrame(padroes)
dados


```

	0	1	2	3
0	4.750000	0	0	inglês

	0	1	2	3
1	4.200000	0	0	espanhol
...
91	4.125000	0	0	português

92 rows × 4 columns

Lista com os novos nomes das colunas

```
novos_nomes_colunas = ['cont_media', 'caracteristicas', 'padroes', 'idioma']
```

Renomeando todas as colunas

```
dados.columns = novos_nomes_colunas
dados.head(2)
```

	cont_media	caracteristicas	padroes	idioma
0	4.75	0	0	inglês
1	4.20	0	0	espanhol

dados2.columns

```
Index(['frase', 'idioma', 'Contem_ing', 'Contem_oy', 'Contem_carc_especial',
      'detector_lingua'],
      dtype='object')
```

```
def freqCaracteres(texto, lower=True):
```

```
    freq = {}
```

```
    for caracter in texto:
```

```
        if lower:
```

```
            caracter = caracter.lower()
```

```
            key = f'freq_{caracter}'
```

```
            freq[key] = freq.get(key, 0) + 1
```

```
    return freq
```

```
def freqCaracteres_df(textos, lower=True):
```

```
    df = pd.DataFrame()
```

```
    for texto in textos:
```

```
        texto = texto[0]
```

```
        df1 = pd.DataFrame.from_dict(freqCaracteres(texto), orient='index').transpose()
```

```
        df = pd.concat([df, df1], axis=0)
```

```
    return df.fillna(0)
```

começa com pontuação?

```
def comeca_pontuacao(frase):
```

```
    if frase[0].strip() in ['.', ',']:
        return 1
```

```
    else:
```

```
        return 0
```

#encontros vocálicos e consonantais e outros padrões como ing.

```
def encontros(frase):
```

```
    lista_padroes = ["ch", "nh", "th", "sh", "ll", "rr", "lh", "nn", "tr", "ss", "ie", "ing", "el"]
```

```
    res = []
```

```
    for c in lista_padroes:
```

```
        res.append(frase.lower().count(c.lower()))
```

```

    return res
#base para a contagem ocorrência das diversas classes de caracteres
def qtde(frase, lista_caracteres):
    qtde = 0
    for c in frase.lower():
        if c.lower() in lista_caracteres:
            qtde = qtde + 1
    return qtde
#vogais
def qtde_vogais(frase):
    lista = ['a','e','i','o','u']
    return qtde(frase, lista)
#pontuacao
def qtde_pontuacao(frase):
    lista = [',',';','?','!']
    return qtde(frase, lista)
#pontuacao especial
def qtde_pontuacao_especial(frase):
    lista = ['¿','¡']
    return qtde(frase, lista)
#acentuados
def qtde_acentuados(frase):
    lista = ['u','á','u','é','u','í','u','ó','u','ú','u','ã','u','â','u','ê','u','õ','u','ô']
    return qtde(frase, lista)
#acentuados especiais
def qtde_acentuados_especiais(frase):
    lista = ['u','ñ']
    return qtde(frase, lista)

#consoantes
def qtde_consoantes(frase):
    lista = ['b','c','d','f','g','h','j','k','l','m','n','p','q','r','s','t','v','w','x','y','z']
    return qtde(frase, lista)
def tamanhoMedioFrases(texto):
    palavras = re.split("\s",texto)
    #print(palavras)
    tamanhos = [len(s) for s in palavras if len(s)>0]
    #print(tamanhos)
    soma = 0
    for t in tamanhos:
        soma=soma+t
    return soma / len(tamanhos)
def extraiCaracteristicas(frase):
    # frase é um vetor [ 'texto', 'lingua' ]
    texto = frase[0]
    caracteristicas = []
    caracteristicas = [comeca_pontuacao(texto)]
    pattern_regex = re.compile('[^\w+]', re.UNICODE)
    texto = re.sub(pattern_regex, '', texto)
    #print(texto)
    caracteristica1=tamanhoMedioFrases(texto)
    # acrescente as suas funcoes no vetor padrao
    caracteristicas = caracteristicas + [qtde_vogais(texto),
        qtde_pontuacao(texto),
        qtde_acentuados(texto),
        qtde_acentuados_especiais(texto),
        qtde_consoantes(texto)]
    caracteristicas = caracteristicas + encontros(texto)
    padrao = caracteristicas+[caracteristica1, frase[1]]

```


	0	1	2	3	4	5	6	7	8	9	.	fre q_	fre q_ç	fre q_à	fre q_ó	fre q_ú	20	Conte m_ing	Conte m_oy	Contem_car c_especial	detector _lingua
8	0	1	0	0	0	1	0	0	0	0	.	0.0	0.0	0.0	0.0	0.0	espan hol	0	1	0	2
9	0	1	0	1	0	1	0	0	0	0	.	0.0	0.0	0.0	0.0	0.0	portu guês	0	0	0	2
9	0	1	0	0	0	1	1	0	0	0	.	0.0	0.0	0.0	0.0	0.0	portu guês	0	0	0	2

92 rows × 73 columns

`dados.columns`

```
Index(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12',
      '13', '14', '15', '16', '17', '18', '19', 'index', 'freq_i', 'freq_ ',
      'freq_e', 'freq_n', 'freq_j', 'freq_o', 'freq_y', 'freq_p', 'freq_l',
      'freq_a', 'freq_g', 'freq_s', 'freq_c', 'freq_r', 'freq_', 'freq_t',
      'freq_m', 'freq_b', 'freq_f', 'freq_v', 'freq_ç', 'freq_d', 'freq_ó',
      'freq_á', 'freq_u', 'freq_?', 'freq_', 'freq_w', 'freq_k', 'freq_h',
      'freq_!', 'freq_q', 'freq_ã', 'freq_é', 'freq_í', 'freq_í', 'freq_z',
      'freq_x', 'freq_â', 'freq_2', 'freq_ê', 'freq_ñ', 'freq_', 'freq_ç',
      'freq_à', 'freq_0', 'freq_ú', '20', 'Contem_ing', 'Contem_oy',
      'Contem_carc_especial', 'detector_lingua'],
      dtype='object')
```

Treinando o modelo com SVM

Separando o conjunto de treinamento do conjunto de testes

```
vet = np.array(dados)
vet
array([[0, 7, 0, ..., 1, 0, 2],
      [0, 10, 0, ..., 0, 0, 3],
      [1, 11, 0, ..., 0, 1, 3],
      ...,
      [0, 13, 0, ..., 1, 0, 2],
      [0, 11, 0, ..., 0, 0, 2],
      [0, 18, 0, ..., 0, 0, 2]], dtype=object)
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report
import numpy as np
def treinaModelo(padroes):
    vet = np.array(padroes)
    classes = padroes['20'] # classes = [p[-1] for p in padroes]
    #print(classes)
    padroes_sem_classe = padroes.drop(['20'], axis = 1) #np.delete(vet, 3, axis=1)
    #print(padroes_sem_classe)
    X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe, classes, test_size=0.25,
    stratify=classes)

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



```

modelo = treinador.fit(X_train, y_train)

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

#
# melhor avaliar com a matriz de confusão
y_pred = modelo.predict(X_train)
print('_'*60)
print('classes: ', modelo.classes_)
cm = confusion_matrix(y_train, y_pred, labels=modelo.classes_)
print(cm)
print(classification_report(y_train, y_pred))
ConfusionMatrixDisplay(confusion_matrix=cm,
                        display_labels=modelo.classes_).plot()

# com dados de teste que não foram usados no treinamento
print('métricas mais confiáveis')
acuracia = modelo.score(X_test, y_test)
print("Acurácia nos dados de teste: {:.2f}%".format(acuracia * 100))
y_pred2 = modelo.predict(X_test)
print('_'*60)
print('classes: ', modelo.classes_)
cm = confusion_matrix(y_test, y_pred2, labels=modelo.classes_)
print(cm)
print(classification_report(y_test, y_pred2))
ConfusionMatrixDisplay(confusion_matrix=cm,
                        display_labels=modelo.classes_).plot()
treinaModelo(dados)
Acurácia nos dados de treinamento: 84.06%

```

```

classes: ['espanhol' 'inglês' 'português']
[[19 0 3]
 [ 2 20 1]
 [ 5 0 19]]
precision recall f1-score support

```

```

espanhol    0.73    0.86    0.79    22
inglês      1.00    0.87    0.93    23
português   0.83    0.79    0.81    24

accuracy                0.84    69
macro avg    0.85    0.84    0.84    69
weighted avg 0.85    0.84    0.84    69

```

```

métricas mais confiáveis
Acurácia nos dados de teste: 73.91%

```

```

classes: ['espanhol' 'inglês' 'português']
[[5 0 3]
 [0 7 0]
 [2 1 5]]
precision recall f1-score support

```

```

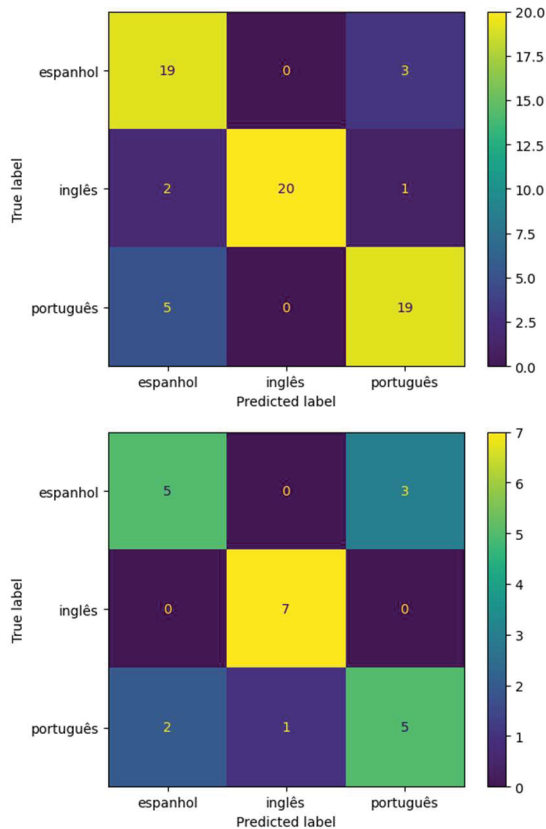
espanhol    0.71    0.62    0.67    8
inglês      0.88    1.00    0.93    7
português   0.62    0.62    0.62    8

```

```

accuracy          0.74    23
macro avg    0.74    0.75    0.74    23
weighted avg    0.73    0.74    0.73    23

```



2) Melhore uma base de dados ruim

Problema: Pegar um dataset que está com uma classificação ruim e fazer pré-processamento para melhorá-lo.

Objetivo: 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 DF correspondente)

- <https://archive.ics.uci.edu/dataset/2/adult>
- <https://archive.ics.uci.edu/dataset/222/bank+marketing>

```

!pip install ucimlrepo matplotlib
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#!pip install ucimlrepo
from ucimlrepo import fetch_ucirepo
import sklearn

```

```

import numpy as np
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, OrdinalEncoder, MinMaxScaler,
StandardScaler, MaxAbsScaler
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import warnings

```

```

warnings.filterwarnings("ignore")
# lendo a base de dados da UCI
# fetch dataset
adult = fetch_ucirepo(id=2)

```

```

# data (as pandas dataframes)
X = adult.data.features
y = adult.data.targets
# juntando a base de dados
df = pd.concat([X,y], axis = 1)

```

```

del X
del y
df.head(3)

```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handle-cleaning	Not-in-family	White	Male	0	0	40	United-States	<=50K

```

for colunas in df.columns:
    print(colunas)
    print()
    print(df[colunas].sort_values().unique())
    print()
age

```

```

[17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88
 89 90]

```

```
workclass
```

['?' 'Federal-gov' 'Local-gov' 'Never-worked' 'Private' 'Self-emp-inc'
'Self-emp-not-inc' 'State-gov' 'Without-pay' nan]

fnlwgt

[12285 13492 13769 ... 1455435 1484705 1490400]

education

['10th' '11th' '12th' '1st-4th' '5th-6th' '7th-8th' '9th' 'Assoc-acdm'
'Assoc-voc' 'Bachelors' 'Doctorate' 'HS-grad' 'Masters' 'Preschool'
'Prof-school' 'Some-college']

education-num

[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]

marital-status

['Divorced' 'Married-AF-spouse' 'Married-civ-spouse'
'Married-spouse-absent' 'Never-married' 'Separated' 'Widowed']

occupation

['?' 'Adm-clerical' 'Armed-Forces' 'Craft-repair' 'Exec-managerial'
'Farming-fishing' 'Handlers-cleaners' 'Machine-op-inspct' 'Other-service'
'Priv-house-serv' 'Prof-specialty' 'Protective-serv' 'Sales'
'Tech-support' 'Transport-moving' nan]

relationship

['Husband' 'Not-in-family' 'Other-relative' 'Own-child' 'Unmarried' 'Wife']

race

['Amer-Indian-Eskimo' 'Asian-Pac-Islander' 'Black' 'Other' 'White']

sex

['Female' 'Male']

capital-gain

[0 114 401 594 914 991 1055 1086 1111 1151 1173 1264
1409 1424 1455 1471 1506 1639 1731 1797 1831 1848 2009 2036
2050 2062 2105 2174 2176 2202 2228 2290 2329 2346 2354 2387
2407 2414 2463 2538 2580 2597 2635 2653 2829 2885 2907 2936
2961 2964 2977 2993 3103 3137 3273 3325 3411 3418 3432 3456
3464 3471 3674 3781 3818 3887 3908 3942 4064 4101 4386 4416
4508 4650 4687 4787 4865 4931 4934 5013 5060 5178 5455 5556
5721 6097 6360 6418 6497 6514 6612 6723 6767 6849 7262 7298
7430 7443 7688 7896 7978 8614 9386 9562 10520 10566 10605 11678
13550 14084 14344 15020 15024 15831 18481 20051 22040 25124 25236 27828
34095 41310 99999]

capital-loss

[0 155 213 323 419 625 653 810 880 974 1092 1138 1258 1340
1380 1408 1411 1421 1429 1485 1504 1510 1539 1564 1573 1579 1590 1594

```
1602 1617 1628 1648 1651 1668 1669 1672 1719 1721 1726 1735 1740 1741
1755 1762 1816 1825 1844 1848 1870 1876 1887 1902 1911 1944 1974 1977
1980 2001 2002 2042 2051 2057 2080 2129 2149 2163 2174 2179 2201 2205
2206 2231 2238 2246 2258 2267 2282 2339 2352 2377 2392 2415 2444 2457
2465 2467 2472 2489 2547 2559 2603 2754 2824 3004 3175 3683 3770 3900
4356]
```

hours-per-week

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 72 73
74 75 76 77 78 79 80 81 82 84 85 86 87 88 89 90 91 92 94 95 96 97 98 99]
```

native-country

```
['?' 'Cambodia' 'Canada' 'China' 'Columbia' 'Cuba' 'Dominican-Republic'
'Ecuador' 'El-Salvador' 'England' 'France' 'Germany' 'Greece' 'Guatemala'
'Haiti' 'Holand-Netherlands' 'Honduras' 'Hong' 'Hungary' 'India' 'Iran'
'Ireland' 'Italy' 'Jamaica' 'Japan' 'Laos' 'Mexico' 'Nicaragua'
'Outlying-US(Guam-USVI-etc)' 'Peru' 'Philippines' 'Poland' 'Portugal'
'Puerto-Rico' 'Scotland' 'South' 'Taiwan' 'Thailand' 'Trinidad&Tobago'
'United-States' 'Vietnam' 'Yugoslavia' nan]
```

income

```
['<=50K' '<=50K.' '>50K' '>50K.']
```

```
def train(X_train, y_train):
    treinador = svm.SVC( random_state=24 )
    modelo = treinador.fit(X_train, y_train)
    return modelo

def split(X,Y):
    X_train, X_test, y_train, y_test = train_test_split(X,Y,
                                                         test_size=0.25,
                                                         stratify=Y,
                                                         random_state=24)
    return X_train, X_test, y_train, y_test

def print_metricas2(modelo, X_train, X_test, y_train, y_test):
    # score com os dados de treinamento
    acuracia = modelo.score(X_train, y_train)
    print("Acurácia nos dados de treinamento: {:.2f}%".format(acuracia * 100))
    # melhor avaliar com a matriz de confusão
    y_pred = modelo.predict(X_train)
    print('_'*60)
    print('classes: ', modelo.classes_)
    cm = confusion_matrix(y_train, y_pred, labels=modelo.classes_)
    print(cm)
    print(classification_report(y_train, y_pred))

    # com dados de teste que não foram usados no treinamento
    print('métricas mais confiáveis')
    acuracia = modelo.score(X_test, y_test)
    print("Acurácia nos dados de teste: {:.2f}%".format(acuracia * 100))
    y_pred2 = modelo.predict(X_test)
    print('_'*60)
    print('classes: ', modelo.classes_)
    cm = confusion_matrix(y_test, y_pred2, labels=modelo.classes_)
```

```

print(cm)
print(classification_report(y_test, y_pred2))
def get_acuracia(cm):
    total = sum(sum(cm)) # Soma de todos os elementos da matriz
    acertos = sum(np.diag(cm)) # Soma dos elementos da diagonal principal (verdadeiros positivos e
    verdadeiros negativos)
    acuracia = acertos / total
    return acuracia
def get_metricas(modelo, X_train, X_test, y_train, y_test):
    # Obtendo métricas sobre o conjunto de treinamento
    y_pred = modelo.predict(X_train)
    cm_train = confusion_matrix(y_train, y_pred, labels=modelo.classes_)
    cr_train = classification_report(y_train, y_pred)
    acuracia_train = get_acuracia(cm_train)
    # Obtendo métricas sobre o conjunto de teste
    y_pred2 = modelo.predict(X_test)
    cm_test = confusion_matrix(y_test, y_pred2, labels=modelo.classes_)
    cr_test = classification_report(y_test, y_pred2)
    acuracia_test = get_acuracia(cm_test)
    return {'modelo': modelo,
            'acuracia_train': acuracia_train,
            'acuracia_test': acuracia_test,
            'cm_train': cm_train,
            'cm_test': cm_test,
            'cr_train': cr_train,
            'cr_test': cr_test
            }

def print_metricas(metricas):
    #metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)
    print('**** TREINAMENTO ****')
    print('Acurácia nos dados de treinamento: {:.2f}%'.format(metricas['acuracia_train'] * 100))
    print('Confusion Matrix nos dados de treinamento')
    print(metricas['cm_train'])
    print('Classification Report nos dados de treinamento')
    print(metricas['cr_train'])
    print('**** TESTE ****')
    print('Acurácia nos dados de teste: {:.2f}%'.format(metricas['acuracia_test'] * 100))
    print('Confusion Matrix nos dados de teste')
    print(metricas['cm_test'])
    print('Classification Report nos dados de teste')
    print(metricas['cr_test'])
    print('**** RESUMO ****')
    print('Acurácia nos dados de treinamento: {:.2f}%'.format(metricas['acuracia_train'] * 100))
    print('Acurácia nos dados de teste: {:.2f}%'.format(metricas['acuracia_test'] * 100))
    #Separa df em X e Y
def preprocess(df, fillna=True):
    #nulls (inicialmente colocar -1 no lugar)
    if fillna:
        df = df.fillna('?')
    target = 'income'
    X = df.drop(target, axis=1)
    Y = df[target]
    return X, Y
def get_features_categoricas(X):
    features_categoricas = [coluna for coluna in X.select_dtypes(include=['object']).columns ]
    return features_categoricas
#Aplica LabelEncoder nas features categóricas
def ordinal_encoder(X):

```

```

enc = OrdinalEncoder()
features_categoricas = get_features_categoricas(X)
df_encoded = pd.DataFrame(enc.fit_transform(X[features_categoricas]),
columns=enc.get_feature_names_out(features_categoricas))
df_final = pd.concat([X.drop(columns=features_categoricas), df_encoded], axis=1)
return df_final

enc.fit(X)
return enc.transform(X)
def ordinal_encoder_completo(X):
enc = OrdinalEncoder()
enc.fit(X)
return enc.transform(X)

def one_hot_encoder_completo(X):
enc = OneHotEncoder()
enc.fit(X)
return enc.transform(X)

def one_hot_encoder(X):
features_categoricas = get_features_categoricas(X)
enc = OneHotEncoder(sparse_output=False)
df_encoded = pd.DataFrame(enc.fit_transform(X[features_categoricas]),
columns=enc.get_feature_names_out(features_categoricas))
df_final = pd.concat([X.drop(columns=features_categoricas), df_encoded], axis=1)
return df_final
df.isnull().sum()
age      0
workclass    963
fnlwgt      0
education    0
education-num    0
marital-status    0
occupation    966
relationship    0
race          0
sex           0
capital-gain    0
capital-loss    0
hours-per-week    0
native-country    274
income         0
dtype: int64
colunas_com_nulls = list(df.columns[df.isnull().sum()>0]) # pega quem nao esta nulo
colunas_com_nulls
['workclass', 'occupation', 'native-country']
# Verificando quantidade de valores únicos e tipos de cada coluna
[[c, df[c].nunique(), df[c].dtype] for c in df.columns]
[['age', 74, dtype('int64')],
 ['workclass', 9, dtype('O')],
 ['fnlwgt', 28523, dtype('int64')],
 ['education', 16, dtype('O')],
 ['education-num', 16, dtype('int64')],
 ['marital-status', 7, dtype('O')],
 ['occupation', 15, dtype('O')],
 ['relationship', 6, dtype('O')],
 ['race', 5, dtype('O')],
 ['sex', 2, dtype('O')],
 ['capital-gain', 123, dtype('int64')],
 ['capital-loss', 99, dtype('int64')],
 ['hours-per-week', 96, dtype('int64')],
 ['native-country', 42, dtype('O')],
 ['income', 4, dtype('O')]]

```

```

# Inicializado Dataframe que armazenará os resultados dos experimentos
df_resultados_geral = pd.DataFrame(columns=['descricao', 'modelo', 'acuracia_train', 'acuracia_test',
'cm_train', 'cm_test', 'cr_train', 'cr_test'])
X, Y = preprocess(df, fillna=False)
X = ordinal_encoder(X)
# o que estiver nulo, preenche com a media
for c in colunas_com_nulls:
    print(c)
    X[c] = X[[c]].fillna(X[c].mean())
workclass
occupation
native-country

```

1 Inicial - Somente OrdinalEncoder em todos campos

```
descricao = '1 Inicial - Somente OrdinalEncoder em todos campos'
```

```

X, Y = preprocess(df)
X = ordinal_encoder_completo(X)

```

```

X_train, X_test, y_train, y_test = split(X, Y)
modelo = train(X_train, y_train)
metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)

```

```

metricas['descricao'] = descricao
df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)

```

```
print_metricas(metricas)
```

```
df_resultados_geral
```

```
**** TREINAMENTO ****
```

```
Acurácia nos dados de treinamento: 50.61%
```

```
Confusion Matrix nos dados de treinamento
```

```
[[18540  0  0  0]
```

```
 [ 9326  0  0  0]
```

```
 [ 5881  0  0  0]
```

```
 [ 2884  0  0  0]]
```

```
Classification Report nos dados de treinamento
```

```
precision recall f1-score support
```

```
<=50K    0.51    1.00    0.67   18540
```

```
<=50K.    0.00    0.00    0.00    9326
```

```
>50K     0.00    0.00    0.00    5881
```

```
>50K.    0.00    0.00    0.00    2884
```

```
accuracy                0.51   36631
```

```
macro avg    0.13    0.25    0.17   36631
```

```
weighted avg    0.26    0.51    0.34   36631
```

```
**** TESTE ****
```

```
Acurácia nos dados de teste: 50.61%
```

```
Confusion Matrix nos dados de teste
```

```
[[6180  0  0  0]
```

```
 [3109  0  0  0]
```

```
 [1960  0  0  0]
```

```
 [ 962  0  0  0]]
```

```
Classification Report nos dados de teste
```

```
precision recall f1-score support
```



```

<=50K    0.51    1.00    0.67    6180
<=50K.    0.00    0.00    0.00    3109
>50K     0.00    0.00    0.00    1960
>50K.    0.00    0.00    0.00    962

accuracy          0.51    12211
macro avg    0.13    0.25    0.17    12211
weighted avg    0.26    0.51    0.34    12211

```

**** RESUMO ****

Acurácia nos dados de treinamento: 50.61%

Acurácia nos dados de teste: 50.61%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0,]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]	precision recall f1- score ...	precision recall f1- score ...

1.1 Inicial - Somente OrdinalEncoder nos campos categóricos

descricao = '1.1 Inicial - Somente OrdinalEncoder nos campos categóricos'

X, Y = preprocess(df)

X = ordinal_encoder(X)

X_train, X_test, y_train, y_test = split(X, Y)

modelo = train(X_train, y_train)

metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)

metricas['descricao'] = descricao

df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)

print_metricas(metricas)

df_resultados_geral

**** TREINAMENTO ****

Acurácia nos dados de treinamento: 52.97%

Confusion Matrix nos dados de treinamento

```
[[18522  0  18  0]
```

```
[ 9316  0  10  0]
```

```
[ 5000  0 881  0]
```

```
[ 2465  0 419  0]]
```

Classification Report nos dados de treinamento

precision recall f1-score support

```

<=50K    0.52    1.00    0.69    18540
<=50K.    0.00    0.00    0.00    9326
>50K     0.66    0.15    0.24    5881
>50K.    0.00    0.00    0.00    2884

```

```

accuracy          0.53    36631
macro avg    0.30    0.29    0.23    36631
weighted avg    0.37    0.53    0.39    36631

```

**** TESTE ****

Acurácia nos dados de teste: 52.96%

Confusion Matrix nos dados de teste

```
[[6173  0  7  0]
 [3108  0  1  0]
 [1666  0 294  0]
 [ 828  0 134  0]]
```

Classification Report nos dados de teste

```
precision recall f1-score support

<=50K    0.52    1.00    0.69    6180
<=50K.    0.00    0.00    0.00    3109
>50K     0.67    0.15    0.25    1960
>50K.    0.00    0.00    0.00     962

accuracy                0.53    12211
macro avg              0.30    0.29    0.23    12211
weighted avg           0.37    0.53    0.39    12211
```

**** RESUMO ****

Acurácia nos dados de treinamento: 52.97%

Acurácia nos dados de teste: 52.96%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]]	precision recall f1- score ...	precision recall f1- score ...
1	1.1 Inicial - Somente OrdinalEncoder nos campo...	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0], [5000, 0...]]	[[6173, 0, 7, 0], [3108, 0, 1, 0], [1666, 0, 2...]]	precision recall f1- score ...	precision recall f1- score ...

```
df[['income']].value_counts()
```

```
income
<=50K    24720
<=50K.   12435
>50K     7841
>50K.    3846
```

Name: count, dtype: int64

ajeitando a variavel resposta

```
df['income'].str.replace(".", "", regex=False).value_counts().index
```

```
df['income'] = df['income'].astype(str).str.replace('.', "", regex=False).str.replace(r'\s+', "", regex=True)
```

```
df[['income']].value_counts() # ok
```

```
income
<=50K    37155
>50K     11687
```

Name: count, dtype: int64

2. Modelo com ajustes na target que estava com problemas (4 categorias)

Agora vamos gerar novamente X e y, separar os conjuntos de treinamento e teste, treinar o modelo e verificar as métricas:

```
descricao = '2. Modelo com ajustes na target que estava com problemas (4 categorias)'
```

```

X, Y = preprocess(df)
X = ordinal_encoder(X)

X_train, X_test, y_train, y_test = split(X,Y)
modelo = train(X_train, y_train)
metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)

metricas['descricao'] = descricao
df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)

print_metricas(metricas)

```

```

df_resultados_geral
**** TREINAMENTO ****
Acurácia nos dados de treinamento: 79.72%
Confusion Matrix nos dados de treinamento
[[27806  60]
 [ 7370 1395]]
Classification Report nos dados de treinamento
      precision    recall  f1-score   support

    <=50K      0.79      1.00      0.88     27866
    >50K      0.96      0.16      0.27      8765

 accuracy                0.80     36631
 macro avg      0.87      0.58      0.58     36631
weighted avg      0.83      0.80      0.74     36631

```

```

**** TESTE ****
Acurácia nos dados de teste: 80.12%
Confusion Matrix nos dados de teste
[[9271  18]
 [2410 512]]
Classification Report nos dados de teste
      precision    recall  f1-score   support

    <=50K      0.79      1.00      0.88      9289
    >50K      0.97      0.18      0.30      2922

 accuracy                0.80     12211
 macro avg      0.88      0.59      0.59     12211
weighted avg      0.83      0.80      0.74     12211

```

**** RESUMO ****

Acurácia nos dados de treinamento: 79.72%
Acurácia nos dados de teste: 80.12%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]]	precision recall f1- score ...	precision recall f1- score ...

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
1	1.1 Inicial - Somente OrdinalEncoder nos campo...	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0], [5000, 0...	[[6173, 0, 7, 0], [3108, 0, 1, 0], [1666, 0, 2...	precision recall f1- score ...	precision recall f1- score ...
2	2. Modelo com ajustes na target que estava com...	SVC(random_state=24)	0.797166	0.801163	[[27806, 60], [7370, 1395]]	[[9271, 18], [2410, 512]]	precision recall f1- score ...	precision recall f1- score ...

2.1. one-hot-encode completo no lugar de labelencoder

Vamos ver o efeito da aplicação de one-hot-encode no lugar de labelencoder

descricao = '2.1. one-hot-encode completo no lugar de labelencoder'

```
X, Y = preprocess(df)
```

```
X = one_hot_encoder_completo(X)
```

```
X_train, X_test, y_train, y_test = split(X, Y)
```

```
modelo = train(X_train, y_train)
```

```
metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)
```

```
metricas['descricao'] = descricao
```

```
df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)
```

```
print_metricas(metricas)
```

```
df_resultados_geral
```

```
**** TREINAMENTO ****
```

```
Acurácia nos dados de treinamento: 88.76%
```

```
Confusion Matrix nos dados de treinamento
```

```
[[26865 1001]
```

```
 [ 3116 5649]]
```

```
Classification Report nos dados de treinamento
```

```
precision recall f1-score support
```

```
<=50K    0.90    0.96    0.93   27866
```

```
>50K     0.85    0.64    0.73   8765
```

```
accuracy                0.89   36631
```

```
macro avg    0.87    0.80    0.83   36631
```

```
weighted avg    0.88    0.89    0.88   36631
```

```
**** TESTE ****
```

```
Acurácia nos dados de teste: 86.71%
```

```
Confusion Matrix nos dados de teste
```

```
[[8837 452]
```

```
 [1171 1751]]
```

```
Classification Report nos dados de teste
```

```
precision recall f1-score support
```

```
<=50K    0.88    0.95    0.92   9289
```

```
>50K     0.79    0.60    0.68   2922
```

```

accuracy          0.87  12211
macro avg    0.84  0.78  0.80  12211
weighted avg    0.86  0.87  0.86  12211

```

**** RESUMO ****

Acurácia nos dados de treinamento: 88.76%

Acurácia nos dados de teste: 86.71%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]]	precision recall f1- score ...	precision recall f1- score ...
1	1.1 Inicial - Somente OrdinalEncoder nos campo...	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0], [5000, 0...]]	[[6173, 0, 7, 0], [3108, 0, 1, 0], [1666, 0, 2...]]	precision recall f1- score ...	precision recall f1- score ...
2	2. Modelo com ajustes na target que estava com...	SVC(random_state=24)	0.797166	0.801163	[[27806, 60], [7370, 1395]]	[[9271, 18], [2410, 512]]	precision recall f1- score ...	precision recall f1- score ...
3	2.1. one-hot- encode completo no lugar de label...	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...

2.1.1 one-hot-encode completo com MaxAbsScaler

descricao = '2.1.1 one-hot-encode completo com MaxAbsScaler'

```
X, Y = preprocess(df)
```

```
X = one_hot_encoder_completo(X)
```

```
X_train, X_test, y_train, y_test = split(X, Y)
```

```
scaler = MaxAbsScaler()
```

```
X_train = scaler.fit_transform(X_train)
```

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

```
modelo = train(X_train, y_train)
```

```
metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)
```

```
metricas['descricao'] = descricao
```

```
df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)
```

```
print_metricas(metricas)
```

```
df_resultados_geral
```

```
**** TREINAMENTO ****
```

Acurácia nos dados de treinamento: 88.76%

Confusion Matrix nos dados de treinamento

[[26865 1001]

[3116 5649]]

Classification Report nos dados de treinamento

precision recall f1-score support

<=50K 0.90 0.96 0.93 27866

>50K 0.85 0.64 0.73 8765

accuracy 0.89 36631

macro avg 0.87 0.80 0.83 36631

weighted avg 0.88 0.89 0.88 36631

**** TESTE ****

Acurácia nos dados de teste: 86.71%

Confusion Matrix nos dados de teste

[[8837 452]

[1171 1751]]

Classification Report nos dados de teste

precision recall f1-score support

<=50K 0.88 0.95 0.92 9289

>50K 0.79 0.60 0.68 2922

accuracy 0.87 12211

macro avg 0.84 0.78 0.80 12211

weighted avg 0.86 0.87 0.86 12211

**** RESUMO ****

Acurácia nos dados de treinamento: 88.76%

Acurácia nos dados de teste: 86.71%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]]	precision recall f1- score ...	precision recall f1- score ...
1	1.1 Inicial - Somente OrdinalEncoder nos campo...	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0], [5000, 0...]]	[[6173, 0, 7, 0], [3108, 0, 1, 0], [1666, 0, 2...]]	precision recall f1- score ...	precision recall f1- score ...
2	2. Modelo com ajustes na target que estava com...	SVC(random_state=24)	0.797166	0.801163	[[27806, 60], [7370, 1395]]	[[9271, 18], [2410, 512]]	precision recall f1- score ...	precision recall f1- score ...
3	2.1. one-hot- encode completo no lugar de label...	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
4	2.1.1 one-hot-encode completo com MaxAbsScaler	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...

2.2 MinMaxScaler sobre ordinalEncoder

#Aplicando MinMaxScaler sobre ordinalEncoder

descricao = '2.2 MinMaxScaler sobre ordinalEncoder'

```
X, Y = preprocess(df)
X = ordinal_encoder(X)
```

```
X_train, X_test, y_train, y_test = split(X, Y)
```

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
modelo = train(X_train, y_train)
metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)
```

```
metricas['descricao'] = descricao
df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)
```

```
print_metricas(metricas)
```

```
df_resultados_geral
**** TREINAMENTO ****
Acurácia nos dados de treinamento: 84.77%
Confusion Matrix nos dados de treinamento
[[26302 1564]
 [4016 4749]]
Classification Report nos dados de treinamento
      precision    recall  f1-score   support

<=50K      0.87      0.94      0.90     27866
>50K       0.75      0.54      0.63      8765

accuracy            0.85     36631
macro avg      0.81      0.74      0.77     36631
weighted avg   0.84      0.85      0.84     36631
```

```
**** TESTE ****
Acurácia nos dados de teste: 85.04%
Confusion Matrix nos dados de teste
[[8767 522]
 [1305 1617]]
Classification Report nos dados de teste
      precision    recall  f1-score   support

<=50K      0.87      0.94      0.91      9289
>50K       0.76      0.55      0.64      2922
```

```

accuracy          0.85  12211
macro avg    0.81  0.75  0.77  12211
weighted avg    0.84  0.85  0.84  12211

```

**** RESUMO ****

Acurácia nos dados de treinamento: 84.77%

Acurácia nos dados de teste: 85.04%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]]	precision recall f1- score ...	precision recall f1- score ...
1	1.1 Inicial - Somente OrdinalEncoder nos campo...	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0], [5000, 0...]]	[[6173, 0, 7, 0], [3108, 0, 1, 0], [1666, 0, 2...]]	precision recall f1- score ...	precision recall f1- score ...
2	2. Modelo com ajustes na target que estava com...	SVC(random_state=24)	0.797166	0.801163	[[27806, 60], [7370, 1395]]	[[9271, 18], [2410, 512]]	precision recall f1- score ...	precision recall f1- score ...
3	2.1. one-hot- encode completo no lugar de label...	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...
4	2.1.1 one-hot- encode completo com MaxAbsScaler	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...
5	2.2 MinMaxScaler sobre ordinalEncoder	SVC(random_state=24)	0.847670	0.850381	[[26302, 1564], [4016, 4749]]	[[8767, 522], [1305, 1617]]	precision recall f1- score ...	precision recall f1- score ...

2.2.1 MinMaxScaler sobre ordinalEncoder - Dropando colunas

#Vamos verificar o efeito de dropar as colunas com nulos.

#Aplicando MinMaxScaler sobre ordinalEncoder

descricao = '2.2.1 MinMaxScaler sobre ordinalEncoder - Dropando colunas'

```

X, Y = preprocess(df)
X = X.drop(colunas_com_nulos,axis=1)
X = ordinal_encoder(X)

```

```
X_train, X_test, y_train, y_test = split(X,Y)
```

```

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)

```



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

```
modelo = train(X_train, y_train)
```

```
metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)
```

```
metricas['descricao'] = descricao
```

```
df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)
```

```
print_metricas(metricas)
```

```
df_resultados_geral
```

```
**** TREINAMENTO ****
```

```
Acurácia nos dados de treinamento: 84.75%
```

```
Confusion Matrix nos dados de treinamento
```

```
[[26162 1704]
```

```
 [3884 4881]]
```

```
Classification Report nos dados de treinamento
```

```
precision recall f1-score support
```

```
<=50K    0.87    0.94    0.90   27866
```

```
>50K     0.74    0.56    0.64    8765
```

```
accuracy                0.85   36631
```

```
macro avg    0.81    0.75    0.77   36631
```

```
weighted avg    0.84    0.85    0.84   36631
```

```
**** TESTE ****
```

```
Acurácia nos dados de teste: 84.96%
```

```
Confusion Matrix nos dados de teste
```

```
[[8715 574]
```

```
 [1263 1659]]
```

```
Classification Report nos dados de teste
```

```
precision recall f1-score support
```

```
<=50K    0.87    0.94    0.90   9289
```

```
>50K     0.74    0.57    0.64   2922
```

```
accuracy                0.85   12211
```

```
macro avg    0.81    0.75    0.77   12211
```

```
weighted avg    0.84    0.85    0.84   12211
```

```
**** RESUMO ****
```

```
Acurácia nos dados de treinamento: 84.75%
```

```
Acurácia nos dados de teste: 84.96%
```

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]	precision recall f1- score ...	precision recall f1- score ...
1	1.1 Inicial - Somente	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0],	[[6173, 0, 7, 0], [3108, 0, 1, 0],	precision recall f1- score ...	precision recall f1- score ...

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
	OrdinalEncoder nos campo...				[5000, 0...	[1666, 0, 2...		
2	2. Modelo com ajustes na target que estava com...	SVC(random_state=24)	0.797166	0.801163	[[27806, 60], [7370, 1395]]	[[9271, 18], [2410, 512]]	precision recall f1- score ...	precision recall f1- score ...
3	2.1. one-hot- encode completo no lugar de label...	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...
4	2.1.1 one-hot- encode completo com MaxAbsScaler	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...
5	2.2 MinMaxScaler sobre ordinalEncoder	SVC(random_state=24)	0.847670	0.850381	[[26302, 1564], [4016, 4749]]	[[8767, 522], [1305, 1617]]	precision recall f1- score ...	precision recall f1- score ...
6	2.2.1 MinMaxScaler sobre ordinalEncoder - Dro...	SVC(random_state=24)	0.847452	0.849562	[[26162, 1704], [3884, 4881]]	[[8715, 574], [1263, 1659]]	precision recall f1- score ...	precision recall f1- score ...

2.3 one-hot-encode somente em categóricas

Vamos ver o efeito da aplicação de one-hot-encode no lugar de labelencoder

descricao = '2.3 one-hot-encode somente em categoricas'

X, Y = preprocess(df)

X = one_hot_encoder(X)

X_train, X_test, y_train, y_test = split(X, Y)

modelo = train(X_train, y_train)

metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)

metricas['descricao'] = descricao

df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)

print_metricas(metricas)

df_resultados_geral

**** TREINAMENTO ****

Acurácia nos dados de treinamento: 79.72%

Confusion Matrix nos dados de treinamento

[[27811 55]

[7375 1390]]

Classification Report nos dados de treinamento

```

precision recall f1-score support

<=50K    0.79    1.00    0.88   27866
>50K     0.96    0.16    0.27    8765

accuracy                0.80   36631
macro avg    0.88    0.58    0.58   36631
weighted avg    0.83    0.80    0.74   36631

```

**** TESTE ****

Acurácia nos dados de teste: 80.08%

Confusion Matrix nos dados de teste

```
[[9271 18]
```

```
[2415 507]]
```

Classification Report nos dados de teste

```
precision recall f1-score support
```

```

<=50K    0.79    1.00    0.88   9289
>50K     0.97    0.17    0.29   2922

```

```

accuracy                0.80   12211
macro avg    0.88    0.59    0.59   12211
weighted avg    0.83    0.80    0.74   12211

```

**** RESUMO ****

Acurácia nos dados de treinamento: 79.72%

Acurácia nos dados de teste: 80.08%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]]	precision recall f1- score ...	precision recall f1- score ...
1	1.1 Inicial - Somente OrdinalEncoder nos campo...	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0], [5000, 0...]]	[[6173, 0, 7, 0], [3108, 0, 1, 0], [1666, 0, 2...]]	precision recall f1- score ...	precision recall f1- score ...
2	2. Modelo com ajustes na target que estava com...	SVC(random_state=24)	0.797166	0.801163	[[27806, 60], [7370, 1395]]	[[9271, 18], [2410, 512]]	precision recall f1- score ...	precision recall f1- score ...
3	2.1. one-hot- encode completo no lugar de label...	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...
4	2.1.1 one-hot- encode completo com MaxAbsScaler	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
5	2.2 MinMaxScaler sobre ordinalEncoder	SVC(random_state=24)	0.847670	0.850381	[[26302, 1564], [4016, 4749]]	[[8767, 522], [1305, 1617]]	precision recall f1- score ...	precision recall f1- score ...
6	2.2.1 MinMaxScaler sobre ordinalEncoder - Dro...	SVC(random_state=24)	0.847452	0.849562	[[26162, 1704], [3884, 4881]]	[[8715, 574], [1263, 1659]]	precision recall f1- score ...	precision recall f1- score ...
7	2.3 one-hot- encode somente em categoricas	SVC(random_state=24)	0.797166	0.800753	[[27811, 55], [7375, 1390]]	[[9271, 18], [2415, 507]]	precision recall f1- score ...	precision recall f1- score ...

Aplicando MinMaxScaler sobre one_hot_encoder

#Aplicando MinMaxScaler sobre one_hot_encoder

descricao = '2.3.1 one-hot-encode somente em categoricas - com minmaxscaler'

```
X, Y = preprocess(df)
X = one_hot_encoder(X)
```

```
X_train, X_test, y_train, y_test = split(X, Y)
```

```
scaler = MinMaxScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
modelo = train(X_train, y_train)
metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)
```

```
metricas['descricao'] = descricao
df_resultados_geral = pd.concat([df_resultados_geral, pd.DataFrame([metricas])], ignore_index=True)
```

```
print_metricas(metricas)
```

```
df_resultados_geral
```

```
**** TREINAMENTO ****
```

```
Acurácia nos dados de treinamento: 84.63%
```

```
Confusion Matrix nos dados de treinamento
```

```
[[26105 1761]
```

```
 [3870 4895]]
```

```
Classification Report nos dados de treinamento
```

```
precision recall f1-score support
```

```
<=50K    0.87    0.94    0.90   27866
```

```
>50K     0.74    0.56    0.63   8765
```

```
accuracy                0.85   36631
```

```
macro avg    0.80    0.75    0.77   36631
```

```
weighted avg    0.84    0.85    0.84   36631
```

**** TESTE ****

Acurácia nos dados de teste: 84.15%

Confusion Matrix nos dados de teste

[[8650 639]

[1297 1625]]

Classification Report nos dados de teste

precision recall f1-score support

<=50K 0.87 0.93 0.90 9289

>50K 0.72 0.56 0.63 2922

accuracy 0.84 12211

macro avg 0.79 0.74 0.76 12211

weighted avg 0.83 0.84 0.83 12211

**** RESUMO ****

Acurácia nos dados de treinamento: 84.63%

Acurácia nos dados de teste: 84.15%

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
0	1 Inicial - Somente OrdinalEncoder em todos ca...	SVC(random_state=24)	0.506129	0.506101	[[18540, 0, 0, 0], [9326, 0, 0, 0], [5881, 0, ...]]	[[6180, 0, 0, 0], [3109, 0, 0, 0], [1960, 0, 0...]]	precision recall f1- score ...	precision recall f1- score ...
1	1.1 Inicial - Somente OrdinalEncoder nos campo...	SVC(random_state=24)	0.529688	0.529604	[[18522, 0, 18, 0], [9316, 0, 10, 0], [5000, 0...]]	[[6173, 0, 7, 0], [3108, 0, 1, 0], [1666, 0, 2...]]	precision recall f1- score ...	precision recall f1- score ...
2	2. Modelo com ajustes na target que estava com...	SVC(random_state=24)	0.797166	0.801163	[[27806, 60], [7370, 1395]]	[[9271, 18], [2410, 512]]	precision recall f1- score ...	precision recall f1- score ...
3	2.1. one-hot- encode completo no lugar de label...	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...
4	2.1.1 one-hot- encode completo com MaxAbsScaler	SVC(random_state=24)	0.887609	0.867087	[[26865, 1001], [3116, 5649]]	[[8837, 452], [1171, 1751]]	precision recall f1- score ...	precision recall f1- score ...
5	2.2 MinMaxScaler sobre ordinalEncoder	SVC(random_state=24)	0.847670	0.850381	[[26302, 1564], [4016, 4749]]	[[8767, 522], [1305, 1617]]	precision recall f1- score ...	precision recall f1- score ...

	descricao	modelo	acuracia_train	acuracia_test	cm_train	cm_test	cr_train	cr_test
6	2.2.1 MinMaxScaler sobre ordinalEncoder - Dro...	SVC(random_state=24)	0.847452	0.849562	[[26162, 1704], [3884, 4881]]	[[8715, 574], [1263, 1659]]	precision recall f1- score ...	precision recall f1- score ...
7	2.3 one-hot- encode somente em categoricas	SVC(random_state=24)	0.797166	0.800753	[[27811, 55], [7375, 1390]]	[[9271, 18], [2415, 507]]	precision recall f1- score ...	precision recall f1- score ...
8	2.3.1 one-hot- encode somente em categoricas - ...	SVC(random_state=24)	0.846278	0.841454	[[26105, 1761], [3870, 4895]]	[[8650, 639], [1297, 1625]]	precision recall f1- score ...	precision recall f1- score ...

Final - Avaliando os modelos

Print the DataFrame with full descriptions

pd.set_option('display.max_colwidth', None)

df_resultados_geral[['descricao', 'acuracia_train', 'acuracia_test']]

	descricao	acuracia_train	acuracia_test
0	1 Inicial - Somente OrdinalEncoder em todos campos	0.506129	0.506101
1	1.1 Inicial - Somente OrdinalEncoder nos campos categóricos	0.529688	0.529604
2	2. Modelo com ajustes na target que estava com problemas (4 categorias)	0.797166	0.801163
3	2.1. one-hot-encode completo no lugar de labelencoder	0.887609	0.867087
4	2.1.1 one-hot-encode completo com MaxAbsScaler	0.887609	0.867087
5	2.2 MinMaxScaler sobre ordinalEncoder	0.847670	0.850381
6	2.2.1 MinMaxScaler sobre ordinalEncoder - Dropando colunas	0.847452	0.849562
7	2.3 one-hot-encode somente em categoricas	0.797166	0.800753
8	2.3.1 one-hot-encode somente em categoricas - com minmaxscaler	0.846278	0.841454

R: Logo, pode-se ver que conseguimos aumentar bem a acurácia dos dados após os tratamentos das variáveis, chegando a 88% de acurácia nos dados de treino como é observado para o modelo "2.1. one-hot-encode completo no lugar de labelencode"

APÊNDICE 7 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

Seed utilizado: **202485**

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

Especificações:

O trabalho pode ser feito por uma equipe de 1 a 5 integrantes.

Para cada problema, preencher as colunas dos quadros com o que pede. Além disso, fazer as solicitações pedidas antes dos quadros.

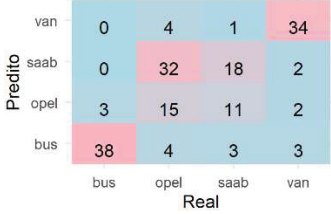
CLASSIFICAÇÃO

Para o experimento de Classificação:

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

Veículo

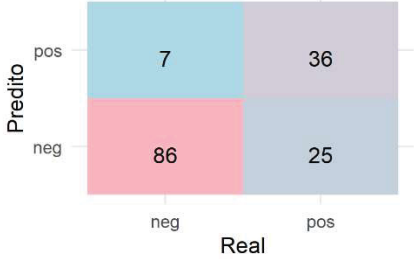
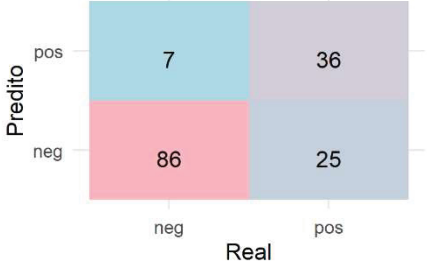
Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM - Hold-out	size=100, sigma=0.01	0,823529412	<div><div><div>Predito</div><div>van</div><div>saab</div><div>opel</div><div>bus</div></div><div><div><div>1</div><div>0</div><div>0</div><div>39</div></div><div><div>0</div><div>24</div><div>30</div><div>0</div></div><div><div>0</div><div>31</div><div>3</div><div>1</div></div><div><div>40</div><div>0</div><div>0</div><div>1</div></div></div><div><div>bus</div><div>opel</div><div>saab</div><div>van</div></div><div>Real</div></div>
SVM - CV	size=100, sigma=0.015	0,817647059	<div><div><div>Predito</div><div>van</div><div>saab</div><div>opel</div><div>bus</div></div><div><div><div>1</div><div>1</div><div>0</div><div>37</div></div><div><div>0</div><div>20</div><div>29</div><div>1</div></div><div><div>0</div><div>33</div><div>4</div><div>2</div></div><div><div>40</div><div>1</div><div>0</div><div>1</div></div></div><div><div>bus</div><div>opel</div><div>saab</div><div>van</div></div><div>Real</div></div>
RNA - CV	size=41, decay=0.7	0,794117647	<div><div><div>Predito</div><div>van</div><div>saab</div><div>opel</div><div>bus</div></div><div><div><div>1</div><div>2</div><div>0</div><div>41</div></div><div><div>0</div><div>22</div><div>24</div><div>0</div></div><div><div>0</div><div>30</div><div>9</div><div>0</div></div><div><div>40</div><div>1</div><div>0</div><div>0</div></div></div><div><div>bus</div><div>opel</div><div>saab</div><div>van</div></div><div>Real</div></div>
RNA - Hold-out	size=41, decay=0.7	0,747058824	<div><div><div>Predito</div><div>van</div><div>saab</div><div>opel</div><div>bus</div></div><div><div><div>1</div><div>2</div><div>0</div><div>40</div></div><div><div>1</div><div>24</div><div>20</div><div>0</div></div><div><div>0</div><div>28</div><div>13</div><div>0</div></div><div><div>39</div><div>1</div><div>0</div><div>1</div></div></div><div><div>bus</div><div>opel</div><div>saab</div><div>van</div></div><div>Real</div></div>
RF - Hold-out	mtry=2	0,729411765	<div><div><div>Predito</div><div>van</div><div>saab</div><div>opel</div><div>bus</div></div><div><div><div>0</div><div>2</div><div>1</div><div>40</div></div><div><div>0</div><div>33</div><div>24</div><div>1</div></div><div><div>0</div><div>19</div><div>8</div><div>0</div></div><div><div>41</div><div>1</div><div>0</div><div>0</div></div></div><div><div>bus</div><div>opel</div><div>saab</div><div>van</div></div><div>Real</div></div>
RF - CV	mtry=5	0,723529412	<div><div><div>Predito</div><div>van</div><div>saab</div><div>opel</div><div>bus</div></div><div><div><div>0</div><div>2</div><div>0</div><div>40</div></div><div><div>0</div><div>34</div><div>23</div><div>1</div></div><div><div>0</div><div>19</div><div>9</div><div>0</div></div><div><div>41</div><div>0</div><div>1</div><div>0</div></div></div><div><div>bus</div><div>opel</div><div>saab</div><div>van</div></div><div>Real</div></div>

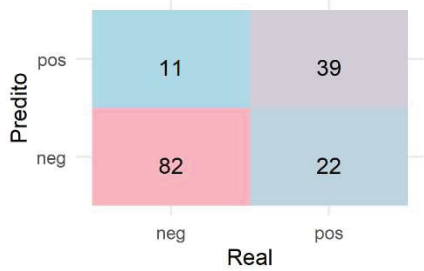
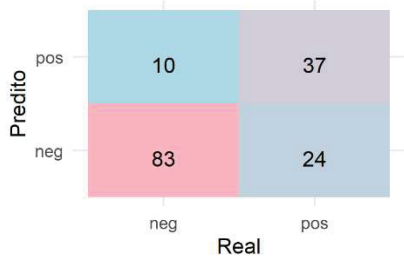
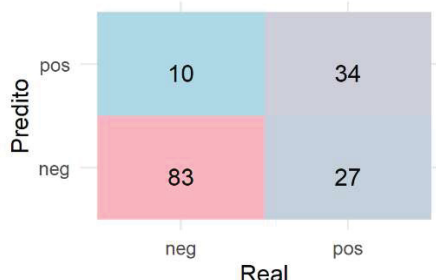
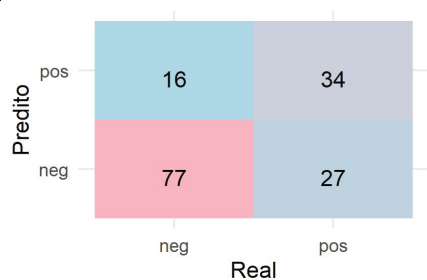
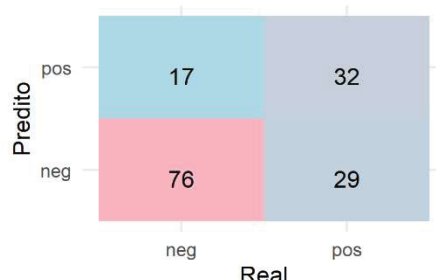
KNN	k=1	0,617647059	
-----	-----	-------------	--

Predição casos novos:

Com p	Cir c	DCir c	RadR a	PrAxis Ra	MaxL Ra	Scat Ra	Elon g	PrAxisR ect	MaxLRe ct	ScVarMa xis	ScVarma xis	RaG yr	SkewMa xis	Skewma xis	Kurtma xis	KurtMa xis	Holl Ra	Predic ao
65	34	88	120	61	10	156	40	21	140	170	320	167	67	4	8	190	200	bus
99	55	73	180	59	6	180	51	17	139	190	420	200	77	10	12	188	191	opel
113	51	91	190	67	9	233	43	24	161	200	550	189	73	7	20	191	199	bus

Diabetes

Técnica	Parâmetros	Acurácia	Matriz de Confusão
SVM - Hold-out	size=1, sigma=0.01	0,792207792	
SVM - CV	size=1, sigma=0.01	0,792207792	

RF - CV	mtry=2	0,785714286	 <table><tr><th>Predito \ Real</th><th>neg</th><th>pos</th></tr><tr><th>pos</th><td>11</td><td>39</td></tr><tr><th>neg</th><td>82</td><td>22</td></tr></table>	Predito \ Real	neg	pos	pos	11	39	neg	82	22
Predito \ Real	neg	pos										
pos	11	39										
neg	82	22										
RF - Hold-out	mtry=2	0,779220779	 <table><tr><th>Predito \ Real</th><th>neg</th><th>pos</th></tr><tr><th>pos</th><td>10</td><td>37</td></tr><tr><th>neg</th><td>83</td><td>24</td></tr></table>	Predito \ Real	neg	pos	pos	10	37	neg	83	24
Predito \ Real	neg	pos										
pos	10	37										
neg	83	24										
RNA – CV	size=11, decay=0.1	0,75974026	 <table><tr><th>Predito \ Real</th><th>neg</th><th>pos</th></tr><tr><th>pos</th><td>10</td><td>34</td></tr><tr><th>neg</th><td>83</td><td>27</td></tr></table>	Predito \ Real	neg	pos	pos	10	34	neg	83	27
Predito \ Real	neg	pos										
pos	10	34										
neg	83	27										
KNN	k=9	0,720779221	 <table><tr><th>Predito \ Real</th><th>neg</th><th>pos</th></tr><tr><th>pos</th><td>16</td><td>34</td></tr><tr><th>neg</th><td>77</td><td>27</td></tr></table>	Predito \ Real	neg	pos	pos	16	34	neg	77	27
Predito \ Real	neg	pos										
pos	16	34										
neg	77	27										
RNA - Hold-out	size=31, decay=0.7	0,701298701	 <table><tr><th>Predito \ Real</th><th>neg</th><th>pos</th></tr><tr><th>pos</th><td>17</td><td>32</td></tr><tr><th>neg</th><td>76</td><td>29</td></tr></table>	Predito \ Real	neg	pos	pos	17	32	neg	76	29
Predito \ Real	neg	pos										
pos	17	32										
neg	76	29										

Predição casos novos:

preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	Predicao
2	100	59	25	90	36	0,378	27	neg

4	127	80	30	120	40	0,5	40 neg
5	151	91	27	480	32	0,47	55 pos

REGRESSÃO

Para o experimento de Regressão:

- Ordenar por R2 descendente, ou seja, a técnica de melhor R2 ficará em primeiro na tabela.

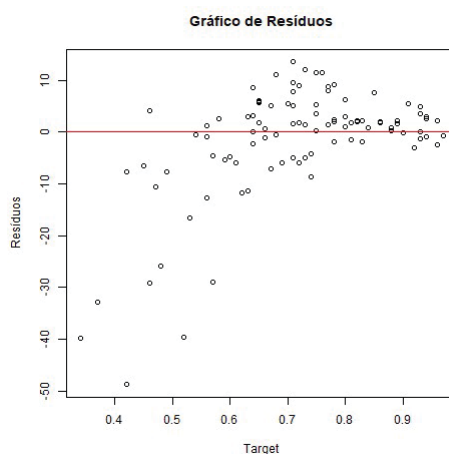
Admissão

Técnica	Parâmetros	r2	rmse	mae	pearson	syx
RNA - CV	size=41, decay=0.1	0,849462	0,057502	0,040688	0,923864	0,059673
SVM - CV	size=10, sigma=0.01	0,843919	0,058551	0,038229	0,923379	0,060761
SVM - Hold-out	size=2, sigma=0.01	0,843675	0,058597	0,038832	0,924785	0,060809
RNA - Hold-out	size=11, decay=0.1	0,830754	0,060971	0,045523	0,916951	0,063272
RF - CV	mtry=2	0,827079	0,061629	0,041319	0,911964	0,063955
RF - Hold-out	mtry=2	0,826637	0,061708	0,041125	0,912288	0,064037
KNN	k=9	0,753607	0,073566	0,054662	0,869567	0,076343

- Após o quadro, colocar:
 - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior R2 (criar um arquivo com novos casos à sua escolha)

GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	Predicao
200	100	1	3,1	1,3	9,2	1	0,420465
400	200	3	4,5	3,3	3	1	-0,06363
140	50	4	4	4	1,45	0	-0,74607

- O Gráfico de Resíduos para a técnica/parâmetro de maior R2



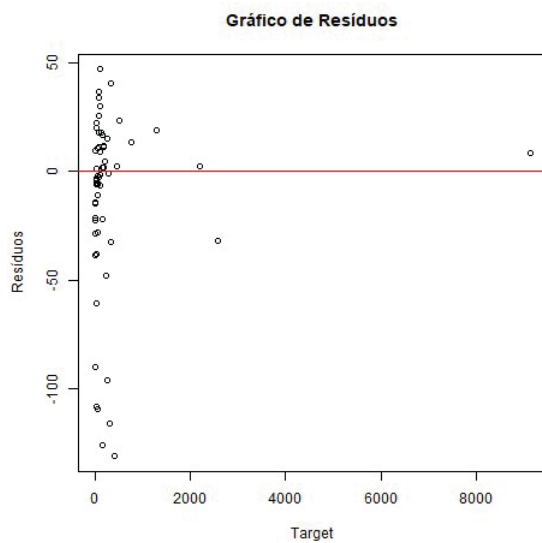
Biomassa

Técnica	Parâmetros	r2	rmse	mae	pearson	syx
RF - CV	mtry=2	0,97879017 9	178,643604 5	72,9725 2	0,99014 7	183,284 5
RF - Hold-out	mtry=2	0,97090226 4	209,241859 4	79,4515 8	0,98797 2	214,677 6
RNA - CV	size=11, decay=0.7	0,94162726 9	296,363021 9	103,718 1	0,99478 2	304,062 2
SVM - Hold-out	size=100, sigma=0.01	0,94101501 1	297,913186 2	115,408 4	0,98020 8	305,652 5
SVM - CV	size=100, sigma=0.01	0,94101501 1	297,913186 2	115,408 4	0,98020 8	305,652 5
KNN	k=3	0,84941177 7	476,008249 2	135,896 4	0,94859 6	488,374 2
RNA - Hold-out	size=11, decay=0.7	0,55057531 7	822,332462 2	348,824 2	0,80834 3	843,695 3

-
- Após o quadro, colocar:
 - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior R2 (criar um arquivo com novos casos à sua escolha)

dap	h	Me	Predicao
10	1	1	30,65512124
9	13	0,2	24,26223933
2	20	1,04	99,14148748

- O Gráfico de Resíduos para a técnica/parâmetro de maior R2



AGRUPAMENTO

Veículo

Lista de Clusters gerados:

10 primeiras linhas do arquivo com o cluster correspondente.

Usa 10 clusters no experimento.

Co mp	Ci rc	DC irc	Rad Ra	PrAxi sRa	Max LRa	Sca tRa	Elo ng	PrAxis Rect	MaxL Rect	ScVar Maxis	ScVar maxis	Ra Gyr	Skew Maxis	Skew maxis	Kurt maxis	Kurt Maxis	Holl Ra	Cla ss	cluster.result s\$cluster
95	84	83	178	72	10	162	42	20	159	176	379	184	70	6	16	187	197	va n	1
91	15	84	141	57	9	149	45	19	143	170	330	158	72	9	14	189	199	va n	3
10	5	10																sa	
4	0	6	209	66	10	207	32	23	158	223	635	220	73	14	9	188	196	ab va	1
93	14	82	159	63	9	144	46	19	143	160	309	127	63	6	10	199	207	n bu	3
85	4	70	205	103	52	149	45	19	144	241	325	188	127	9	11	180	183	s bu	6
10	5	10																bu	
7	7	6	172	50	6	255	26	28	169	280	957	264	85	5	9	181	183	s bu	6
97	34	73	173	65	6	153	42	19	143	176	361	172	66	13	1	200	204	s va	5
90	3	66	157	65	9	137	48	18	146	162	281	164	67	3	3	193	202	n va	9
86	44	62	140	61	7	122	54	17	127	141	223	112	64	2	14	200	208	n sa	4
93	4	98	197	62	11	183	36	22	146	202	505	152	64	4	14	195	204	ab	2

REGRAS DE ASSOCIAÇÃO

Musculação

Regras geradas com uma configuração de Suporte e Confiança.

rules	support	confidence	coverage	lift	count
{Agachamento} => {LegPress}	0,307692308	1	0,307692	1,238095	8
{Afundo} => {Gemeos}	0,346153846	1	0,346154	1,529412	9
{AgachamentoSmith} => {Extensor}	0,346153846	0,9	0,384615	1,8	9
{Esteira} => {Extensor}	0,423076923	0,916666667	0,461538	1,833333	11
{Extensor} => {Bicicleta}	0,461538462	0,923076923	0,5	1,714286	12
{AgachamentoSmith,Bicicleta} => {Extensor}	0,307692308	1	0,307692	2	8
{Esteira,Extensor} => {Bicicleta}	0,384615385	0,909090909	0,423077	1,688312	10
{Bicicleta,Esteira} => {Extensor}	0,384615385	1	0,384615	2	10

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 a arquitetura de RNN vista no curso.

3 GERADOR DE DÍGITOS *FAKE* (GAN)

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

4 TRADUTOR DE TEXTOS (TRANSFORMER)

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

B – RESOLUÇÃO

1 Classificação de Imagens (CNN)

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
# Carga da base
cifar10 = tf.keras.datasets.cifar10
# Já está separado em dados de treino e teste
# Não precisa separar
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [=====] - 18s 0us/step
# Normalização os dados
# Imagens em pixels de 0 - 255
```

```

# / 255.0 transforma em 0 - 1
x_train, x_test = x_train / 255.0, x_test / 255.0
# O dado y é a classe a qual faz parte
# O flatten torna os dados vetorizados
y_train, y_test = y_train.flatten(), y_test.flatten()
# Dimensão dos dados
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)
x_train.shape: (50000, 32, 32, 3)
y_train.shape: (50000,)
x_test.shape: (10000, 32, 32, 3)
y_test.shape: (10000,)
# A rede completa fica:
K = len(set(y_train))
# Aqui começa o Estágio 1
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
# Todas as imagens são do mesmo tamanho, não precisa de Global Pooling
x = Flatten()(x)
# Aqui começa o Estágio 2
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)
# Model ( lista entrada, lista saída)
model = Model(i, x)
# Relatório sobre a arquitetura da rede
model.summary()
Model: "model"

```

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 32, 32, 3)]	0
conv2d (Conv2D)	(None, 15, 15, 32)	896
conv2d_1 (Conv2D)	(None, 7, 7, 64)	18496
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dropout (Dropout)	(None, 1152)	0
dense (Dense)	(None, 1024)	1180672
dropout_1 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 10)	10250
=====		
Total params: 1284170 (4.90 MB)		
Trainable params: 1284170 (4.90 MB)		
Non-trainable params: 0 (0.00 Byte)		

```

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

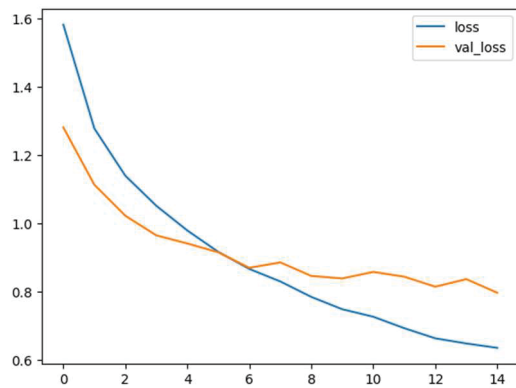
```



```

epochs=15)
Epoch 1/15
1563/1563 [=====] - 16s 6ms/step - loss: 1.5809 - accuracy: 0.4224 - val_
_loss: 1.2802 - val_accuracy: 0.5404
(...)
Epoch 15/15
1563/1563 [=====] - 9s 6ms/step - loss: 0.6350 - accuracy: 0.7743 - val_l
oss: 0.7963 - val_accuracy: 0.7307
#Plotar a função de perda, treino e validação
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

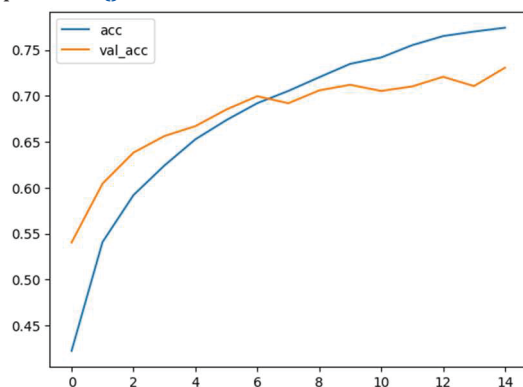
```



```

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

```



```

# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)
313/313 [=====] - 1s 2ms/step
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
show_normed=True)
(<Figure size 700x700 with 1 Axes>,
<Axes: xlabel='predicted label', ylabel='true label'>)

```

0	755 (0.76)	24 (0.02)	40 (0.04)	23 (0.02)	32 (0.03)	6 (0.01)	5 (0.01)	9 (0.01)	67 (0.07)	39 (0.04)
1	7 (0.01)	834 (0.83)	4 (0.00)	6 (0.01)	3 (0.00)	3 (0.00)	11 (0.01)	2 (0.00)	22 (0.02)	108 (0.11)
2	66 (0.07)	6 (0.01)	606 (0.61)	62 (0.06)	100 (0.10)	56 (0.06)	47 (0.05)	33 (0.03)	10 (0.01)	14 (0.01)
3	13 (0.01)	11 (0.01)	67 (0.07)	535 (0.54)	71 (0.07)	142 (0.14)	73 (0.07)	55 (0.06)	10 (0.01)	23 (0.02)
4	14 (0.01)	4 (0.00)	64 (0.06)	59 (0.06)	709 (0.71)	26 (0.03)	39 (0.04)	75 (0.07)	7 (0.01)	3 (0.00)
5	7 (0.01)	4 (0.00)	46 (0.05)	183 (0.18)	57 (0.06)	592 (0.59)	30 (0.03)	57 (0.06)	9 (0.01)	15 (0.01)
6	4 (0.00)	7 (0.01)	38 (0.04)	51 (0.05)	38 (0.04)	13 (0.01)	822 (0.82)	10 (0.01)	6 (0.01)	11 (0.01)
7	14 (0.01)	4 (0.00)	19 (0.02)	43 (0.04)	52 (0.05)	46 (0.05)	8 (0.01)	788 (0.79)	2 (0.00)	24 (0.02)
8	47 (0.05)	34 (0.03)	11 (0.01)	12 (0.01)	11 (0.01)	9 (0.01)	4 (0.00)	0 (0.00)	834 (0.83)	38 (0.04)
	18 (0.02)	75 (0.07)	7 (0.01)	13 (0.01)	6 (0.01)	5 (0.01)	5 (0.01)	17 (0.02)	22 (0.02)	832 (0.83)

Mostrar algumas classificações erradas

labels= ["airplane", "automobile", "bird", "cat", "deer", "dog",
"frog", "horse", "ship", "truck"]

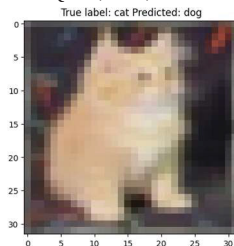
misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)

plt.imshow(x_test[i], cmap="gray")

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

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



2 Detector de SPAM (RNN)

Importação das Bibliotecas

import tensorflow **as** tf

import numpy **as** np

import matplotlib.pyplot **as** plt

import pandas **as** pd

from sklearn.model_selection **import** train_test_split

from tensorflow.keras.layers **import** Input, Embedding, LSTM, Dense

from tensorflow.keras.layers **import** GlobalMaxPooling1D

from tensorflow.keras.models **import** Model

from tensorflow.keras.preprocessing.sequence **import** pad_sequences

from tensorflow.keras.preprocessing.text **import** Tokenizer

carrega e arruma a base

!wget http://www.razer.net.br/datasets/spam.csv

df = pd.read_csv("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

```
--2024-07-06 23:45:06-- http://www.razer.net.br/datasets/spam.csv
Resolving www.razer.net.br (www.razer.net.br)... 178.128.150.229
Connecting to www.razer.net.br (www.razer.net.br)|178.128.150.229|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 503663 (492K) [text/csv]
Saving to: 'spam.csv'
```

```
spam.csv      100%[=====>] 491.86K 1.45MB/s  in 0.3s
```

```
2024-07-06 23:45:07 (1.45 MB/s) - 'spam.csv' saved [503663/503663]
```

```
# 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)
```

```
7211 tokens
```

```
# Acerta o tamanho das sequências (padding)
```

```
data_train = pad_sequences(sequences_train) # usa o tamanho da maior seq.
```

```
T = data_train.shape[1] # tamanho da sequência
```

```
data_test = pad_sequences(sequences_test, maxlen=T)
```

```
print("data_train.shape: ", data_train.shape)
```

```
print("data_test.shape: ", data_test.shape)
```

```
data_train.shape: (3733, 189)
```

```
data_test.shape: (1839, 189)
```

```
# Define o modelo
```

```
D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido
```

```
M = 5 # tamanho do hidden state, quantidade de unidades LSTM
```

```
i = Input(shape=(T,)) # Entra uma frase inteira
```

```
x = Embedding(V+1, D)(i)
```

```
x = LSTM(M)(x)
```

```
x = Dense(1, activation="sigmoid")(x) # Sigmoid pois só tem 2 valores
```

```
model = Model(i, x)
```

```
# Resumo do modelo
```

```
model.summary()
```

```
Model: "model"
```

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 189)]	0
embedding (Embedding)	(None, 189, 20)	144240
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6
=====		
Total params: 144766 (565.49 KB)		
Trainable params: 144766 (565.49 KB)		
Non-trainable params: 0 (0.00 Byte)		

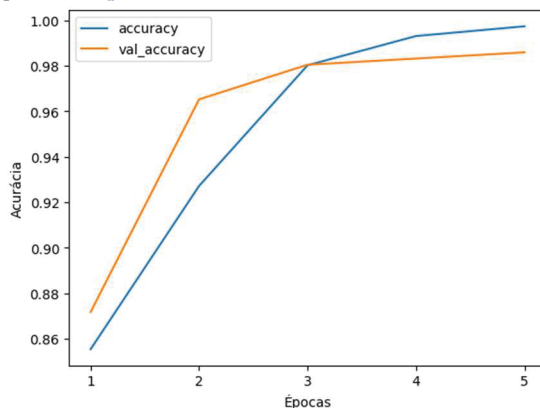
```
# Compila e treina o modelo
```

```
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 [=====] - 33s 200ms/step - loss: 0.4973 - accuracy: 0.8553 - val_
(..)
Epoch 5/5
117/117 [=====] - 3s 29ms/step - loss: 0.0570 - accuracy: 0.9973 - val_lo
ss: 0.0748 - val_accuracy: 0.9859
# Plota função de perda e acurácia
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.xlabel("Épocas")
plt.ylabel("loss")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
plt.legend()
plt.show()
plt.plot(r.history["accuracy"], label="accuracy")
plt.plot(r.history["val_accuracy"], label="val_accuracy")
plt.xlabel("Épocas")
plt.ylabel("Acurácia")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
plt.legend()
plt.show()

```



```

# Efetua a predição de um texto novo
texto = "Hi, my name is Razer and want to tell you something."
texto = "Is your car dirty? Discover our new product. Free for all. Click the link."
seq_texto = tokenizer.texts_to_sequences([texto]) # Tokeniza
data_texto = pad_sequences(seq_texto, maxlen=T) # Padding
pred = model.predict(data_texto) # Predição
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")
1/1 [=====] - 0s 27ms/step
[[0.483917]]
OK

```

3 Gerador de Dígitos Fake (GAN)

```

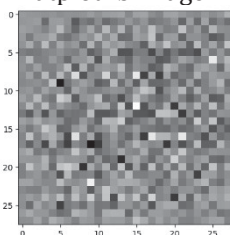
# Para Gerar os GIFs
!pip install imageio
!pip install git+https://github.com/tensorflow/docs
# Importações
import tensorflow as tf
import glob
import imageio

```

```

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)
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [=====] - 0s 0us/step
# 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')
<matplotlib.image.AxesImage at 0x7d85a1056800>

```



```

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

# Teste do DISCRIMINADOR, ainda não treinado
discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print (decision)
tf.Tensor([[0.00052592]], shape=(1, 1), dtype=float32)

# Define as funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

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

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

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

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

tf.train.Checkpoint(generator_optimizer=generator_optimizer, discriminator_optimizer=discriminator_opt
imizer, 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])
# Antes do passo de treinamento, vamos ver como são gerados os gradientes
# Seja a função  $3x^2 + 2xy$ 
# Para os valores  $x=5$  e  $y=3$  queremos as derivadas de  $f$  com respeito a  $x$  e  $y$ 
def f(x, y):
    return 3*x**2 + 2*x*y
x, y = tf.Variable(5.), tf.Variable(3.)
with tf.GradientTape() as tape:
    z = f(x, y)
gradients = tape.gradient(z, [x, y])
print(gradients)
[<tf.Tensor: shape=(), dtype=float32, numpy=36.0>, <tf.Tensor: shape=(), dtype=float32, numpy=10.0>]

# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função

```

```

@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)
        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)
        gen_loss = generator_loss(fake_output)
        disc_loss = discriminator_loss(real_output, fake_output)
        gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
        gradients_of_discriminator = disc_tape.gradient(disc_loss, discriminator.trainable_variables)
        generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.trainable_variables))
        discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
        discriminator.trainable_variables))
    # Treinamento completo/laço
def train(dataset, epochs):
    for epoch in range(epochs):
        start = time.time()
        for image_batch in dataset:
            train_step(image_batch)
        # Produce images for the GIF as you go
        display.clear_output(wait=True)
        generate_and_save_images(generator, epoch + 1, seed)
        # Save the model every 15 epochs
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)
            print('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-start))
        # Generate after the final epoch
        display.clear_output(wait=True)
        generate_and_save_images(generator, epochs, seed)
    # Gerar e salvar imagens
def generate_and_save_images(model, epoch, test_input):
    # Notice `training` is set to False.
    # This is so all layers run in inference mode (batchnorm).
    predictions = model(test_input, training=False)
    fig = plt.figure(figsize=(4, 4))
    for i in range(predictions.shape[0]):
        plt.subplot(4, 4, i+1)
        plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
        plt.axis('off')
    plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
    plt.show()
    # Treinar o modelo e restaurar o último ponto de verificação
    train(train_dataset, EPOCHS)
    checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))


<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x7d852062b7c0>
# 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 = 'drgan.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)
```

<ipython-input-30-9a34a2a6bfc1>:14: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.

```
    image = imageio.imread(filename)
```

<ipython-input-30-9a34a2a6bfc1>:16: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.

```
    image = imageio.imread(filename)
```

4 Tradutor de Textos (Transformer)

Instalação e importação

```
!pip uninstall -y 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
```

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']
```

Dataset ted_hrlr_translate downloaded and prepared to /root/tensorflow_datasets/ted_hrlr_translate/pt_to_en/1.0.0. Subsequent calls will reuse this data.

Verificar o dataset

```
for pt_examples, en_examples in train_examples.batch(3).take(1):
```

```
    for pt in pt_examples.numpy():
```

```
        print(pt.decode('utf-8'))
```

```
print()
```

```
for en in en_examples.numpy():
```

```
    print(en.decode('utf-8'))
```

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

mas e se estes fatores fossem ativos ?

mas eles não tinham a curiosidade de me testar .

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

but what if it were active ?

but they did n't test for curiosity .

Tokenização e Destokenização do texto

model_name = "ted_hrlr_translate_pt_en_converter"

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

Tem 2 tokenizers: um pt outro em en

tokenizers.en tokeniza e detokeniza

tokenizers = tf.saved_model.load(model_name)

Downloading data from https://storage.googleapis.com/download.tensorflow.org/models/ted_hrlr_translate_pt_en_converter.zip

184801/184801 [=====] - 0s 1us/step

PIPELINE DE ENTRADA

Codificar/tokenizar lotes de texto puro

def tokenize_pairs(pt, en):

pt = tokenizers.pt.tokenize(pt)

Converte ragged (irregular, tam variável) para dense

Faz padding com zeros.

pt = pt.to_tensor()

en = tokenizers.en.tokenize(en)

ragged -> dense

en = en.to_tensor()

return pt, en

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

Datasets de entrada terminam com prefetch

BUFFER_SIZE = 20000

BATCH_SIZE = 64

def make_batches(ds):

return (

ds

.cache()

.shuffle(BUFFER_SIZE)

.batch(BATCH_SIZE)

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

.prefetch(tf.data.AUTOTUNE))

train_batches = make_batches(train_examples)

val_batches = make_batches(val_examples)

def get_angles(pos, i, d_model):

angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model))

return pos * angle_rates

def positional_encoding(position, d_model):

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

np.arange(d_model)[np.newaxis, :], d_model)

sin em índices pares no array; 2i

angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])

cos em índices ímpares no array; 2i+1

angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])

newaxis, aumenta a dimensão [] -> [[]]

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

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

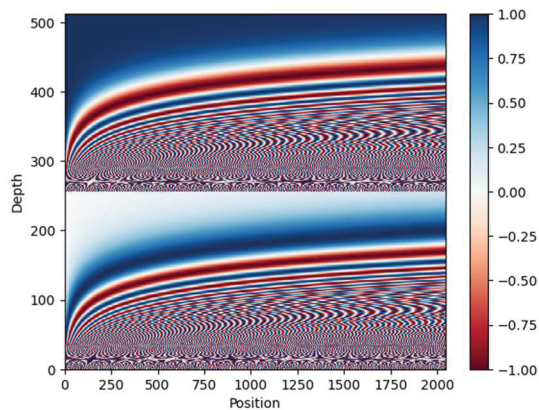
CODIFICAÇÃO POSICIONAL

n, d = 2048, 512

```

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

```



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_l)

```

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, soman 1. // (... , seq_len_q, seq_len_k)
    attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
    output = tf.matmul(attention_weights, v) # (... , seq_len_q, depth_v)
    return output, attention_weights

```

Atenção Multi-cabeças

```

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

```

```

assert d_model % self.num_heads == 0
self.depth = d_model // self.num_heads
self.wq = tf.keras.layers.Dense(d_model)
self.wk = tf.keras.layers.Dense(d_model)
self.wv = tf.keras.layers.Dense(d_model)
self.dense = tf.keras.layers.Dense(d_model)

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

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

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

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

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

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

class EncoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.1):
        super(EncoderLayer, self).__init__()
        self.mha = MultiHeadAttention(d_model, num_heads)
        self.ffn = point_wise_feed_forward_network(d_model, dff)
        self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = tf.keras.layers.Dropout(rate)
        self.dropout2 = tf.keras.layers.Dropout(rate)
    def call(self, x, training, mask):
        attn_output, _ = self.mha(x, x, x, mask) # (batch_size, input_seq_len, d_model)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(x + attn_output) # (batch_size, input_seq_len, d_model)
        ffn_output = self.ffn(out1) # (batch_size, input_seq_len, d_model)
        ffn_output = self.dropout2(ffn_output, training=training)
        out2 = self.layernorm2(out1 + ffn_output) # (batch_size, input_seq_len, d_model)
        return out2

class DecoderLayer(tf.keras.layers.Layer):

```

```

def __init__(self, d_model, num_heads, dff, rate=0.1):
    super(DecoderLayer, self).__init__()
    self.mha1 = MultiHeadAttention(d_model, num_heads)
    self.mha2 = MultiHeadAttention(d_model, num_heads)
    self.ffn = point_wise_feed_forward_network(d_model, dff)
    self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.dropout1 = tf.keras.layers.Dropout(rate)
    self.dropout2 = tf.keras.layers.Dropout(rate)
    self.dropout3 = tf.keras.layers.Dropout(rate)
def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    # enc_output.shape == (batch_size, input_seq_len, d_model)
    # (batch_size, target_seq_len, d_model)
    attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
    attn1 = self.dropout1(attn1, training=training)
    out1 = self.layernorm1(attn1 + x)
    # (batch_size, target_seq_len, d_model)
    attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1, padding_mask)
    attn2 = self.dropout2(attn2, training=training)
    out2 = self.layernorm2(attn2 + out1) # (batch_size, target_seq_len, d_model)
    ffn_output = self.ffn(out2) # (batch_size, target_seq_len, d_model)
    ffn_output = self.dropout3(ffn_output, training=training)
    out3 = self.layernorm3(ffn_output + out2) # (batch_size, target_seq_len, d_model)
    return out3, attn_weights_block1, attn_weights_block2
class Encoder(tf.keras.layers.Layer):
def __init__(self, num_layers, d_model, num_heads, dff,
    input_vocab_size, maximum_position_encoding, rate=0.1):
    super(Encoder, self).__init__()
    self.d_model = d_model
    self.num_layers = num_layers
    self.embedding = tf.keras.layers.Embedding(input_vocab_size, d_model)
    self.pos_encoding = positional_encoding(maximum_position_encoding, self.d_model)
    self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate) for _ in range(num_layers)]
    self.dropout = tf.keras.layers.Dropout(rate)
def call(self, x, training, mask):
    seq_len = tf.shape(x)[1]
    # adding embedding and position encoding.
    x = self.embedding(x) # (batch_size, input_seq_len, d_model)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]
    x = self.dropout(x, training=training)
    for i in range(self.num_layers):
        x = self.enc_layers[i](x, training, mask)
    return x # (batch_size, input_seq_len, d_model)
class Decoder(tf.keras.layers.Layer):
def __init__(self, num_layers, d_model, num_heads, dff, target_vocab_size,
    maximum_position_encoding, rate=0.1):
    super(Decoder, self).__init__()
    self.d_model = d_model
    self.num_layers = num_layers
    self.embedding = tf.keras.layers.Embedding(target_vocab_size, d_model)
    self.pos_encoding = positional_encoding(maximum_position_encoding, d_model)
    self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate)
        for _ in range(num_layers)]
    self.dropout = tf.keras.layers.Dropout(rate)
def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    seq_len = tf.shape(x)[1]
    attention_weights = {}

```

```

x = self.embedding(x) # (batch_size, target_seq_len, d_model)
x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
x += self.pos_encoding[:, :seq_len, :]
x = self.dropout(x, training=training)
for i in range(self.num_layers):
    x, block1, block2 = self.dec_layers[i](x, enc_output, training,
                                           look_ahead_mask, padding_mask)
    attention_weights[f'decoder_layer{i+1}_block1'] = block1
    attention_weights[f'decoder_layer{i+1}_block2'] = block2
# x.shape == (batch_size, target_seq_len, d_model)
return x, attention_weights
class Transformer(tf.keras.Model):
    def __init__(self, num_layers, d_model, num_heads, dff, input_vocab_size,
                 target_vocab_size, pe_input, pe_target, rate=0.1):
        super().__init__()
        self.encoder = Encoder(num_layers, d_model, num_heads, dff, input_vocab_size,
                               pe_input, rate)
        self.decoder = Decoder(num_layers, d_model, num_heads, dff, target_vocab_size,
                               pe_target, rate)
        self.final_layer = tf.keras.layers.Dense(target_vocab_size)
    def call(self, inputs, training):
        # Keras models prefer if you pass all your inputs in the first argument
        inp, tar = inputs
        enc_padding_mask, look_ahead_mask, dec_padding_mask = self.create_masks(inp, tar)
        # (batch_size, inp_seq_len, d_model)
        enc_output = self.encoder(inp, training, enc_padding_mask)
        # dec_output.shape == (batch_size, tar_seq_len, d_model)
        dec_output, attention_weights = self.decoder(tar, enc_output,
                                                    training, look_ahead_mask, dec_padding_mask)
        # (batch_size, tar_seq_len, target_vocab_size)
        final_output = self.final_layer(dec_output)
        return final_output, attention_weights
    def create_masks(self, inp, tar):
        # Encoder padding mask
        enc_padding_mask = create_padding_mask(inp)
        # Used in the 2nd attention block in the decoder.
        # This padding mask is used to mask the encoder outputs.
        dec_padding_mask = create_padding_mask(inp)
        # Used in the 1st attention block in the decoder.
        # It is used to pad and mask future tokens in the input received by
        # the decoder.
        look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])
        dec_target_padding_mask = create_padding_mask(tar)
        look_ahead_mask = tf.maximum(dec_target_padding_mask, look_ahead_mask)
        return enc_padding_mask, look_ahead_mask, dec_padding_mask
# Hiperparâmetros
num_layers = 4
d_model = 128
dff = 512
num_heads = 8
dropout_rate = 0.1
class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
    def __init__(self, d_model, warmup_steps=4000):
        super(CustomSchedule, self).__init__()
        self.d_model = d_model
        self.d_model = tf.cast(self.d_model, tf.float32)
        self.warmup_steps = warmup_steps
    def __call__(self, step):
        step = tf.cast(step, tf.float32) # Adicionado para evitar ERRO

```

```

arg1 = tf.math.rsqrt(step)
arg2 = step * (self.warmup_steps ** -1.5)
return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

learning_rate = CustomSchedule(d_model)
optimizer = tf.keras.optimizers.Adam(learning_rate, beta_1=0.9, beta_2=0.98, epsilon=1e-9)
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True, reduction='none')
def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask
    return tf.reduce_sum(loss_)/tf.reduce_sum(mask)
def accuracy_function(real, pred):
    accuracies = tf.equal(real, tf.argmax(pred, axis=2))
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    accuracies = tf.math.logical_and(mask, accuracies)
    accuracies = tf.cast(accuracies, dtype=tf.float32)
    mask = tf.cast(mask, dtype=tf.float32)
    return tf.reduce_sum(accuracies)/tf.reduce_sum(mask)

train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy = tf.keras.metrics.Mean(name='train_accuracy')
transformer = Transformer(num_layers=num_layers,
                          d_model=d_model,
                          num_heads=num_heads,
                          dff=dff,
                          input_vocab_size=tokenizers.pt.get_vocab_size().numpy(),
                          target_vocab_size=tokenizers.en.get_vocab_size().numpy(),
                          pe_input=1000,
                          pe_target=1000,
                          rate=dropout_rate)
checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(transformer=transformer,
                           optimizer=optimizer)
ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path, max_to_keep=5)
# if a checkpoint exists, restore the latest checkpoint.
if ckpt_manager.latest_checkpoint:
    ckpt.restore(ckpt_manager.latest_checkpoint)
    print('Latest checkpoint restored!!')
EPOCHS = 20
train_step_signature = [
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]
@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
    tar_inp = tar[:, :-1]
    tar_real = tar[:, 1:]
    with tf.GradientTape() as tape:
        predictions, _ = transformer([inp, tar_inp],
                                     training = True)
        loss = loss_function(tar_real, predictions)
        gradients = tape.gradient(loss, transformer.trainable_variables)
        optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))
        train_loss(loss)
        train_accuracy(accuracy_function(tar_real, predictions))
for epoch in range(EPOCHS):
    start = time.time()

```



```

train_loss.reset_state()
train_accuracy.reset_state()
# inp -> portuguese, tar -> english
for (batch, (inp, tar)) in enumerate(train_batches):
    train_step(inp, tar)
    if batch % 50 == 0:
        print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}')
    if (epoch + 1) % 5 == 0:
        ckpt_save_path = ckpt_manager.save()
        print(f'Saving checkpoint for epoch {epoch+1} at {ckpt_save_path}')
        print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy {train_accuracy.result():.4f}')
        print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')
Epoch 1 Batch 0 Loss 8.8473 Accuracy 0.0000
Epoch 1 Batch 50 Loss 8.7913 Accuracy 0.0120
(...)
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-4
Epoch 20 Loss 1.4442 Accuracy 0.6807
Time taken for 1 epoch: 97.37 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
translator = Translator(tokenizers, transformer)
sentence = "Eu li sobre triceratops na enciclopédia."
translated_text, translated_tokens, attention_weights = translator(tf.constant(sentence))
print(f'{"Prediction":15s}: {translated_text}')
Prediction   : b'i read about triumphs in encyclood .'

```

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

Projeto de sistema de Big Data para investimentos no mercado de ações brasileiro(B3).

1. Introdução

O presente trabalho tem como objetivo apresentar um projeto para um sistema de Big Data para que contemple as etapas mínimas para coletar, transformar, estruturar, analisar, modelar e visualizar dados do mercado de ações nacional com o objetivo de construir um sistema de apoio à decisão para auxiliar na melhor alocação de recursos financeiros.

2. Fontes de Dados

Dados de Mercado:

- Preços em tempo real (tickers).
- Preços históricos (OHLCV).
- Book de ofertas.

Dados Financeiros:

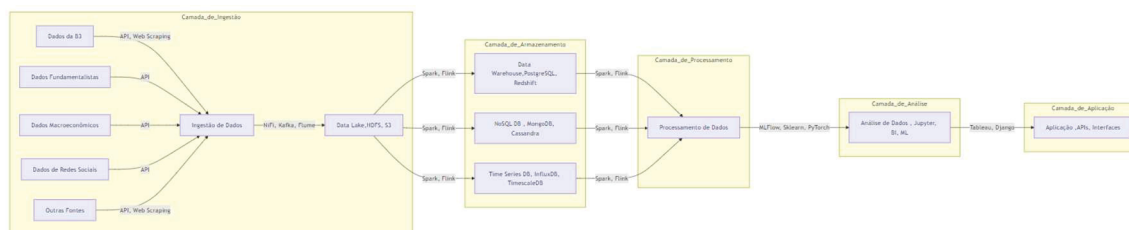
- Balanços patrimoniais.
- Demonstrações de resultados.
- Indicadores fundamentalistas.
- Dados macroeconômicos: Taxa de juros, Inflação, PIB, etc.

Notícias e Sentimento de Mercado:

- Notícias financeiras.
- Dados de mídias sociais.
- Dados Macroeconômicos:

3. Arquitetura Geral do Sistema

A arquitetura geral do sistema é apresentada no diagrama abaixo.



A seguir são apresentados os componentes principais da arquitetura geral do sistema proposto.

Camada de Ingestão de Dados:

Coleta de dados de múltiplas fontes em tempo real.

Ferramentas:

- Apache Kafka: Plataforma de streaming para ingestão de dados em tempo real.
- Apache Flume: Plataforma para coletar, agregar e mover grandes quantidades de dados.
- Apache NiFi: Automatização de fluxos de dados e ETL.
- APIs e Feeds de Dados: Integração com provedores como a B3, Bloomberg, Reuters.

Camada de Armazenamento:

Data Lake para armazenamento dos dados brutos.

Data Warehouse para dados estruturados e processados.

Ferramentas:

- Data Lake: Utilizando Hadoop HDFS ou Amazon S3 para armazenamento de dados brutos.
- Data Warehouse: PostgreSQL, Amazon Redshift, Google BigQuery ou Snowflake para dados estruturados.
- Bancos de Dados SQL: PostgreSQL.
- Bancos de Dados Temporais: InfluxDB, TimeScaleDB.
- Bancos de Dados NoSQL: Apache HBase, Cassandra, MongoDB.

Processamento:

Processamento em lote e em tempo real.

Limpeza, transformação e enriquecimento de dados.

Ferramentas:

- Apache Spark: Processamento distribuído para grandes volumes de dados.
- Apache Flink: Processamento em tempo real de fluxos de dados.

- Airflow: Orquestração de workflows e agendamento de tarefas.

Análise dados e Machine Learning:

Descoberta de padrões.

Treinamento e validação de modelos.

Ferramentas:

- Linguagens: Python, R.
- Bibliotecas: Pandas, NumPy, Scikit-learn, TensorFlow, Keras, PyTorch.
- Plataformas de ML: MLflow para gerenciamento de modelos.
- Notebooks: Jupyter Notebooks para análise de dados interativa.

Aplicação, GUI e Visualização:

Interface para usuários finais.

Dashboards, relatórios e APIs.

Ferramentas:

- Dashboards: Power BI, Tableau, Power BI, Grafana.
- Aplicações Web: Django, Flask para desenvolvimento de GUI personalizadas.

4. Modelo de Dados

4.1. Esquema de Dados Estruturados

Tabela de Preços Históricos:

Campos: Data/Hora, Ação, Preço de Abertura, Máximo, Mínimo, Preço de Fechamento, Volume.

Tabela de Dados Financeiros:

Campos: Ação, Período, Receita, Lucro Líquido, EBITDA, etc.

Tabela de Notícias e Sentimento:

Campos: Data/Hora, Fonte, Título, Corpo, Sentimento (positivo, negativo, neutro).

4.2. Metadados e Catálogo de Dados

Catálogo de Dados: Utilização de um catálogo para gestão e descoberta de dados.

Schemas e Validações: Definição de schemas para validação de dados durante a ingestão.

4.3. Dados não estruturados

Os textos de notícias, posts de redes sociais, entre outros dados não estruturados, serão armazenados em sistema de arquivos distribuído Hadoop HDFS ou Amazon S3.

5. Pipeline de Dados

Ingestão:

- Coleta de dados em tempo real via Kafka.
- Ingestão de dados históricos em lote.

Armazenamento Inicial:

- Dados brutos armazenados no Data Lake.

Processamento Primário:

- Limpeza e transformação usando Spark.
- Cálculo de indicadores técnicos e fundamentalistas.

Armazenamento Processado:

- Dados transformados armazenados no Data Warehouse.

Análise e Modelagem:

- Análise exploratória de dados.
- Treinamento de modelos de machine learning.

Distribuição e Visualização:

- Dados disponibilizados para aplicações e dashboards.
- APIs para acesso a modelos preditivos.

6. Desenvolvimento de Estratégias e Portfólios

Análise Técnica:

- Identificação de padrões gráficos.
- Utilização de indicadores como RSI, MACD.

Análise Fundamentalista:

- Avaliação de empresas com base em indicadores financeiros.

Machine Learning e IA:

- Modelos de previsão de preços.
- Clusterização de ativos.
- Análise de sentimento com NLP.

Backtesting:

- Teste de estratégias em dados históricos.
- Avaliação de desempenho e risco.

Conclusão

Foram apresentados a concepção e projeto geral para um sistema de apoio à decisão de investimentos no mercado financeiro, especificamente do mercado de ações brasileiro (B3). O projeto contempla uma arquitetura básica, ferramentas a serem utilizadas, um pipeline de dados e produtos e serviços finais.

Os dados envolvidos incluem dados financeiros de alta frequência (ticks), books de ofertas, notícias financeiras, redes sociais, relatórios trimestrais, dados relacionados ao cenário do

mercado financeiro local e global, entre outros. Novas fontes de dados podem ser adicionadas sempre que houver necessidade. Como trata-se de uma proposta ampla, estão envolvidos todos os V's do Big Data.

A seguir, são feitos breves comentários sobre cada um dos V's e seu impacto no projeto desse trabalho.

- **Volume:** O histórico dos preços, volumes e books de ofertas embora sejam relativamente compactos são de uma grande variedade de ativos e são coletados e armazenados de forma cumulativa, o que leva a um volume significativo e com tendência a aumentar ao longo do tempo. As informações relacionadas às menções aos ativos e cenário financeiro coletadas nas redes sociais tendem a gerar um volume mais significativo, pois são dados não estruturados e que também serão coletados de forma cumulativa ao longo do tempo. Além disso, o fato de o sistema estar aberto a novas fontes de dados, faz com que o volume seja um ponto muito importante no projeto.
- **Velocidade:** Os preços e livros de ofertas são atualizados praticamente em tempo real e o tempo é importante para algumas estratégias que exploram oportunidades que oferecem uma janela de tempo muito estreita. Além dos dados financeiros propriamente ditos, as fontes auxiliares de dados, como as menções nas redes sociais também são produzidas em uma velocidade alta. Assim, há necessidade de ingestão e processamento em tempo praticamente real para identificar oportunidades de investimento.
- **Variedade:** Como o sistema pretende ser um sistema completo que agregue qualquer tipo de informação relevante para apoiar a tomada de decisão, vários tipos de dados serão utilizados, como os dados estruturados de preços, indicadores financeiros, assim como dados não estruturados vindos de notícias, posts em redes sociais, entre outros.
- **Veracidade:** É importante garantir a confiabilidade dos dados para minimizar o risco de decisões erradas.
- **Valor:** O mais importante dos V's, pois sem ele nenhum dos outros V's teria sentido. Aqui o valor é oferecer aos investidores informações que subsidiem a tomada de decisões de investimentos que levem ao atingimento dos seus objetivos de forma racional e objetiva.

APÊNDICE 10 – VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

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

B – RESOLUÇÃO

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:

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

#TODO

"""

** Ajustar dataset validação para ter 1 exemplo de cada classe*

"""

'\n Ajustar dataset validação para ter 1 exemplo de cada classe\n'*

Módulo para imprimir os gráficos de treinamento de forma dinâmica.

!pip install livelossplot

Baixando e descompactando datasets

```
!wget https://github.com/valeriow/IAA-PUB/raw/refs/heads/main/Test_Warwick.zip
!wget https://github.com/valeriow/IAA-PUB/raw/refs/heads/main/Train_Warwick.zip
!unzip -o Test_Warwick.zip
!unzip -o Train_Warwick.zip
```

Archive: Test_Warwick.zip

inflating: Test_4cl_amostra/0/70_HER2_23828.png

inflating: Test_4cl_amostra/0/70_HER2_23827.png

(...)

inflating: Train_4cls_amostra/1/15_HER2_16661.png

inflating: Train_4cls_amostra/1/15_HER2_16897.png

inflating: Train_4cls_amostra/1/14_HER2_8051.png

import random

import glob

import os

from collections **import** defaultdict

import shutil

import math

import cv2

import numpy **as** np

from skimage.feature **import** local_binary_pattern

import matplotlib.pyplot **as** plt

import pandas **as** pd

from tensorflow.keras.models **import** Model

#from tensorflow.keras.applications.resnet50 import preprocess_input

#from tensorflow.keras.applications.resnet50 import ResNet50

from tensorflow.keras.preprocessing.image **import** ImageDataGenerator

from tensorflow.keras.layers **import** Dense, Flatten

from tensorflow.keras.applications.vgg16 **import** preprocess_input

from tensorflow.keras.applications **import** VGG16

from sklearn.svm **import** SVC

from sklearn.ensemble **import** RandomForestClassifier

from sklearn.neural_network **import** MLPClassifier

from sklearn.metrics **import** classification_report, accuracy_score

Obtendo a lista de pacientes para fazer a separação da base de treinamento em treino e validação

Obtém a lista de arquivos

all_train_images = glob.glob("/content/Train_4cls_amostra/*/*.png")

Itera sobre a lista de arquivos obtendo o id do paciente

patients_images = defaultdict(list)

for img_path **in** all_train_images:

 filename = os.path.basename(img_path)

 patient_id = filename.split('_')[0] *# Pega o XX (número do paciente)*

 patients_images[patient_id].append(img_path)

Obter a lista de pacientes (as chaves do defaultdict)

patients = list(patients_images.keys())

random.seed(42)

random.shuffle(patients)

```

# Dividir 80% para treino e 20% para validação
num_patients = len(patients)
num_train = math.ceil(0.8 * num_patients)

train_patients = patients[:num_train] # Primeiro 80% dos pacientes
val_patients = patients[num_train:] # Último 20% dos pacientes

# Função para mover os arquivos para as respectivas pastas
def move_images(patients_list, destination_folder):
    if not os.path.exists(destination_folder):
        os.makedirs(destination_folder)

    for patient_id in patients_list:
        for img_path in patients_images[patient_id]:
            if img_path: # Verifica se há um caminho de imagem
                subdir = img_path.split('/')[2]
                # Define o caminho de destino
                destination_path = os.path.join(destination_folder, subdir)
                # Criar o diretório de destino se não existir
                if not os.path.exists(destination_path):
                    os.makedirs(destination_path)
                # Move o arquivo para o diretório de destino
                #print(img_path, destination_path)
                shutil.copy(img_path, destination_path)

# Diretórios para armazenar treino e validação
train_dir = '/content/train/'
val_dir = '/content/valid/'
test_dir = '/content/test/'

# Exclui diretórios antes da cópia se existirem
for dir in [train_dir, val_dir, test_dir]:
    if os.path.exists(dir):
        shutil.rmtree(dir)

# Copiar 80% dos pacientes para a pasta train
move_images(train_patients, train_dir)

# Copiar 20% dos pacientes para a pasta valid
move_images(val_patients, val_dir)

# Copiar os pacientes para a pasta test
shutil.copytree("/content/Test_4cl_amostra", f"{test_dir}")
'/content/test/'

# Função para calcular LBP de uma imagem
def calculate_lbp(image_path, radius=1, n_points=8):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    lbp = local_binary_pattern(img, n_points, radius, method='uniform')
    return lbp

# Função que calcula o histograma lbp de uma imagem.
def calculate_lbp_hist(image_path, radius=1, n_points=8):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    height, width = img.shape
    lbp = local_binary_pattern(img, n_points, radius, method='uniform')
    # (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range=(0, n_points + 2))
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, height+1), range=(0, width + 1))
    hist = hist.astype("float")
    hist /= hist.sum() # Normalizar o histograma
    return hist

```


Função auxiliar que separa caminho, nome e extensão de um caminho completo

```
def separa_caminho(img_path):
    len_img_path = len(img_path)
    img_path_split = img_path.split('/')
    nome_arquivo = img_path_split[-1].split('.')[0]
    extensao_arquivo = img_path_split[-1].split('.')[1]
    caminho_arquivo = "/" + ".".join(img_path_split[:len(img_path_split)-1]) + "/"
    return (nome_arquivo, extensao_arquivo, caminho_arquivo)
```

Percorre todas as imagens de treinamento e extrai LPB de cada salva num arquivo com o mesmo nome da imagem + _lbp.csv

```
def percorre_arquivos_extraindo_caracteristicas_lbp(base_dir="train", radius=1, n_points=8):
```

#base_dir = train_dir

```
files = glob.glob(f"{base_dir}/*/*.png")
```

```
for img_path in files:
```

```
    lbp_hist = calculate_lbp_hist(img_path, radius=1, n_points=8)
```

```
    lbp_hist = lbp_hist.reshape(1, -1)
```

```
    nome_arquivo, extensao_arquivo, caminho_arquivo = separa_caminho(img_path)
```

```
    #novo_arquivo = f"{caminho_arquivo}{nome_arquivo}_lbp.npy"
```

```
    #np.save(novo_arquivo, lbp_hist)
```

```
    novo_arquivo = f"{caminho_arquivo}{nome_arquivo}_lbp.csv"
```

```
    np.savetxt(novo_arquivo, lbp_hist, delimiter=',', fmt='%f')
```

Percorre todas as imagens de treinamento e extrai LPB de cada salva num arquivo com o mesmo nome da imagem + _lbp.npy

```
percorre_arquivos_extraindo_caracteristicas_lbp("train", radius=1, n_points=8)
```

```
percorre_arquivos_extraindo_caracteristicas_lbp("valid", radius=1, n_points=8)
```

```
percorre_arquivos_extraindo_caracteristicas_lbp("test", radius=1, n_points=8)
```

```
classes = [classe.split("/")[-1] for classe in glob.glob("/content/Train_4cls_amostra/*")]
```

```
classes
```

```
['3', '1', '0', '2']
```

#Modelo que será utilizado, sem a camada top (adicionada manualmente depois)

```
vgg16_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
```

```
vgg16_model.trainable = False
```

```
def extrai_caracteristicas_vgg16(img_path, model):
```

```
    img = cv2.imread(img_path)
```

```
    img = cv2.resize(img, (224, 224))
```

```
    img = np.expand_dims(img, axis=0)
```

```
    return model.predict(img)
```

```
def percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir, model):
```

```
    files = glob.glob(f"{base_dir}/*/*.png")
```

```
    for img_path in files:
```

```
        nome_arquivo, extensao_arquivo, caminho_arquivo = separa_caminho(img_path)
```

```
        novo_arquivo = f"{caminho_arquivo}{nome_arquivo}_vgg16.csv"
```

```
        data = extrai_caracteristicas_vgg16(img_path, model)
```

```
        data = data.reshape(1, -1)
```

```
        np.savetxt(novo_arquivo, data, delimiter=',', fmt='%f')
```

```
        #np.save(novo_arquivo, extrai_caracteristicas_vgg16(img_path))
```

```
"""base_dir = 'train'
```

```
train_files = glob.glob(f"{base_dir}/*/*.png")
```

```
base_dir = 'valid'
```

```
val_files = glob.glob(f"{base_dir}/*/*.png") """
```

```
percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir= "train", model = vgg16_model)
```

```
percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir= "valid", model = vgg16_model)
```

```
percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir= "test", model = vgg16_model)
```

```

1/1 ————— 1s 857ms/step
(...)
1/1 ————— 1s 541ms/step
base_dir = 'train'
paths_lbp_features_train = glob.glob(f'{base_dir}/*/*_lbp.csv')
paths_vgg16_features_train = glob.glob(f'{base_dir}/*/*_vgg16.csv')

base_dir = 'valid'
paths_lbp_features_valid = glob.glob(f'{base_dir}/*/*_lbp.csv')
paths_vgg16_features_valid = glob.glob(f'{base_dir}/*/*_vgg16.csv')

base_dir = 'test'
paths_lbp_features_test = glob.glob(f'{base_dir}/*/*_lbp.csv')
paths_vgg16_features_test = glob.glob(f'{base_dir}/*/*_vgg16.csv')
# Percorre os arquivos csv com as características LBP e VGG16 extraídas antes para cada dataset (train, valid, test)

#LBP
lbp_features_train = []
lbp_labels_train = []
for file_path in paths_lbp_features_train:
    data = np.loadtxt(file_path, delimiter=',')
    lbp_features_train.append(data)
    classe = separa_caminho(file_path)[2].split("/")[1]
    lbp_labels_train.append(classe)

lbp_features_valid = []
lbp_labels_valid = []
for file_path in paths_lbp_features_valid:
    data = np.loadtxt(file_path, delimiter=',')
    lbp_features_valid.append(data)
    classe = separa_caminho(file_path)[2].split("/")[1]
    lbp_labels_valid.append(classe)

lbp_features_test = []
lbp_labels_test = []
for file_path in paths_lbp_features_test:
    data = np.loadtxt(file_path, delimiter=',')
    lbp_features_test.append(data)
    classe = separa_caminho(file_path)[2].split("/")[1]
    lbp_labels_test.append(classe)

#VGG16
vgg16_features_train = []
vgg16_labels_train = []
for file_path in paths_vgg16_features_train:
    data = np.loadtxt(file_path, delimiter=',')
    vgg16_features_train.append(data)
    classe = separa_caminho(file_path)[2].split("/")[1]
    vgg16_labels_train.append(classe)

vgg16_features_valid = []
vgg16_labels_valid = []
for file_path in paths_vgg16_features_valid:
    data = np.loadtxt(file_path, delimiter=',')
    vgg16_features_valid.append(data)
    classe = separa_caminho(file_path)[2].split("/")[1]
    vgg16_labels_valid.append(classe)

```

```

vgg16_features_test = []
vgg16_labels_test = []
for file_path in paths_vgg16_features_test:
    data = np.loadtxt(file_path, delimiter=',')
    vgg16_features_test.append(data)
    classe = separa_caminho(file_path)[2].split("/")[1]
    vgg16_labels_test.append(classe)
print(len(lbp_features_valid), len(lbp_labels_valid), len(lbp_labels_test))
117 117 371
from sklearn.metrics import confusion_matrix

def calculate_metrics(cm):
    # Calcular a matriz de confusão
    # cm = confusion_matrix(y_val, val_predictions)

    # Inicializar contadores para micro F1-Score
    TP_total = 0
    FP_total = 0
    FN_total = 0
    TN_total = 0
    f1_scores = []
    sensibilities = []
    specificities = []

    # Número de classes
    num_classes = cm.shape[0]

    # Calcular precisão, recall e F1-Score para cada classe
    for i in range(num_classes):
        TP = cm[i, i]
        FP = cm[:, i].sum() - TP # Falsos positivos
        FN = cm[i, :].sum() - TP # Falsos negativos
        TN = cm.sum() - (TP + FP + FN) # Verdadeiros negativos

        # Calcular Precisão e Recall
        precision = TP / (TP + FP) if (TP + FP) > 0 else 0
        recall = TP / (TP + FN) if (TP + FN) > 0 else 0
        sensibilidade = TP / (TP + FN) if (TP + FN) > 0 else 0
        especificidade = TN / (TN + FP) if (TN + FP) > 0 else 0

        f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

        # Acumular valores
        TP_total += TP
        FP_total += FP
        FN_total += FN
        TN_total += TN

        f1_scores.append(f1)
        sensibilities.append(sensibilidade)
        specificities.append(especificidade)

    accuracy = (TP_total + TN_total) / (TP_total + TN_total + FP_total + FN_total) if (TP_total + TN_total +
    FP_total + FN_total) > 0 else 0

    n_classes = [cm[i, :].sum() for i in range(num_classes)] # Número de instâncias por classe

    # Cálculo do F1-Score Micro

```

```

#precision_micro = TP_total / (TP_total + FP_total) if (TP_total + FP_total) > 0 else 0
#recall_micro = TP_total / (TP_total + FN_total) if (TP_total + FN_total) > 0 else 0
#f1_micro = 2 * (precision_micro * recall_micro) / (precision_micro + recall_micro) if (precision_micro +
recall_micro) > 0 else 0

# Cálculo do F1-Score Weighted
f1_weighted = sum(f1 * n for f1, n in zip(f1_scores, n_classes)) / sum(n_classes) if sum(n_classes) > 0
else 0

# Sensibilidade ponderada
sensibility_weighted = sum(sens * n for sens, n in zip(sensibilities, n_classes)) / sum(n_classes) if
sum(n_classes) > 0 else 0

# Especificidade ponderada
specificity_weighted = sum(spec * n for spec, n in zip(specificities, n_classes)) / sum(n_classes) if
sum(n_classes) > 0 else 0

return accuracy, sensibility_weighted, specificity_weighted, f1_weighted
def train_evaluate(model, X_train, y_train, X_val, y_val):
    model.fit(X_train, y_train)
    train_predictions = model.predict(X_train)
    score_train = accuracy_score(y_train, train_predictions)

    val_predictions = model.predict(X_val)
    score_val = accuracy_score(y_val, val_predictions)

    return model, train_predictions, val_predictions, score_train, score_val
#Modelos para características LBP
rf_lbp = RandomForestClassifier(n_estimators=100)
svm_lbp = SVC(kernel='linear')
mlp_lbp = MLPClassifier(hidden_layer_sizes=(40, 40, 40, 4), max_iter=10000, random_state=42)

#Modelos para características VGG16
rf_vgg16 = RandomForestClassifier(n_estimators=100)
svm_vgg16 = SVC(kernel='linear')
mlp_vgg16 = MLPClassifier(hidden_layer_sizes=(40, 40, 40, 4), max_iter=10000, random_state=42)
# Vamos usar inicialmente as características lbp

X_train = lbp_features_train
y_train = lbp_labels_train

X_val = lbp_features_valid
y_val = lbp_labels_valid

X_test = lbp_features_test
y_test = lbp_labels_test
def metricas(tipo, df_resultados, y_val, val_predictions):
    cm = confusion_matrix(y_val, val_predictions)
    accuracy, sensibility_weighted, specificity_weighted, f1_weighted = calculate_metrics(cm)
    data = { 'Tipo': tipo,
            'Acurácia': [accuracy],
            'Sensibilidade Ponderada': [sensibility_weighted],
            'Especificidade Ponderada': [specificity_weighted],
            'F1 Ponderada': [f1_weighted]}

    df_resultados = pd.concat([df_resultados, pd.DataFrame(data)], ignore_index=True)
    return df_resultados
df_resultados_valid = pd.DataFrame()
df_resultados_test = pd.DataFrame()

```

```

#Random Forest
model, train_predictions, val_predictions, score_train, score_val = train_evaluate(model=rf_lbp,
X_train=X_train, y_train=y_train, X_val=X_val, y_val=y_val)
tipo = 'rf_lbp'
#Resultados Validação
df_resultados_valid = metricas(tipo, df_resultados_valid, y_val, val_predictions)
df_resultados_valid
#Resultados Test
test_predictions = model.predict(X_test)
df_resultados_test = metricas(tipo, df_resultados_test, y_test, test_predictions)
#SVM
model, train_predictions, val_predictions, score_train, score_val = train_evaluate(model=svm_lbp,
X_train=X_train, y_train=y_train, X_val=X_val, y_val=y_val)

tipo = 'svm_lbp'
#Resultados Validação
df_resultados_valid = metricas(tipo, df_resultados_valid, y_val, val_predictions)
df_resultados_valid
#Resultados Test
test_predictions = model.predict(X_test)
df_resultados_test = metricas(tipo, df_resultados_test, y_test, test_predictions)
#RNA
model, train_predictions, val_predictions, score_train, score_val = train_evaluate(model=mlp_lbp,
X_train=X_train, y_train=y_train, X_val=X_val, y_val=y_val)
tipo = 'rna_lbp'
#Resultados Validação
df_resultados_valid = metricas(tipo, df_resultados_valid, y_val, val_predictions)
df_resultados_valid
#Resultados Test
test_predictions = model.predict(X_test)
df_resultados_test = metricas(tipo, df_resultados_test, y_test, test_predictions)
# Vamos usar agora as características extraídas da última camada de convolução da Vgg16

X_train = vgg16_features_train
y_train = vgg16_labels_train

X_val = vgg16_features_valid
y_val = vgg16_labels_valid

X_test = vgg16_features_test
y_test = vgg16_labels_test
#Random Forest
model, train_predictions, val_predictions, score_train, score_val = train_evaluate(model=rf_vgg16,
X_train=X_train, y_train=y_train, X_val=X_val, y_val=y_val)
tipo = 'rf_vgg16'
#Resultados Validação
df_resultados_valid = metricas(tipo, df_resultados_valid, y_val, val_predictions)
df_resultados_valid
#Resultados Test
test_predictions = model.predict(X_test)
df_resultados_test = metricas(tipo, df_resultados_test, y_test, test_predictions)
#SVM
model, train_predictions, val_predictions, score_train, score_val = train_evaluate(model=svm_vgg16,
X_train=X_train, y_train=y_train, X_val=X_val, y_val=y_val)
tipo = 'svm_vgg16'
#Resultados Validação
df_resultados_valid = metricas(tipo, df_resultados_valid, y_val, val_predictions)
df_resultados_valid
#Resultados Test

```

```

test_predictions = model.predict(X_test)
df_resultados_test = metricas(tipo, df_resultados_test, y_test, test_predictions)
#RNA
model, train_predictions, val_predictions, score_train, score_val = train_evaluate(model=mlp_vgg16,
X_train=X_train, y_train=y_train, X_val=X_val, y_val=y_val)
tipo = 'rna_vgg16'
#Resultados Validação
df_resultados_valid = metricas(tipo, df_resultados_valid, y_val, val_predictions)
df_resultados_valid
#Resultados Test
test_predictions = model.predict(X_test)
df_resultados_test = metricas(tipo, df_resultados_test, y_test, test_predictions)
#Ordenando os resultados de validação por 'F1 Ponderada'
df_resultados_valid = df_resultados_valid.sort_values('F1 Ponderada', ascending=False)
df_resultados_valid

```

	Tipo	Acurária	Sensibilidade Ponderada	Especificidade Ponderada	F1 Ponderada
4	svm_vgg16	0.858974	0.717949	1.0	0.812858
0	rf_lbp	0.773504	0.547009	1.0	0.696478
3	rf_vgg16	0.756410	0.512821	1.0	0.578626
2	rna_lbp	0.705128	0.410256	1.0	0.506758
1	svm_lbp	0.333333	0.000000	1.0	0.000000
5	rna_vgg16	0.333333	0.000000	1.0	0.000000

```

#Ordenando os resultados de teste por 'F1 Ponderada'
df_resultados_test = df_resultados_test.sort_values('F1 Ponderada', ascending=False)
df_resultados_test

```

	Tipo	Acurária	Sensibilidade Ponderada	Especificidade Ponderada	F1 Ponderada
4	svm_vgg16	0.928571	0.857143	0.951490	0.849591
3	rf_vgg16	0.861186	0.722372	0.905136	0.641626
2	rna_lbp	0.784367	0.568733	0.852303	0.578530
0	rf_lbp	0.780323	0.560647	0.853483	0.566296
1	svm_lbp	0.621294	0.242588	0.757412	0.094720
5	rna_vgg16	0.621294	0.242588	0.757412	0.094720

Conclusão:

Verificando as métricas obtidas tanto no conjunto de validação quanto no conjunto de testes percebe-se a melhor F1 Ponderada foi do Modelo SVM utilizando características extraídas da rede VGG16.

A F1 Ponderada é uma métrica importante para problemas de classificação com múltiplas classes, especialmente quando há um desbalanceamento entre as classes. Ela considera tanto a precisão quanto a sensibilidade (recall) e fornece uma medida balanceada do desempenho do modelo em todas as classes. Um valor de F1 ponderada próximo de 1 indica um bom equilíbrio entre precisão e recall.

Além disso, percebe-se que as demais métricas também seguem o mesmo padrão. A alta acurácia e a especificidade ponderada indicam que o modelo SVM/VGG16 possui uma boa capacidade de identificar corretamente as amostras negativas, sem perder muitas positivas, o que é crucial em contextos médicos para evitar falsos negativos.

Gerando os arquivos zip com os csvs vgg16 e lbp

import zipfile

def zipa(tipo):

arquivos = glob.glob(f'{train_dir}/*/*_{tipo}.csv') # + glob.glob(f'{val_dir}/*/*_vgg16.csv') +
glob.glob(f'{test_dir}/*/*_vgg16.csv')

nome_arquivo_zip = f'csv_{tipo}.zip'

with zipfile.ZipFile(nome_arquivo_zip, 'w') **as** zipf:

for arquivo **in** arquivos:

zipf.write(arquivo, arcname=arquivo.split('/')[-1]) *# arcname mantém apenas o nome do arquivo dentro do zip*

#lbp

zipa('lbp')

#vgg16

zipa('vgg16')

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. Treine só as novas camadas. Compare os treinos de 10 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:

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

```

# Módulo para imprimir os gráficos de treinamento de forma dinâmica.
!pip install livelossplot
# Baixando e descompactando datasets
!wget https://github.com/valeriow/IAA-PUB/raw/refs/heads/main/Test_Warwick.zip
!wget https://github.com/valeriow/IAA-PUB/raw/refs/heads/main/Train_Warwick.zip
!unzip -o Test_Warwick.zip
!unzip -o Train_Warwick.zip
import random
import glob
import zipfile
import os
from collections import defaultdict
import shutil
import math
import cv2
import numpy as np
from skimage.feature import local_binary_pattern
import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.applications.resnet50 import preprocess_input as preprocess_input_resnet
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.applications.vgg16 import preprocess_input as preprocess_input_vgg16
from tensorflow.keras.applications import VGG16
# Aqui utilizamos o mesmo código do exercício anterior para separar a base de treino apropriadamente
# Função para mover os arquivos para as respectivas pastas
def move_images(patients_list, destination_folder):
    if not os.path.exists(destination_folder):
        os.makedirs(destination_folder)

    for patient_id in patients_list:
        for img_path in patients_images[patient_id]:
            if img_path: # Verifica se há um caminho de imagem
                subdir = img_path.split('/')[-2]
                # Define o caminho de destino
                destination_path = os.path.join(destination_folder, subdir)
                # Criar o diretório de destino se não existir
                if not os.path.exists(destination_path):
                    os.makedirs(destination_path)
                # Move o arquivo para o diretório de destino
                # print(img_path, destination_path)
                shutil.copy(img_path, destination_path)

# Obtendo a lista de pacientes para fazer a separação da base de treinamento em treino e validação
# Obtém a lista de arquivos
all_train_images = glob.glob("/content/Train_4cls_amostra/*/*.png")

# Itera sobre a lista de arquivos obtendo o id do paciente
patients_images = defaultdict(list)
for img_path in all_train_images:
    filename = os.path.basename(img_path)
    patient_id = filename.split('_')[0] # Pega o XX (número do paciente)
    patients_images[patient_id].append(img_path)

# Obter a lista de pacientes (as chaves do defaultdict)
patients = list(patients_images.keys())

```



```

random.seed(42)
random.shuffle(patients)

# Dividir 80% para treino e 20% para validação
num_patients = len(patients)
num_train = math.ceil(0.8 * num_patients)
train_patients = patients[:num_train] # Primeiro 80% dos pacientes
val_patients = patients[num_train:] # Último 20% dos pacientes

# Diretórios para armazenar treino e validação
train_dir = '/content/train/'
val_dir = '/content/valid/'
test_dir = '/content/test/'

# Exclui diretórios antes da cópia se existirem
for dir in [train_dir, val_dir, test_dir]:
    if os.path.exists(dir):
        shutil.rmtree(dir)

# Copiar 80% dos pacientes para a pasta train
move_images(train_patients, train_dir)

# Copiar 20% dos pacientes para a pasta valid
move_images(val_patients, val_dir)

# Copiar os pacientes para a pasta test
shutil.copytree("/content/Test_4cl_amostra", f"{test_dir}")
'/content/test/'
classes = [classe.split("/")[-1] for classe in glob.glob("/content/Train_4cls_amostra/*")]
classes
['1', '0', '3', '2']
# Com data augmentation

BATCH_SIZE = 32 # quantidade de imagens criadas em cada ciclo

#Vgg16
train_generator_dataaug_vgg16 = ImageDataGenerator(
    rescale=1 / 255.,
    rotation_range=90,
    brightness_range=[0.1, 0.7],
    width_shift_range=0.5,
    height_shift_range=0.5,
    horizontal_flip=True,
    vertical_flip=True,
    preprocessing_function=preprocess_input_vgg16)

train_generator_vgg16 = ImageDataGenerator(rescale=1 / 255.,
    preprocessing_function=preprocess_input_vgg16)

valid_generator_vgg16 = ImageDataGenerator(rescale=1 / 255.,
    preprocessing_function=preprocess_input_vgg16)

test_generator_vgg16 = ImageDataGenerator(rescale=1 / 255.,
    preprocessing_function=preprocess_input_vgg16)

#REsnet
train_generator_dataaug_resnet = ImageDataGenerator(
    #rescale=1 / 255.,

```

```

        rotation_range=90,
        brightness_range=[0.1, 0.7],
        width_shift_range=0.5,
        height_shift_range=0.5,
        horizontal_flip=True,
        vertical_flip=True,
        preprocessing_function=preprocess_input_resnet)

train_generator_resnet = ImageDataGenerator(rescale=1 / 255.,
        preprocessing_function=preprocess_input_resnet)

valid_generator_resnet = ImageDataGenerator(rescale=1 / 255.,
        preprocessing_function=preprocess_input_resnet)

test_generator_resnet = ImageDataGenerator(rescale=1 / 255.,
        preprocessing_function=preprocess_input_resnet)

#VGG16
print("VGG16")
print("DataAug")
traingen_dataaug_vgg16 = train_generator_dataaug_vgg16.flow_from_directory("/content/train/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

print("No DataAug")
traingen_vgg16 = train_generator_vgg16.flow_from_directory("/content/train/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

print("Valid")
validgen_vgg16 = valid_generator_vgg16.flow_from_directory("/content/valid/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

print("Test")
testgen_vgg16 = test_generator_vgg16.flow_from_directory("/content/test/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

print("Resnet")
print("DataAug")
traingen_dataaug_resnet = train_generator_dataaug_resnet.flow_from_directory("/content/train/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,

```

```

        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

print("No DataAug")
traingen_resnet = train_generator_resnet.flow_from_directory("/content/train/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

print("Valid")
validgen_resnet = valid_generator_resnet.flow_from_directory("/content/valid/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

print("Test")
testgen_resnet = test_generator_resnet.flow_from_directory("/content/test/",
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
        classes=classes,
        shuffle=True,
        seed=42)

VGG16
DataAug
Found 476 images belonging to 4 classes.
No DataAug
Found 476 images belonging to 4 classes.
Valid
Found 117 images belonging to 4 classes.
Test
Found 371 images belonging to 4 classes.
Resnet
DataAug
Found 476 images belonging to 4 classes.
No DataAug
Found 476 images belonging to 4 classes.
Valid
Found 117 images belonging to 4 classes.
Test
Found 371 images belonging to 4 classes.
# Modelo VGG16 sem top
vgg16_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# não treinar os pesos existentes
for layer in vgg16_model.layers:
    layer.trainable = False

#Copia o modelo anterior para treinar com data augmentation
vgg16_model_dataaug = Model(inputs=vgg16_model.input, outputs=vgg16_model.output)
#Modelo ResNet50
# A opção include_top=False não inclui as camadas de aprendizado da rede original
# Utiliza os pesos treinados na base imagenet
resnet_model = ResNet50(input_shape=(224,224,3), weights='imagenet', include_top=False)

```

```

# não treinar os pesos existentes
for layer in resnet_model.layers:
    layer.trainable = False
# camadas próprias - você pode colocar mais se quiser
# A saída da resnet será a entrada da camada criada
x_tl = Flatten()(resnet_model.output)

# camada de classificação com as 4 classes utilizadas
prediction = Dense(len(classes), activation='softmax')(x_tl)

# Criação do Objeto Modelo (a parte da resnet + as camadas Fully connected criadas)
resnet_model = Model(inputs=resnet_model.input, outputs=prediction)

#Copia o modelo para treinar com data augmentation
resnet_model_dataaug = Model(inputs=resnet_model.input, outputs=resnet_model.output)
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

def train(training, validgen, model_tl, modelname):
    steps_per_epoch = training.samples // BATCH_SIZE
    val_steps = validgen.samples // BATCH_SIZE

    n_epochs = 100

    optimizer = RMSprop(learning_rate=0.0001)

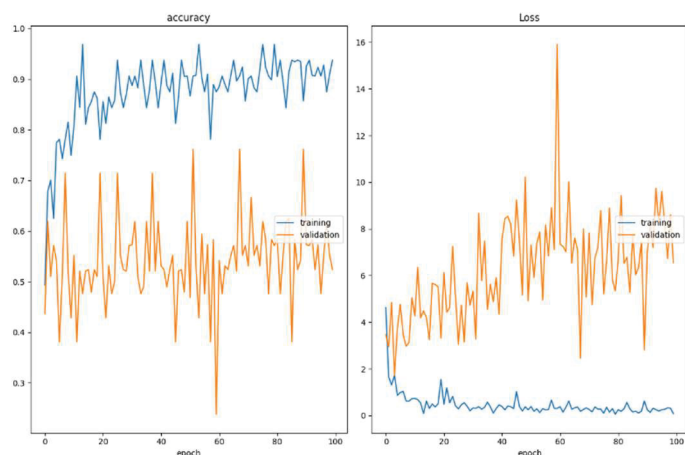
    model_tl.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])

    # Salva o modelo Keras após cada época, porém só o de melhor resultado
    checkpointer = ModelCheckpoint(filepath=f'img_model_{modelname}.weights.best.keras',
                                   verbose=1,
                                   save_best_only=True)

    # Para o treinamento para prevenir o overfitting
    # Não utilizei aqui, pois queria que rodasse todas as 30 épocas
    early_stop = EarlyStopping(monitor='val_loss',
                               patience=10,
                               restore_best_weights=True,
                               mode='min')

    # Treinamento do Modelo
    history_tl = model_tl.fit(training,
                              epochs=n_epochs,
                              steps_per_epoch=steps_per_epoch,
                              validation_data=validgen,
                              validation_steps=val_steps,
                              callbacks=[checkpointer, PlotLossesKeras()],
                              #callbacks=[early_stop, checkpointer, PlotLossesKeras()],
                              verbose=False)
    return model_tl, history_tl
# VGG16 com data augmentation
trained_vgg16_dataaug = train(training_dataaug_vgg16, validgen_vgg16, vgg16_model_dataaug,
"training_dataaug_vgg16")

```



accuracy

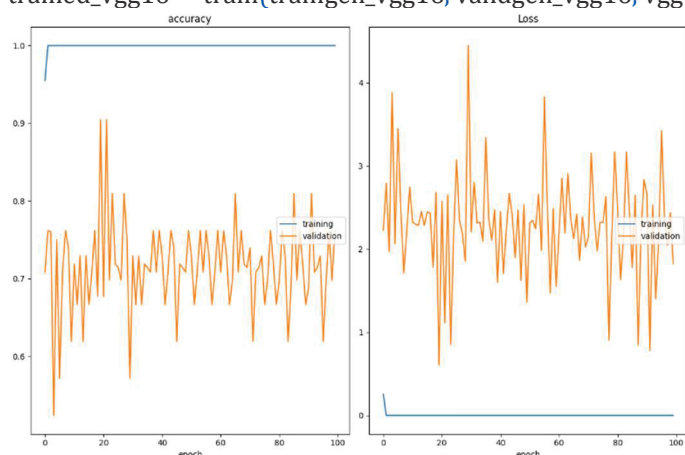
training	(min: 0.493, max: 0.969, cur: 0.938)
validation	(min: 0.238, max: 0.762, cur: 0.524)

Loss

training	(min: 0.054, max: 4.626, cur: 0.078)
validation	(min: 1.728, max: 15.903, cur: 6.541)

VGG16 sem data augmentation

trained_vgg16 = train(traugen_vgg16, validgen_vgg16, vgg16_model, "traugen_vgg16")



accuracy

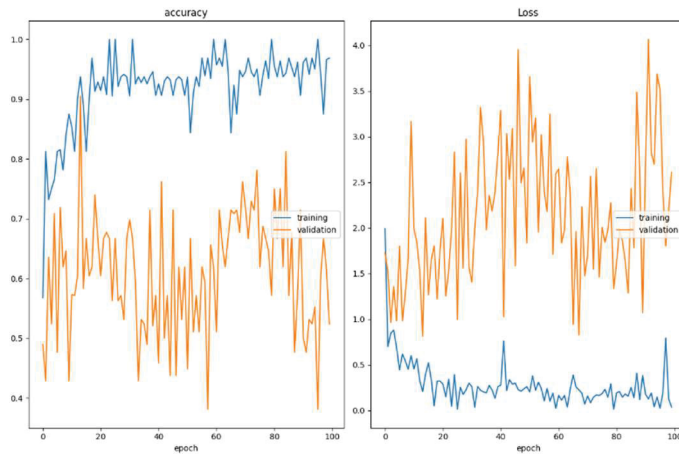
training	(min: 0.955, max: 1.000, cur: 1.000)
validation	(min: 0.524, max: 0.905, cur: 0.762)

Loss

training	(min: 0.000, max: 0.255, cur: 0.000)
validation	(min: 0.607, max: 4.448, cur: 1.821)

ResNet com data augmentation

trained_resnet_dataaug = train(traugen_dataaug_resnet, validgen_resnet, resnet_model_dataaug, "traugen_dataaug_resnet")



accuracy

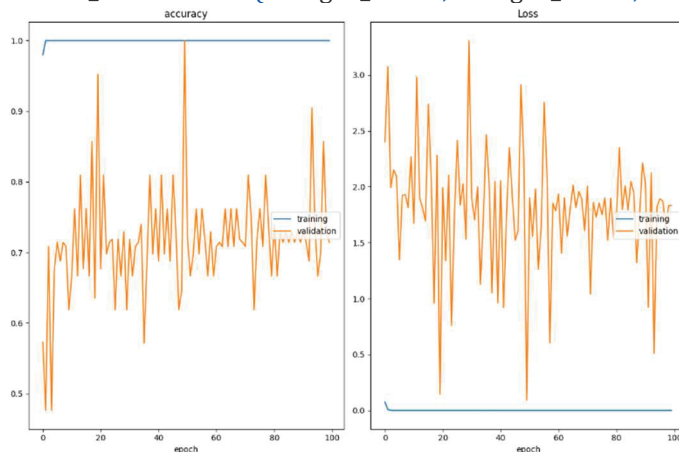
training	(min: 0.568, max: 1.000, cur: 0.969)
validation	(min: 0.381, max: 0.905, cur: 0.524)

Loss

training	(min: 0.011, max: 1.990, cur: 0.036)
validation	(min: 0.811, max: 4.066, cur: 2.611)

ResNet sem data augmentation

trained_resnet = train(training_resnet, validation_resnet, resnet_model, "training_resnet")



accuracy

training	(min: 0.980, max: 1.000, cur: 1.000)
validation	(min: 0.476, max: 1.000, cur: 0.714)

Loss

training	(min: 0.000, max: 0.074, cur: 0.000)
validation	(min: 0.091, max: 3.307, cur: 1.833)

from sklearn.metrics import confusion_matrix

def calculate_metrics(cm):

Calcular a matriz de confusão

#cm = confusion_matrix(y_val, val_predictions)

Inicializar contadores para micro F1-Score

TP_total = 0

FP_total = 0

FN_total = 0

TN_total = 0

f1_scores = []

sensibilities = []

specificities = []

Número de classes

```

num_classes = cm.shape[0]

# Calcular precisão, recall e F1-Score para cada classe
for i in range(num_classes):
    TP = cm[i, i]
    FP = cm[:, i].sum() - TP # Falsos positivos
    FN = cm[i, :].sum() - TP # Falsos negativos
    TN = cm.sum() - (TP + FP + FN) # Verdadeiros negativos

    # Calcular Precisão e Recall
    precision = TP / (TP + FP) if (TP + FP) > 0 else 0
    recall = TP / (TP + FN) if (TP + FN) > 0 else 0
    sensibilidade = TP / (TP + FN) if (TP + FN) > 0 else 0
    especificidade = TN / (TN + FP) if (TN + FP) > 0 else 0

    f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) > 0 else 0

    # Acumular valores
    TP_total += TP
    FP_total += FP
    FN_total += FN
    TN_total += TN

    f1_scores.append(f1)
    sensibilities.append(sensibilidade)
    specificities.append(especificidade)

accuracy = (TP_total + TN_total) / (TP_total + TN_total + FP_total + FN_total) if (TP_total + TN_total +
FP_total + FN_total) > 0 else 0

n_classes = [cm[i, :].sum() for i in range(num_classes)] # Número de instâncias por classe

# Cálculo do F1-Score Micro
#precision_micro = TP_total / (TP_total + FP_total) if (TP_total + FP_total) > 0 else 0
#recall_micro = TP_total / (TP_total + FN_total) if (TP_total + FN_total) > 0 else 0
#f1_micro = 2 * (precision_micro * recall_micro) / (precision_micro + recall_micro) if (precision_micro +
recall_micro) > 0 else 0

# Cálculo do F1-Score Weighted
f1_weighted = sum(f1 * n for f1, n in zip(f1_scores, n_classes)) / sum(n_classes) if sum(n_classes) > 0
else 0

# Sensibilidade ponderada
sensitivity_weighted = sum(sens * n for sens, n in zip(sensibilities, n_classes)) / sum(n_classes) if
sum(n_classes) > 0 else 0

# Especificidade ponderada
specificity_weighted = sum(spec * n for spec, n in zip(specificities, n_classes)) / sum(n_classes) if
sum(n_classes) > 0 else 0

return accuracy, sensibility_weighted, specificity_weighted, f1_weighted
trained_resnet[0].predict(testgen_resnet)
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs`
can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as t
hey will be ignored.
self.warn_if_super_not_called()
12/12 ————— 11s 565ms/step
array([[9.87282634e-01, 1.06999648e-18, 8.83815066e-09, 1.27173057e-02],

```

```

[6.90599024e-01, 4.13748193e-15, 2.10919060e-09, 3.09400976e-01],
[7.84158647e-01, 2.26755489e-07, 1.25245325e-08, 2.15841204e-01],
...,
[8.28316353e-27, 6.70703358e-29, 1.00000000e+00, 3.06785082e-19],
[1.60936883e-13, 1.87543801e-20, 6.52806702e-05, 9.99934673e-01],
[2.23787396e-15, 9.99999523e-01, 1.54875877e-15, 4.57421493e-07]],
dtype=float32)
traingen_dataaug_resnet
<keras.src.legacy.preprocessing.image.DirectoryIterator at 0x7ab968c334c0>
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_fscore_support, accuracy_score

def display_results(y_true, y_preds, class_labels):

    # Converte para labels das classes para valores discretos se elas forem valores contínuos
    # Assuming y_preds contains probabilities or raw scores
    y_preds = np.argmax(y_preds, axis=1)

    # Ensure both y_true and y_preds are of the same data type (e.g., int)
    y_true = y_true.astype(int)
    y_preds = y_preds.astype(int)

    results = pd.DataFrame(precision_recall_fscore_support(y_true, y_preds),
                           columns=class_labels).T

    results.rename(columns={0: 'Precision', 1: 'Recall',
                           2: 'F-Score', 3: 'Support'}, inplace=True)

    results.sort_values(by='F-Score', ascending=False, inplace=True)
    global_acc = accuracy_score(y_true, y_preds)

    print("Acurácia: {:.2f}%".format(global_acc*100))
    return results

def plot_predictions(y_true, y_preds, test_generator, class_indices):

    fig = plt.figure(figsize=(20, 10))
    for i, idx in enumerate(np.random.choice(test_generator.samples, size=20, replace=False)):
        ax = fig.add_subplot(4, 5, i + 1, ticks=[], yticks=[])
        ax.imshow(np.squeeze(test_generator[idx]))
        pred_idx = y_preds[idx]
        true_idx = y_true[idx]

        plt.tight_layout()
        ax.set_title("{}\n{}".format(class_indices[pred_idx], class_indices[true_idx]),
                     color=("green" if pred_idx == true_idx else "red"))

def plot_predictions(y_true, y_preds, test_generator, class_indices):
    fig = plt.figure(figsize=(20, 10))

    # Get the actual number of samples in the test generator
    num_samples = len(test_generator)

    # Ensure you only select indices within the valid range
    indices = np.random.choice(np.arange(num_samples), size=min(num_samples, 20), replace=False)

    for i, idx in enumerate(indices): # Use the valid indices here

```



```

ax = fig.add_subplot(4, 5, i + 1, xticks=[], yticks=[])

# Accessing data from the generator might require a different approach
# depending on how your generator is implemented.
# If your generator yields (image, label) pairs, you can use:
# image, _ = test_generator[idx]
# ax.imshow(np.squeeze(image))

# However, if you're using ImageDataGenerator.flow_from_directory,
# you need to iterate through the generator to get data:
batch_index = idx // test_generator.batch_size
index_within_batch = idx % test_generator.batch_size

# Iterate to the desired batch
for _ in range(batch_index + 1):
    batch_x, batch_y = test_generator.next()

ax.imshow(np.squeeze(batch_x[index_within_batch]))

pred_idx = np.argmax(y_preds[idx]) # Get the predicted class index
true_idx = y_true[idx]

plt.tight_layout()
ax.set_title("{}\n{}".format(class_indices[pred_idx], class_indices[true_idx]),
            color=("green" if pred_idx == true_idx else "red"))
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score, confusion_matrix

def calculate_metrics(cm):
    # Converte para labels das classes para valores discretos se elas forem valores contínuos
    # y_preds = np.argmax(y_preds, axis=1)

    # Garante que ambos y_true e y_preds são do mesmo tipo de dados (e.g., int)
    # Inicializa listas para armazenar as métricas por classe
    precisions = []
    recalls = []
    f1_scores = []
    specificities = []
    supports = []

    TP_total = 0
    FP_total = 0
    FN_total = 0
    TN_total = 0

    # Número de classes
    num_classes = cm.shape[0]

    # Calcula métricas para cada classe
    for i in range(num_classes):
        tp = cm[i, i]
        fn = np.sum(cm[i, :]) - tp
        fp = np.sum(cm[:, i]) - tp
        tn = np.sum(cm) - (tp + fn + fp)

        # Evita divisão por zero
        precision = tp / (tp + fp) if (tp + fp) > 0 else 0

```

```

recall = tp / (tp + fn) if (tp + fn) > 0 else 0
specificity = tn / (tn + fp) if (tn + fp) > 0 else 0
f1 = 2 * precision * recall / (precision + recall) if (precision + recall) > 0 else 0
support = np.sum(cm[i, :]) # Número de amostras reais da classe i

# Acumular valores
TP_total += tp
FP_total += fp
FN_total += fn
TN_total += tn

# Adiciona as métricas calculadas às listas
precisions.append(precision)
recalls.append(recall)
f1_scores.append(f1)
specificities.append(specificity)
supports.append(support)

# Calcula métricas ponderadas
f1_weighted = np.average(f1_scores, weights=supports)
recall_weighted = np.average(recalls, weights=supports)
specificity_weighted = np.average(specificities, weights=supports)
accuracy = (TP_total + TN_total) / (TP_total + TN_total + FP_total + FN_total) if (TP_total + TN_total +
FP_total + FN_total) > 0 else 0

# Acurácia global

print("\nAcurácia: {:.2f}%".format(accuracy * 100))
print(f"F1-Score (Weighted): {f1_weighted:.2f}")
print(f"Sensibilidade (Recall Weighted): {recall_weighted:.2f}")
print(f"Especificidade (Weighted): {specificity_weighted:.2f}")

return accuracy, recall_weighted, specificity_weighted, f1_weighted

def metricas(tipo, df_resultados, y_true, y_preds):
    # Converte para labels das classes para valores discretos se elas forem valores contínuos
    y_preds = np.argmax(y_preds, axis=1)

    # Garante que ambos y_true e y_preds são do mesmo tipo de dados (e.g., int)
    y_true = y_true.astype(int)
    y_preds = y_preds.astype(int)

    cm = confusion_matrix(y_true, y_preds)
    accuracy, sensibility_weighted, specificity_weighted, f1_weighted = calculate_metrics(cm)
    data = { 'Tipo': tipo,
             'Acurácia': [accuracy],
             'Sensibilidade Ponderada': [sensibility_weighted],
             'Especificidade Ponderada': [specificity_weighted],
             'F1 Ponderada': [f1_weighted]}

    df_resultados = pd.concat([df_resultados, pd.DataFrame(data)], ignore_index=True)
    return df_resultados

# Carrega os melhores modelos
resnet_model.load_weights('/content/img_model_traingen_resnet.weights.best.keras')
resnet_model_dataaug.load_weights('/content/img_model_traingen_dataaug_resnet.weights.best.keras')

vgg16_model.load_weights('/content/img_model_traingen_vgg16.weights.best.keras')
vgg16_model_dataaug.load_weights('/content/img_model_traingen_dataaug_vgg16.weights.best.keras')
#Preparando os dataframes para receber os resultados

```

```
df_resultados_test = pd.DataFrame()
```

```
# Calculando as métricas
```

```
#Resnet
```

```
predicted_classes = resnet_model.predict(testgen_resnet)
class_indices = traingen_resnet.class_indices
true_classes = testgen_resnet.classes
#labels = traingen_resnet.class_indices
df_resultados_test = metricas("resnet", df_resultados_test, true_classes, predicted_classes)
```

```
#Resnet Data Augmentation
```

```
predicted_classes = resnet_model_dataaug.predict(testgen_resnet)
class_indices = traingen_resnet.class_indices
true_classes = testgen_resnet.classes
df_resultados_test = metricas("resnet_dataaug", df_resultados_test, true_classes, predicted_classes)
```

```
#Vgg16 Data
```

```
predicted_classes = vgg16_model.predict(testgen_vgg16)
class_indices = traingen_vgg16.class_indices
true_classes = testgen_vgg16.classes
df_resultados_test = metricas("vgg16", df_resultados_test, true_classes, predicted_classes)
```

```
#Vgg16 Data Augmentation
```

```
predicted_classes = vgg16_model_dataaug.predict(testgen_vgg16)
class_indices = traingen_vgg16.class_indices
true_classes = testgen_vgg16.classes
df_resultados_test = metricas("vgg16_dataaug", df_resultados_test, true_classes, predicted_classes)
```

```
12/12 ————— 3s 238ms/step
```

Acurácia: 62.80%

F1-Score (Weighted): 0.25

Sensibilidade (Recall Weighted): 0.26

Especificidade (Weighted): 0.75

```
12/12 ————— 11s 688ms/step
```

Acurácia: 63.21%

F1-Score (Weighted): 0.26

Sensibilidade (Recall Weighted): 0.26

Especificidade (Weighted): 0.75

/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.

```
self.warn_if_super_not_called()
```

```
12/12 ————— 13s 1s/step
```

Acurácia: 64.29%

F1-Score (Weighted): 0.28

Sensibilidade (Recall Weighted): 0.29

Especificidade (Weighted): 0.76

```
12/12 ————— 3s 181ms/step
```

Acurácia: 62.67%

F1-Score (Weighted): 0.25

Sensibilidade (Recall Weighted): 0.25

Especificidade (Weighted): 0.75

```
#Ordenando os resultados de teste por 'F1 Ponderada'
```

```
df_resultados_test = df_resultados_test.sort_values('F1 Ponderada', ascending=False)
df_resultados_test
```

	Tipo	Acurária	Sensibilidade Ponderada	Especificidade Ponderada	F1 Ponderada
2	vgg16	0.642857	0.285714	0.761511	0.284902
1	resnet_dataaug	0.632075	0.264151	0.753735	0.264135
0	resnet	0.628032	0.256065	0.750855	0.254400
3	vgg16_dataaug	0.626685	0.253369	0.750717	0.252442

Conclusão

O modelo que melhor métricas apresentou no conjunto de teste foi o VGG16 sem Data Augmentation. Logo após temos o Resnet50 com Data Augmentation. Seguido pelo Resnet50 Sem Data Augmentation e por último, mas com métricas muito semelhantes, o VGG16 com Data Augmentation.

Os valores das métricas foi muito baixo, o que atribuímos ao dataset possuir poucos exemplos de cada classe para cada paciente, assim o treinamento e validação durante o treinamento não foram adequados. O dataset de treinamento possuía 20 pacientes apenas e dividindo em 80%/20%, conforme o que solicitava o enunciado do trabalho, acabamos com 16 pacientes para treinamento e 4 para validação. Como fizemos a separação aleatoriamente, ficamos com apenas 2 classes de exemplo no dataset de validação. É possível que os resultados melhorassem um pouco se fizemos uma separação de forma que os 4 pacientes do dataset de validação fossem um de cada classe. Infelizmente não tivemos tempo para isso e os nossos créditos gratuitos de uso de GPU no Google Colab terminaram.

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

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

RESUMO

O uso das novas tecnologias de Inteligência Artificial (IA), como o ChatGPT, apresenta um grande salto na evolução da chamada sociedade informacional, possibilitando acesso rápido à informação e conhecimento sobre diversos temas. É natural que uma ferramenta tão poderosa fosse também utilizada de maneira que levasse a questionamentos éticos e legais por diversas partes. Esse tipo de debate deve se tornar a tônica das futuras discussões envolvendo o uso da IA.

O presente trabalho apresenta uma reflexão sobre as implicações legais e éticas na aplicação do ChatGPT em alguns cenários específicos como na educação, atendimento ao cliente, criação de conteúdo, privacidade, consentimento de dados e tomada de decisões. Além de buscar soluções, o artigo visa ser um chamamento à reflexão sobre os valores primordiais das interações sociais humanas e até que ponto os ganhos devem justificar os meios utilizados para o seu atingimento. Os benefícios também não devem ser ignorados, mas é preciso entender as implicações, onde, como, e a que custo queremos chegar obtê-los. Como toda nova ferramenta tecnológica, é natural que surjam essas questões e é importante que elas sejam debatidas de forma ampla e aberta para que sejamos capazes de entender os caminhos percorridos e propor eventuais ajustes ao longo desses caminhos.

Palavras-chave: ChatGPT, ética, inteligência artificial, vieses algorítmicos, privacidade.

ABSTRACT

The use of new Artificial Intelligence (AI) technologies, such as ChatGPT, represents a significant leap in the evolution of the so-called information society, enabling rapid access to information and knowledge on a wide range of topics. It is natural that such a powerful tool would also be used in ways that raise ethical and legal questions from various parties. This type of debate should become the focus of future discussions involving the use of AI.

This paper presents a reflection on the legal and ethical implications of applying ChatGPT in specific scenarios such as education, customer service, content creation,

privacy, data consent, and decision-making. In addition to seeking solutions, the article aims to be a call for reflection on the fundamental values of human social interactions and to what extent the benefits should justify the means used to achieve them. The benefits should not be ignored either, but it is necessary to understand the implications, where, how, and at what cost we want to achieve them.

As with any new technological tool, it is natural that these questions arise, and it is important that they are debated widely and openly so that we are able to understand the paths taken and propose any necessary adjustments along the way.

Keywords: ChatGPT, ethics, artificial intelligence, algorithmic biases, privacy.

1. INTRODUÇÃO

A Inteligência Artificial (IA) é cada vez mais presente na sociedade, impactando diversos setores, como o trabalho, a educação e a comunicação. O ChatGPT foi uma das ferramentas de IA que mais tiveram destaque nos últimos anos por sua capacidade de gerar textos coerentes, contextualizados e com informações relevantes.

Diversas atividades manuais e demoradas, agora podem ser executadas de forma rápida e relativamente automatizada com o ChatGPT. Mas todo benefício traz consigo potenciais problemas. Uma das preocupações, cada vez mais relevantes, é quanto aos impactos do seu uso em questões legais e éticas dentro das sociedades. Como lidar com assuntos que envolvem privacidade, direitos autorais, disseminação de falsas informações, manipulação da opinião pública, entre outros? Apesar desses assuntos não serem novidade, a capacidade de geração e disseminação em escala industrial a um custo relativamente baixo, é que os torna ainda mais relevantes nesse novo cenário.

O presente trabalho tem como objetivo discutir algumas implicações éticas e legais do uso do ChatGPT em diferentes cenários e em estudos de caso específicos, tentando identificar questões relevantes e eventuais soluções e posicionamentos para lidar com essas novas situações viabilizadas pelo atual estágio de desenvolvimento da IA. Esse tipo de reflexão é de extrema importância para que a sociedade busque

utilizar essas novas tecnologias de forma responsável e que gere mais benefícios do que prejuízos no seu uso.

2. CHATGPT NA EDUCAÇÃO

O ChatGPT na educação pode facilitar a aprendizagem e a pesquisa, mas, se mal utilizado, pode impactar em questões importantes como plágio e a falta de preocupação com o desenvolvimento do pensamento analítico e crítico. Segundo CHEN(2019):

"A inteligência artificial na educação tem o potencial de transformar a forma como os estudantes aprendem, permitindo que os professores se concentrem em atividades de alto valor agregado, como a criação de currículos e o planejamento de aulas, enquanto a IA lida com tarefas mais repetitivas e administrativas."

Mais do que olhar para os estudantes como potenciais fraudadores, é preciso repensar a educação e seu papel, incluindo as formas de avaliação. Isso inclui ensinar boas práticas e técnicas de uso de ferramentas como o ChatGPT. Em vez de solicitar que o ChatGPT gere um artigo completo sobre determinado assunto, o aluno pode alimentar a ferramenta com fontes bibliográficas relevantes sobre o assunto, solicitar resumos e fazer questionamentos sobre o conteúdo para que vá desenvolvendo sua capacidade de análise crítica. Isso não exclui a necessidade de ler trechos das fontes bibliográficas e a pesquisa em outras fontes para confirmar as informações. Do ponto de vista das avaliações, o surgimento dessas ferramentas acaba, paradoxalmente, valorizando cada vez mais o ser humano e as interações humanas, pois para termos uma avaliação um pouco mais efetiva, enriquecedora e isonômica, cada vez mais será necessária a interação humana, para que o estudante seja capaz de demonstrar realmente o que absorveu. E se ele for capaz de demonstrar isso sem grandes aparatos tecnológicos, pouco importa se durante o processo fez ou não uso desses aparatos, pois os objetivos de desenvolver autonomia e capacidade de pensamento crítico foram atingidos.

Analisando do ponto de vista do custo, as interações humanas de qualidade na educação têm um custo relativamente alto e talvez não acessível a todos. Por outro lado, o custo de criar cursos e materiais didáticos autoinstrucionais e automatizados,

com pouca interação humana, inclusive nas avaliações, deve acabar sendo reduzido pelo uso de ferramentas, como o ChatGPT, por parte dos criadores de conteúdo e professores. Nesses casos, cabe ao estudante e ao seu código de moral fazer um bom uso das ferramentas de IA, para que sua experiência resulte no aprimoramento de suas capacidades e não somente em um certificado, com valor questionável. De certa forma, é como se a avaliação do estudante ocorra, de fato, na utilização dos conhecimentos, seja na vida profissional, numa entrevista de emprego ou na sua rotina diária. O grande ganho desse tipo de educação é o acesso cada vez mais amplo ao conhecimento e, de certa forma, paradoxalmente, uma importância menor dos certificados. As certificações profissionais, obtidas em ambiente controlado e com questões e situações práticas elaboradas por especialistas, podem preencher essa lacuna e acabarem sendo uma prova mais contundente de domínio de alguns tipos de conhecimento, o que já é um movimento em andamento, especialmente no âmbito das TICs. No final, tudo é uma questão de rever os objetivos e desmistificar o uso das novas tecnologias. Mal comparando, as calculadoras eletrônicas não substituíram os matemáticos e engenheiros e não foram abolidas das disciplinas. Talvez em uma ou outra disciplina, de acordo com o objetivo, o uso delas não seja encorajado, como para uma criança que aprende as primeiras operações aritméticas, mas quando apropriado, não há prejuízo, pelo contrário, acelera a realização de cálculos manuais em favor do aprendizado de novos conceitos e relações.

Durante as aulas síncronas da presente disciplina, um colega mencionou que “conversou” com o ChatGPT sobre o Mito da Caverna de Platão como analogia às bolhas informacionais e Fake News da atualidade e isso possibilitou que ele argumentasse sobre diversos pontos de vista durante a aula síncrona sem o auxílio, naquele exato momento, do ChatGPT, mostrando que a ferramenta possibilitou que ele refletisse sobre diversos aspectos de forma crítica, que é exatamente o que a educação objetiva. Muito diferente seria se ele tivesse apenas solicitado um artigo ao ChatGPT e sem ler, sem refletir sobre ele, fizesse a entrega. Nosso grupo frequentemente discute alguns conceitos e dúvidas e com muita naturalidade compartilhamos nosso processo de perguntas e respostas com ferramentas de IA, como o ChatGPT, para subsidiar nossas convicções e argumentos, tudo isso, sempre buscando construir uma visão crítica e reflexiva.

3. CHATGPT NO ATENDIMENTO AO CLIENTE

O ChatGPT também pode melhorar a eficiência do atendimento ao cliente, mas pode levar à desumanização das interações e à perda de empregos. De acordo com o relatório da McKinsey (2023), “Um em cada 16 trabalhadores poderá ter de mudar de profissão até 2030.” e “O crescimento do emprego estará mais concentrado em empregos de alta qualificação (por exemplo, nas áreas de saúde ou ciência, tecnologia, engenharia e matemática [STEM]), enquanto empregos de média e baixa qualificação (como serviços de alimentação, trabalho de produção ou funções de suporte de escritório) diminuirão”. Novamente vemos na história um momento disruptivo, com grande ganho de produtividade pelo uso de ferramentas tecnológicas e isso representa um problema social, pois talvez nem todos os cidadãos estejam preparados para assumirem outras atividades laborais, que tendem a ficar cada vez mais complexas e especializadas. Isso remete à educação e a necessidade de sua reformulação e universalização do acesso. Como abordado anteriormente, a tendência é que haja uma redução no custo de produção de conteúdos educacionais, o que é importante para uma ampliação do acesso e também para que seja possível que a educação acompanhe o ritmo cada vez mais acelerado de desenvolvimento tecnológico.

Sobre a desumanização das interações, já existem iniciativas de IA para levar em consideração questões mais sutis como as emoções nas interações, mas à medida que seja possível resolver alguma demanda rapidamente, essa questão acaba sendo cultural e as necessidades de interação mais humana podem ser satisfeitas de outras formas. Provavelmente, cada vez menos pessoas trocariam as facilidades de resolver suas demandas bancárias por meio de aplicativos em favor de passar horas em uma fila pela simples interação social.

Uma questão importante no atendimento automatizado que vem sendo discutida é a possibilidade de intervenção humana a qualquer momento, para seja possível a resolução dos problemas, já que as ferramentas não têm toda autonomia e capacidade para resolver todo tipo de situação. Muitas empresas têm apostado em solução automatizadas de atendimento, restringido totalmente o acesso a operadores humanos, mas não conseguem oferecer a solução dos problemas com essas

soluções e isso deixa os usuários, que demandam alguma solução, reféns dos chatbots e suas limitações naturais ou impostas intencionalmente pelos seus proprietários. É preciso tomar cuidado para que isso não viole direitos fundamentais dos cidadãos.

4. CHATGPT NA CRIAÇÃO DE CONTEÚDO

Na criação de conteúdo, o ChatGPT pode auxiliar, mas também pode ser usado para gerar conteúdo enganoso ou tendencioso. Conforme apontado por Radford et al. (2019), "É importante que os modelos de linguagem sejam treinados com dados diversos e representativos para evitar a geração de conteúdo discriminatório ou ofensivo.". Nesse aspecto o uso do aprendizado por reforço humano tem grande importância, pois visa balizar em termos de objetivos e também em termos éticos as saídas dos modelos de IA para que não ofendam e discriminem pessoas e grupos. Esse tipo de ajuste é muito sensível ao momento e ao local, assim como pode ser bastante polêmico, assim como é a ética. Outra questão ética e legal que surge é sobre a responsabilização sobre a geração de conteúdos que firam direitos fundamentais das pessoas e grupos. Quem é o responsável? Quem treinou o modelo? Quem conduziu o aprendizado por reforço humano? Quem escreveu o "prompt"? Quem divulgou? São questões que estarão cada vez mais presentes nos debates da sociedade e deverão ter alguma regulamentação mínima para balizar o uso dessas ferramentas.

Como apontado por Floridi (2020): "A inteligência artificial generativa, como o ChatGPT, pode ser usada para criar notícias falsas e outros tipos de conteúdo enganoso em escala industrial." Com a proximidade das eleições municipais esse ano, essa questão ganha maior relevância, pois potenciais efeitos da geração de conteúdo falso, como Deep Fakes e Fake News, podem ter um impacto sobre uma cidade, um estado ou até mesmo um país, como já se presenciou nas eleições norte-americanas em 2016, com o escândalo da assessoria Cambridge Analytica, posteriormente na eleição norte-americana seguinte e até mesmo nas eleições brasileiras.

5. CHATGPT E A PROTEÇÃO AOS DADOS PESSOAIS

Recentemente no Brasil saiu uma série de reportagens de Rossi (2023) sobre a comercialização dos dados de compras de clientes de uma grande rede de farmácias nacional que, para obter dados sensíveis, se utiliza de incentivos como descontos em medicamentos, que são questionáveis. Em nenhum momento fica claro para o cliente que esses descontos são condicionados ao uso e comercialização de seus dados pessoais e de consumo. Essa rede de farmácias possui até mesmo uma empresa própria para a coleta, transformação e comercialização desses dados, dado o valor comercial dessas informações. Com a utilização de modelos de IA que codificam de uma forma opaca as relações entre os dados que alimentam o treinamento desses modelos, fica até difícil estabelecer essa relação entre a saída e o dado que alimentou e originou essa saída, evidenciando que talvez seja necessário algum tipo de regulamentação sobre a guarda dos dados originais utilizados para o treinamento para seja possível auditar eventuais violações de privacidade e até mesmo de direitos autorais, como muito tem se discutido atualmente, onde autores e editoras de livros têm denunciado o uso não autorizado de suas obras para o treinamento desses modelos.

O uso de dados pessoais para treinar o ChatGPT traz reflexões importantes sobre privacidade, propriedade e consentimento de uso de dados. É essencial que os cidadãos tenham controle sobre seus próprios dados e que fique claro o tipo de uso que será feito daqueles dados que, eventualmente, haja consentimento de uso. Além de algum mecanismo que garanta que os dados serão utilizados exclusivamente para os fins acordados.

6. CHATGPT NA TOMADA DE DECISÕES

A tomada de decisões automatizada com o apoio do ChatGPT também é um tema que requer cuidado e atenção no que se refere à responsabilidade e transparência. À medida que a sociedade deposita muita confiança nessas ferramentas, elas se tornam grandes oráculos modernos, o que leva muitos a acatarem decisões como as melhores possíveis e como verdades absolutas,

enquanto sabe-se que essas ferramentas não são perfeitas e estão suscetíveis a falhas e a perpetuar vieses já existentes e indesejáveis. Nesse aspecto, é importante buscar garantir que essas falhas e vieses sejam minimizados e monitorados para evitar prejuízo para pessoas e grupos. A intervenção humana deverá estar disponível a qualquer momento para evitar que os prejuízos sejam amplificados. Esse tipo de debate remete também à discussão sobre a manipulação na criação de conteúdo, nesse caso especificamente, pessoas com influência no projeto dessas ferramentas de IA, podem criar vieses intencionais para obterem algum tipo de benefício. Assim, é importante entender como essas ferramentas são projetadas e como funcionam para que não haja mistificações sobre elas e para seja sempre possível redirecionar as saídas à medida em que se comportam de forma antagônica à ética e a lei vigente naquele momento e local.

7. CONSIDERAÇÕES FINAIS

Para lidar com os desafios éticos do ChatGPT, é necessário adotar uma abordagem responsável em seu desenvolvimento e uso. Algumas soluções incluem:

- **Transparência:** Os usuários devem ser informados sobre como o modelo funciona e quais dados são utilizados para treiná-lo.
- **Responsabilidade:** Os desenvolvedores devem ser responsabilizados pelos resultados do modelo e tomar medidas para mitigar os riscos.
- **Diversidade e Inclusão:** É importante garantir que o modelo seja treinado com dados representativos de diferentes grupos sociais para evitar vieses e discriminação.
- **Educação e Conscientização:** É fundamental educar os usuários sobre os benefícios e riscos do ChatGPT para que possam tomar decisões informadas sobre seu uso.

O ChatGPT tem o potencial de revolucionar diversas áreas, mas seu uso também apresenta desafios éticos significativos. Ao adotar uma abordagem responsável, podemos garantir que essa tecnologia seja utilizada de forma ética e benéfica para a sociedade. É crucial que pesquisadores, desenvolvedores,

legisladores e a sociedade em geral trabalhem juntos para encontrar soluções que promovam o uso responsável e ético do ChatGPT.

REFERÊNCIAS

- CHEN, B.; LIU, H.; ZHANG, J. Integrating artificial intelligence into educational technology research and development. New Jersey: Educational Technology Research and Development, 2019.
- Floridi, L. (2020). Inteligência artificial: Uma breve introdução. Rio de Janeiro: Zahar.
- MCKINSEY & COMPANY. What is the future of work? McKinsey Explainers. Disponível em: <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-the-future-of-work>. Acesso em: 30 jun. 2024.
- O'Neil, C. (2016). Weapons of math destruction: How big data increases inequality and threatens democracy. New York: Crown.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog.
- Rossi, Amanda. Vitamina usada para tentar engravidar pode direcionar até anúncio de carro. UOL, São Paulo, 1 set. 2023. Disponível em: <https://noticias.uol.com.br/saude/ultimas-noticias/redacao/2023/09/01/remedio-para-engravidar-pode-direcionar-ate-propaganda-de-carro.htm?cmpid=copiaecola>. Acesso em: 30 jun. 2024.
- Zuboff, S. (2019). The age of surveillance capitalism: The fight for a human future at the new frontier of power. New York: PublicAffairs.

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 OBJETIVO

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

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

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

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

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

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

2 ORIENTAÇÕES ADICIONAIS

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

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

Qual o objetivo do estudo descrito pelo artigo?	Qual o problema/oportunidade/situação que levou à necessidade de realização desse estudo?	Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?	Quais os principais resultados obtidos pelo estudo?
O objetivo do estudo descrito pelo artigo é entender como as ferramentas de Auto-ML são usadas na prática atualmente. Ele busca identificar as necessidades dos usuários, os benefícios e deficiências das ferramentas de Auto-ML existentes, assim como os papéis do ser humano e da automação nos fluxos de trabalho de aprendizado de máquina. A partir da sua análise, o estudo visa fornecer recomendações para o desenvolvimento futuro de ferramentas de Auto-ML, com destaque para a importância de uma parceria produtiva entre os usuários e a ferramenta, em vez de buscar a automação completa.	<p>A necessidade de realizar este estudo surgiu da crescente popularidade das ferramentas de Auto-ML, que prometem tornar o aprendizado de máquina mais acessível e reduzir o esforço manual envolvido na aplicação de ML.</p> <p>Apesar dessa promessa, há uma percepção de que o aprendizado de máquina ainda é uma "arte obscura", acessível apenas para poucos especialistas e envolvendo um processo trabalhoso de tentativa e erro.</p> <p>O estudo foi motivado pela necessidade de entender como essas ferramentas de Auto-ML são realmente usadas na prática, quais são os benefícios e deficiências percebidos pelos usuários, assim como a automação e a intervenção humana podem ser equilibradas nos fluxos de trabalho de ML.</p> <p>A oportunidade identificada é a de melhorar o projeto das ferramentas de Auto-ML para apoiar melhor os usuários, promovendo uma colaboração produtiva entre humanos e automação, em vez de buscar a automação completa.</p>	<p>Os autores usaram uma metodologia qualitativa para obter e analisar as informações do estudo.</p> <p>Recrutamento dos participantes: onde os participantes foram recrutados por meio de mensagens em listas de discussão, canais do Slack, conexões pessoais e redes sociais. Os selecionados tinham experiência real no uso de ferramentas de Auto-ML em casos de uso do mundo real.</p> <p>Perfil dos participantes: foram entrevistados 16 participantes com experiência em Auto-ML, abrangendo uma variedade de domínios de aplicação e níveis de habilidade em ML.</p>	<p>Os principais resultados obtidos pelo estudo são:</p> <p>Como benefícios do Auto-ML foram encontrados os seguintes benefícios:</p> <p>Facilita o acesso ao ML, pois ela torna o aprendizado de máquina mais acessível para usuários novatos, permitindo que tomem decisões informadas sem a necessidade de consultores especializados.</p> <p>Padronização e reprodutibilidade: padroniza o processo de desenvolvimento de ML, melhorando a reprodutibilidade, a manutenção do código e a transferência de conhecimento.</p>

		<p>Procedimento de entrevista: as entrevistas foram semi-estruturadas e realizadas entre outubro de 2019 e março de 2020, durando aproximadamente uma hora cada. Cobriram três componentes principais: descrição do papel do participante e seus casos de uso de ML, desafios enfrentados no desenvolvimento de ML sem Auto-ML, e experiências e percepções sobre o uso de ferramentas de Auto-ML.</p> <p>Análise do estudo: as entrevistas foram gravadas em áudio e transcritas e os dados foram codificados de forma colaborativa e iterativa usando a ferramenta Dedoose para mapear os dados em categorias. Os autores realizaram um exercício de categorização para identificar temas comuns, como vantagens e desvantagens do Auto-ML, percepções sobre Auto-ML, estratégias de fluxo de trabalho com e sem Auto-ML, e decisões de adoção de Auto-ML.</p> <p>Limitações: as entrevistas focaram em um único caso de uso de ML por participante, o que pode ter limitado a diversidade de casos de uso. A amostra de participantes pode não ser representativa</p>	<p>Prevenção de resultados subótimos: pode evitar resultados subótimos causados por práticas específicas de desenvolvedores experientes.</p> <p>Eficiência e efetividade: melhora a eficiência (reduzindo o tempo de desenvolvimento) e a efetividade (melhorando o desempenho dos modelos).</p> <p>Prototipagem rápida: facilita a criação de protótipos rápidos para avaliar a viabilidade e o impacto potencial do ML em aplicações industriais.</p> <p>Aprendizado: usuários aprendem novas técnicas de modelagem e implementações de algoritmos de ML ao inspecionar o histórico de busca do Auto-ML.</p> <p>Deficiências do Auto-ML:</p> <p>Falta de suporte completo de ponta a ponta: como ela usada principalmente para treinamento de modelos, acaba exigindo esforço manual significativo para pré-processamento de dados e pós-processamento.</p> <p>Suporte limitado para pré-processamento de dados: não oferecem suporte adequado para tarefas de</p>
--	--	--	--

		<p>da base geral de usuários de Auto-ML.</p>	<p>limpeza e transformação de dados.</p> <p>Suporte limitado para modelos complexos e tipos de dados: como ela foca principalmente em dados tabulares e tarefas de classificação e regressão, oferece suporte limitado para aprendizado não supervisionado e dados não estruturados.</p> <p>Falhas de sistema devido a cargas de trabalho intensivas: podem causar falhas de sistema devido à alta demanda computacional.</p> <p>Falta de personalização: não permitem maior controle sobre a alocação de recursos computacionais, procedimentos de limpeza de dados, espaço de busca de modelos e técnicas de interpretabilidade.</p> <p>Falta de transparência e interpretabilidade: frequentemente são vistas como caixas-pretas, o que dificulta a confiança e a compreensão dos modelos gerados.</p> <p>Papel dos humanos nas técnicas de Auto-ML:</p> <p>Aprimoramento do desempenho e eficiência das técnicas de Auto-ML: humanos guiam o Auto-ML, restringindo o espaço</p>
--	--	--	---

			<p>de busca e compensando suas deficiências.</p> <p>Segurança e prevenção de uso indevido: humanos supervisionam o Auto-ML para garantir a segurança das decisões e evitar o uso indevido.</p> <p>Desenvolvimento manual para confiabilidade: em projetos críticos, os humanos preferem desenvolver manualmente para garantir a confiabilidade e a explicabilidade dos modelos.</p> <p>Todos esses pontos destacam a importância de uma colaboração equilibrada entre humanos e ferramentas de Auto-ML, no lugar de buscar a automação completa.</p>
--	--	--	--

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)

2. importacao e separação da base de dados

```
from tensorflow.keras.datasets import fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", y_test.shape)
print("y_test.shape: ", y_test.shape)
x_train.shape: (60000, 28, 28)
y_train.shape: (60000,)
x_test.shape: (10000,)
y_test.shape: (10000,)
```

3. Pré-processamento

```
# Normalizar os valores dos pixels para o intervalo [0, 1]
x_train, x_test = x_train/255.0, x_test/255.0

print(x_train.shape, x_test.shape) # Mantém o formato original
(60000, 28, 28) (10000, 28, 28)
```

4. Criação do Modelo

```
i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)

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

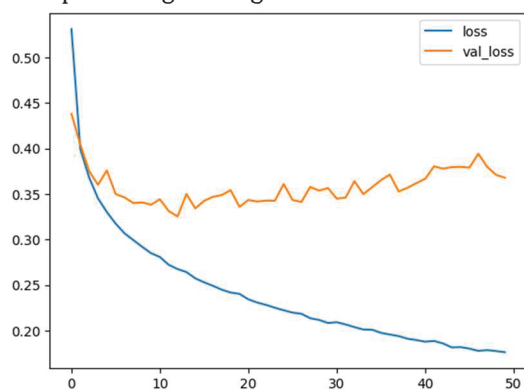
5. Compilação e Treinamento do Modelo

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
r = model.fit(x_train,
             y_train,
             validation_data=(x_test, y_test),
             epochs=50)
```

Epoch 1/50
1875/1875 ————— 6s 2ms/step - accuracy: 0.7616 - loss: 0.6706 - val_accuracy: 0.8447 - val_loss: 0.4293
 (...)
 Epoch 49/50
1875/1875 ————— 4s 2ms/step - accuracy: 0.9337 - loss: 0.1715 - val_accuracy: 0.8888 - val_loss: 0.4121
 Epoch 50/50
1875/1875 ————— 3s 2ms/step - accuracy: 0.9359 - loss: 0.1641 - val_accuracy: 0.8926 - val_loss: 0.3885

6. Avaliação do Modelo

```
# Plotar a função de perda
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
<matplotlib.legend.Legend at 0x7a1f5019ee00>
```



A função de perda no conjunto de treinamento (loss) está diminuindo de forma consistente, indicando que o modelo está aprendendo e ajustando seus pesos corretamente durante o treinamento.

Porém, após algumas épocas (cerca de 10), a perda no conjunto de validação (val_loss) começa a se estabilizar ou até mesmo a aumentar, enquanto a perda no conjunto de treinamento continua diminuindo.

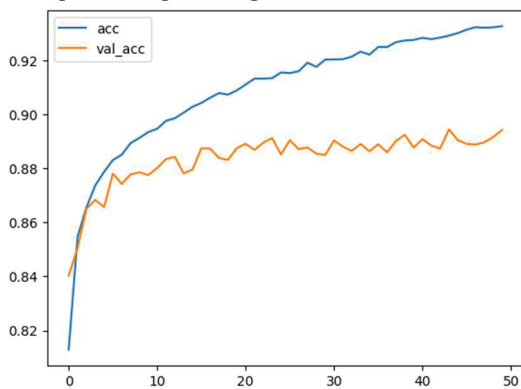
Isso sugere overfitting, ou seja, o modelo está memorizando os dados de treinamento e perdendo a capacidade de generalizar para novos dados.

Para tentar contornar a situação, podemos fazer as seguintes técnicas:

- 1) melhorar o desempenho do modelo interrompendo o treinamento antes que o val_loss comece a aumentar. Isso pode ser feito utilizando uma técnica chamada Early Stopping no Keras.
- 2) regularização para evitar overfitting, adicionando dropout no modelo ou regularização L2.
- 3) Ajustando os hiperparâmetros, para que reduza a taxa de aprendizado do otimizador adam, o que pode ajudar a encontrar um ponto de convergência melhor.

Plotar a acurácia

```
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
<matplotlib.legend.Legend at 0x7a1f1f356980>
```



O mesmo comentário se aplica quando olhamos para a acurácia do modelo após as 10 épocas.

Avaliar o modelo com a base de teste

```
print(model.evaluate(x_test, y_test))
```

```
313/313 ————— 0s 1ms/step - accuracy: 0.8931 - loss: 0.3775
[0.36792024970054626, 0.8942999839782715]
```

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]
```

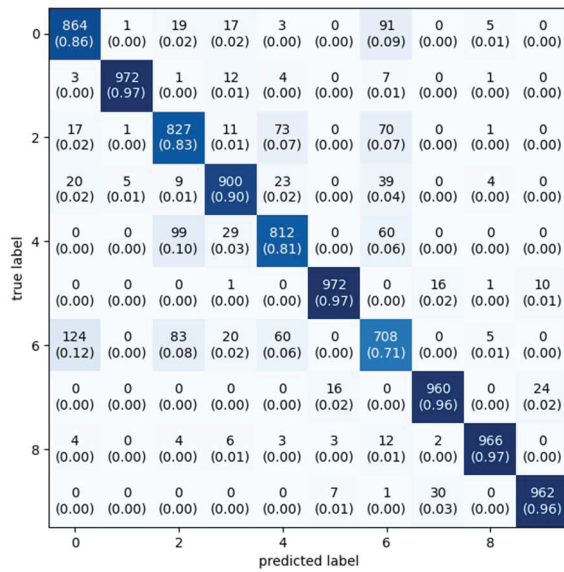
Matriz de confusão

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

```
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
                      show_normed=True)
```

```
(<Figure size 700x700 with 1 Axes>,
```

```
<Axes: xlabel='predicted label', ylabel='true label'>)
```

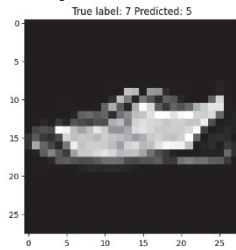



8. Mostrando algumas predições erradas

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

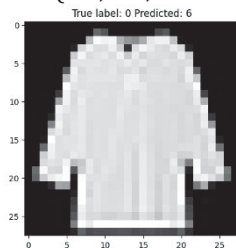
```
i = np.random.choice(misclassified)
```

```
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: 7 Predicted: 5')
```



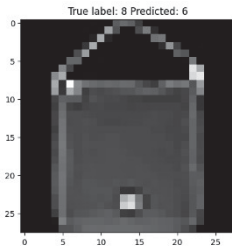
```
i = np.random.choice(misclassified)
```

```
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')
```



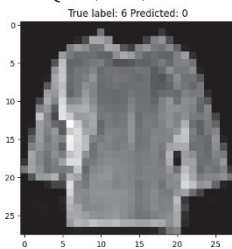
```
i = np.random.choice(misclassified)
```

```
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: 8 Predicted: 6')
```



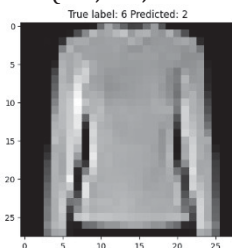
```
i = np.random.choice(misclassified)
```

```
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: 6 Predicted: 0')
```



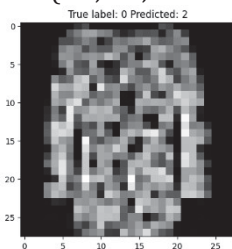
```
i = np.random.choice(misclassified)
```

```
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: 6 Predicted: 2')
```



```
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: 2')
```



Analisar exemplos mal classificados é crucial para entender onde o modelo está falhando e como ele pode ser melhorado. Pode indicar a necessidade de mais dados de treinamento, ajustes nos hiperparâmetros ou melhorias na arquitetura da rede.

Este gráfico é uma ferramenta útil para diagnosticar e melhorar o desempenho do modelo de classificação. Ele destaca as áreas onde o modelo precisa de ajustes e ajuda a direcionar os esforços de melhoria.

2 REGRESSÃO (RNA)

1. importação das bibliotecas

```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

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

2. importação da base de dados

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

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

```
data.head(2)
```

	acidez_fixa	acidez_volatil	acido_citrico	acucar_residual	cloratos	dioxido_de_enxofre_livre	dioxido_de_enxofre_total	densidade	pH	sulfatos	alcoool	score_qualidade_vinho
0	7.4	0.70	0.0	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.0	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5

3. Separação do modelo

```
# Se parando em variavel resposta e explicativa
X = data.drop('score_qualidade_vinho',axis = 1)
Y = data['score_qualidade_vinho'].astype(np.float32)
```

[illegible]

4. Criando o modelo

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

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

5. Compilação e treinamento do modelo

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

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

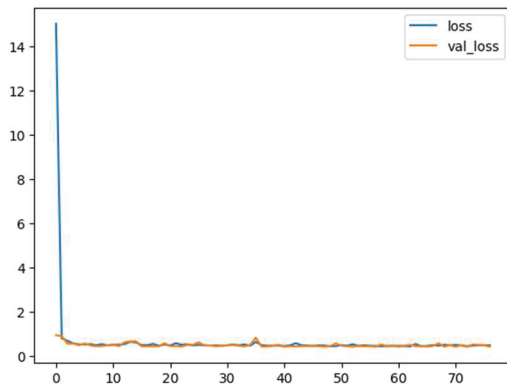
model.compile(optimizer=optimizer,
              loss="mse",
              metrics=[rmse, r2])
# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True)
r = model.fit(x_train, y_train,
              epochs=1500,
  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

Epoch 1/1500
35/35 ————— 1s 7ms/step - loss: 35.8539 - r2: -61.3804 - rmse: 3.7833 - val_loss: 0.9386 - v
al_r2: -0.5609 - val_rmse: 0.7534
(...)
35/35 ————— 0s 3ms/step - loss: 0.4718 - r2: 0.2675 - rmse: 0.5414 - val_loss: 0.4227 - val_r
2: 0.2979 - val_rmse: 0.4978
```

6. Avaliação do modelo

```
# função da perda

plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
<matplotlib.legend.Legend at 0x7a460470a320>
```



Este gráfico é crucial para diagnosticar o comportamento do modelo durante o treinamento e a validação. Ele permite ajustes no modelo ou nos hiperparâmetros, se necessário, para melhorar o desempenho.

O gráfico de perda do modelo se ajustei muito rápido aos dados, tanto de validação quanto de treino, não tendo tanta necessidade de chegar até as 70 épocas.

Observações Específicas

- **Início do Treinamento:**

A perda é alta, mas rapidamente diminui, indicando que o modelo está aprendendo.

- **Estabilização:**

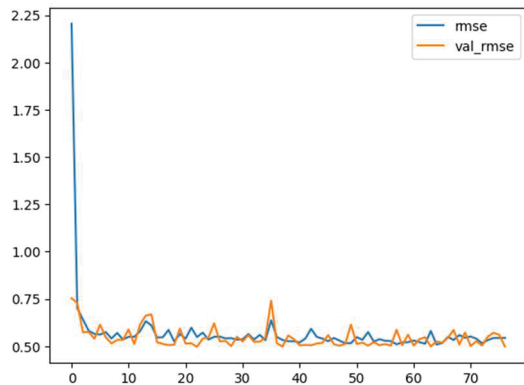
Após algumas épocas, tanto a perda quanto a perda de validação se estabilizam em valores baixos, sugerindo que o modelo está convergindo.

- **Proximidade das Linhas:**

A proximidade entre as linhas de perda e perda de validação sugere que o modelo não está sofrendo de overfitting significativo, pois a perda de validação não está aumentando enquanto a perda de treinamento continua a diminuir.

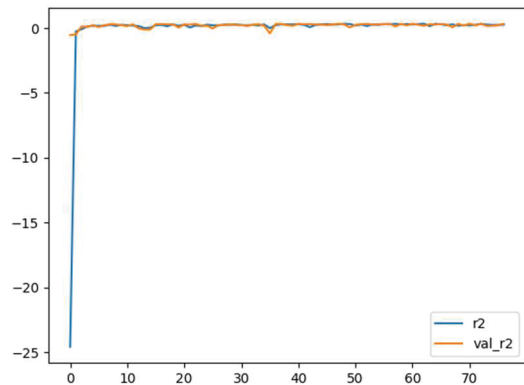
Valor de RMSE

```
plt.plot( r.history["rmse"], label="rmse" )
plt.plot( r.history["val_rmse"], label="val_rmse" )
plt.legend()
<matplotlib.legend.Legend at 0x7a4604e06200>
```



Valor do R2

```
plt.plot( r.history["r2"], label="r2" )
plt.plot( r.history["val_r2"], label="val_r2" )
plt.legend()
<matplotlib.legend.Legend at 0x7a4604eae320>
```



7. 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 = ", round(r2*100,2) )
mse = 0.41172823
rmse = 0.6416605279265538
r2 = 35.47
```

1. MSE e RMSE: O modelo tem um erro de previsão razoável, mas o erro quadrático (RMSE) pode ser alto, dependendo do contexto e das unidades dos dados.
2. R^2 : Com 35.47%, o modelo está explicando uma parte moderada da variabilidade dos dados, mas ainda há bastante variabilidade que o modelo não está capturando. Isso pode indicar que o modelo pode ser aprimorado, ou que outras variáveis explicativas devem ser consideradas.

Proximos passos para tentar melhorar as métricas:

- Ajuste do modelo: mudar a arquitetura da rede neural ou adicionar novas variáveis para melhorar a explicação da variabilidade dos dados.
- Análise de Residuals: Verifique os resíduos (diferença entre valores previstos e reais) para avaliar se há padrões que o modelo não está capturando.
- Ajustes de Hiperparâmetros: Teste diferentes hiperparâmetros, como número de camadas, neurônios, taxas de aprendizado, etc.

2 SISTEMAS DE RECOMENDAÇÃO

1. Importando as bibliotecas

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Embedding, Flatten, Concatenate, Dropout
from tensorflow.keras.models import Model
from keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import SGD, Adam

from sklearn.utils import shuffle

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

2. Importando as bases

```
df = pd.read_excel('Base_livros.xlsx')

# removendo as linhas em que nao tivemos nota
print(df.shape)
df = df.dropna(subset=['Notas', 'Titulo'])
print(df.shape)
(128838, 7)
(128142, 7)
```

3. Conversão de userId e movield para categoria

```
df['ID_usuario'].isna().sum(), df['Titulo'].isna().sum()
(0, 0)
# userId e movield não estão no formato certo para usar
# Embeddings > devem ser categóricos

df.ID_usuario = pd.Categorical(df.ID_usuario)
df['new_user_id'] = df.ID_usuario.cat.codes

df.Titulo = pd.Categorical(df.Titulo)
df['new_book_id'] = df.Titulo.cat.codes
# Dimensões
N = len(set(df.new_user_id))
M = len(set(df.new_book_id))

# dimensão do embedding (tentar outros)
K = 10
```

4. Criar o modelo

```
# Definindo o modelo com 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)

# 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

# filme
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])

# Camadas densas com Dropout
x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)
```

5. Compilação do modelo

```
model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)
```

6. Separação dos dados e pré-processamento

```
user_ids, book_ids, ratings = shuffle(df.new_user_id, df.new_book_id, df.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

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
import numpy as np

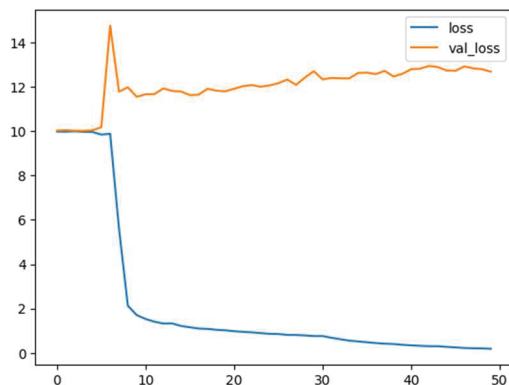
# Verifica NaNs ou Inf nos dados
print(np.any(np.isnan(train_user)), np.any(np.isnan(train_book)), np.any(np.isnan(train_ratings)))
print(np.any(np.isinf(train_user)), np.any(np.isinf(train_book)), np.any(np.isinf(train_ratings)))
False False False
False False False
```


7. Treinamento do modelo

```
epochs = 50
r = model.fit(
    x=[train_user, train_book],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso
    validation_data=([test_user, test_book], test_ratings)
)
Epoch 1/50
101/101 - 2s - 18ms/step - loss: 9.9868 - val_loss: 10.0307
Epoch 2/50
(...)
Epoch 50/50
101/101 - 0s - 3ms/step - loss: 0.1837 - val_loss: 12.6909
```

8. 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()
```



Para o conjunto de treinamento a função perda diminui rapidamente nas primeiras épocas, o que indica que o modelo está aprendendo a ajustar seus pesos para minimizar o erro. Por outro lado nos dados de treinamento diminui no início, mas estabiliza e começa a aumentar levemente ao longo das épocas levando indícios de possível overfitting, ou seja, o modelo está se ajustando bem aos dados de treinamento, mas está perdendo sua capacidade de generalização para novos dados (os dados de validação).

Este gráfico é crucial para diagnosticar o comportamento do modelo durante o treinamento e a validação. Ele permite ajustes no modelo ou nos hiperparâmetros, se necessário, para melhorar o desempenho.

9. Recomendações para o usuário 276726

```
# Gerar o array com o usuário único
# repete a quantidade de filmes
input_usuario = np.repeat(a=276726.0, repeats=M)
film = np.array(list(set(book_ids)))

preds = model.predict([input_usuario, film])
```

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

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

```
print("Recomendação: Filme - ", film[idx], " / ", rat[idx], "*")
3622/3622 ————— 6s 2ms/step
```

```
Recomendação: Filme - 19841 / 16.128386 *
df[df['new_book_id'] == 7801].Titulo.unique()
```

```
['Rage']
Categories (115904, object): ['Clara Callan', 'Decision in Normandy',
                             'Flu: The Story of the Great Influenza Pandemi...', 'The Mummies of Urumchi', ...,
                             'Iron Fist (Star Wars: X-Wing Series, Book 6)', 'Funny Money', 'Sun Valley',
                             'The Adventures of Lando Calrissian: Lando Cal...']
```

O modelo recomenda o filme "Rage" para o usuário com ID 276726, baseado nas predições do modelo e ajustado pela média das avaliações.

4 DEEPPDREAM

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: <a
```

```
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg">Von.grzanka</a>'))
```

Downloading data from https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-cat_on_snow.j

pg

```
2125399/2125399 ————— 0s 0us/step
```



Image cc-by: Von.grzanka

3. 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/inc
eption_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)
```

4. Cálculo da perda (loss)

```
def calc_loss(img, model):
    # Passe a imagem pelo modelo para recuperar as ativações.
    # Converte a imagem em um batch de tamanho 1.
    img_batch = tf.expand_dims(img, axis=0)
    layer_activations = model(img_batch)
    if len(layer_activations) == 1:
        layer_activations = layer_activations

    losses = []
    for act in layer_activations:
        loss = tf.math.reduce_mean(act)
        losses.append(loss)

    return tf.reduce_sum(losses)
```

5. Subida de gradiente (Gradient ascent)

```
class DeepDream(tf.Module):
    def __init__(self, model):
        self.model = model

    @tf.function(
        input_signature=(
            tf.TensorSpec(shape=[None, None, 3], dtype=tf.float32),
            tf.TensorSpec(shape=[], dtype=tf.int32),
            tf.TensorSpec(shape=[], dtype=tf.float32),
        )
    )
    def __call__(self, img, steps, step_size):
        print("Tracing")
        loss = tf.constant(0.0)

        for n in tf.range(steps):
            with tf.GradientTape() as tape:
                # Gradientes relativos a img
                tape.watch(img)
                loss = calc_loss(img, self.model)

            # Calculo do gradiente da perda em relação aos pixels da imagem de entrada.
            gradients = tape.gradient(loss, img)

            # Normalizacao dos gradintes
            gradients /= tf.math.reduce_std(gradients) + 1e-8
```

*# Na subida gradiente, a "perda" é maximizada.
Você pode atualizar a imagem adicionando diretamente os gradientes (porque eles têm o mesmo formato!)*

```
img = img + gradients*step_size
img = tf.clip_by_value(img, -1, 1)
```

```
return loss, img
deepdream = DeepDream(dream_model)
```

6. 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)
```



O resultado é uma imagem modificada que realça os padrões vistos pela rede neural, criando um efeito visual diferente da imagem original e interessante, porém com baixa granularidade.

7. Levando o modelo até um oitava

```
import time
start = time.time()
```

```
OCTAVE_SCALE = 1.30
```

```
img = tf.constant(np.array(original_img))
```

```

base_shape = tf.shape(img)[:1]
float_base_shape = tf.cast(base_shape, tf.float32)

for n in range(-2, 3):
    new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

    img = tf.image.resize(img, new_shape).numpy()

    img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start

```



261.60134625434875

O resultado é uma imagem modificada, mais próxima do objetivo de criar uma imagem que remete ao 'sonho', criando um efeito visual diferente da imagem original e surreal, com aumento da granularidade.

Ao comparar as duas imagens, percebemos que a primeira imagem gerada é mais semelhante à imagem original, com pouca granularidade. Ao elevar o modelo à oitava, temos uma imagem gerada com maior granularidade e mais surreal, remetendo melhor a imagens oníricas.

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

Análise da Desvalorização de SUVs 2024

Conjunto de Dados Brutos

Os conjuntos de dados utilizados nesta análise consistem em valores históricos da tabela FIPE (Fundação Instituto de Pesquisas Econômicas, <https://veiculos.fipe.org.br/#carro>) para três modelos de SUVs do ano 2024:

- **Jeep Compass Sport T270 1.3 TB 4x2 Flex Aut.**
- **Honda HR-V EXL 1.5 Sensing 16V 5p Aut.**
- **Toyota Corolla Cross XR 2.0 16V Flex Aut.**

Os dados compreendem o período de janeiro de 2024 a março de 2025, contendo valores mensais da tabela FIPE. Estes números representam a referência oficial de preços de veículos no mercado brasileiro, servindo como parâmetro para negociações de compra e venda, financiamentos, seguros e questões tributárias.

Foi utilizado código em Python para obter os dados e transformá-los no formato final apresentado abaixo:

index,month,month_full,jeep,honda,toyota,jeep_perc,honda_perc,toyota_perc									
0	Jan/24	janeiro de 2024	151364	155330	152746	100.0	100.0	100.0	
1	Fev/24	fevereiro de 2024	146066	155889	151615	96.5	100.4	99.3	
2	Mar/24	março de 2024	140953	154827	151084	93.1	99.7	98.9	
3	Abr/24	abril de 2024	136019	153155	151053	89.9	98.6	98.9	
4	Mai/24	maio de 2024	137745	152445	151415	91.0	98.1	99.1	
5	Jun/24	junho de 2024	140672	153394	150012	92.9	98.8	98.2	
6	Jul/24	julho de 2024	135748	152339	148462	89.7	98.1	97.2	
7	Ago/24	agosto de 2024	130996	153557	145210	86.5	98.9	95.1	
8	Set/24	setembro de 2024	125564	153445	148541	83.0	98.8	97.2	
9	Out/24	outubro de 2024	129682	152490	146656	85.7	98.2	96.0	
10	Nov/24	novembro de 2024	125143	152084	147900	82.7	97.9	96.8	
11	Dez/24	dezembro de 2024	123998	149258	147134	81.9	96.1	96.3	
12	Jan/25	janeiro de 2025	122015	151098	146194	80.6	97.3	95.7	
13	Fev/25	fevereiro de 2025	119853	151033	144772	79.2	97.2	94.8	
14	Mar/25	março de 2025	118654	150812	143924	78.4	97.1	94.2	

Contexto e Público-Alvo

Contexto

O mercado automotivo brasileiro experimenta dinâmicas particulares de desvalorização que afetam diretamente consumidores e investidores. Veículos novos frequentemente apresentam curvas de desvalorização acentuadas nos primeiros meses após o lançamento de um modelo, porém com intensidades diferentes entre marcas e categorias. Esta análise busca explorar essas diferenças no segmento de SUVs, um dos mais disputados no país atualmente.

Público-Alvo

Esta visualização e narrativa são direcionadas a:

- Consumidores planejando a compra de um SUV: Pessoas avaliando qual modelo pode representar melhor custo-benefício considerando a desvalorização futura.
- Proprietários atuais destes veículos: Interessados em entender o comportamento de valor de seus bens.
- Profissionais do setor automotivo: Concessionárias, revendedores e consultores que precisam compreender tendências de mercado.
- Investidores: Indivíduos que consideram veículos como ativos financeiros e precisam avaliar sua depreciação.

Visualização dos Dados

Escolha do Tipo de Visualização

1. Ponto de Partida – Janeiro de 2024

Foi utilizado um gráfico de barras comparativo, com o objetivo de exibir lado a lado os preços iniciais dos três modelos (Honda HR-V, Toyota Corolla Cross e Jeep Compass). Cada barra representa o preço de lançamento de cada modelo, facilitando a comparação visual dos valores iniciais (R\$ 155.330,00 para HR-V; R\$ 152.746,00 para Corolla Cross; R\$ 151.364,00 para Compass).

2. Primeiro Trimestre 2024

Foi utilizado um gráfico de linhas de evolução dos preços, com o objetivo de mostrar a variação dos preços ao longo dos primeiros seis meses. Foi utilizada uma linha para cada modelo, destacando que o HR-V manteve uma estabilidade (98,7% do valor original),

enquanto o Compass apresentou uma queda significativa em abril seguida de recuperação parcial, e o Corolla Cross teve uma desvalorização moderada. Eixo X representando os meses (Jan, Abril, Junho) e eixo Y os preços (ou percentual do valor inicial).

3. Segundo Semestre de 2024

Foi utilizado um gráfico de linhas focado no segundo semestre, com o objetivo de destacar a trajetória de queda mais acentuada, especialmente do Jeep Compass, durante esse período. Foram utilizadas linhas separadas para cada modelo, com marcações para pontos críticos (ex.: queda do Compass em setembro para 82,9% do valor inicial e recuperação leve no Corolla Cross). Essa visualização pode evidenciar a diferença na estratégia de precificação e o impacto das promoções.

4. Jornada completa

Foi utilizado um gráfico de linhas do período completo da análise, com o objetivo de fornecer uma visão integrada da jornada de desvalorização dos três modelos desde janeiro de 2024 até março de 2025. Cada linha representa um modelo, com o eixo X representando os meses e o eixo Y os preços. Essa abordagem permite identificar visualmente os pontos de estabilidade, inflexão e a comparação final entre os modelos.

5. Comparativo final

Foi utilizado um gráfico de barras duplas, com o objetivo de comparar os valores finais dos três modelos e visualizar a porcentagem de desvalorização ao fim dos 15 meses. Foi utilizado um conjunto de barras para cada modelo mostrando o preço final (ex.: HR-V em R\$ 150.812,00, Corolla Cross em R\$ 143.924,00, Compass em R\$ 118.654,00).

Outra informação apresentada foi a porcentagem de desvalorização (2,9% para HR-V, 5,8% para Corolla Cross e 21,6% para Compass).

Escolha da Ferramenta

Para esta visualização, utilizamos a linguagem Python e a biblioteca Matplotlib, assim como Pandas e Numpy como bibliotecas auxiliares para o processamento dos dados.

A escolha ocorreu naturalmente já que havíamos utilizado Python e Pandas para processar os dados. Com poucas linhas de código conseguimos obter o gráfico desejado sem alternar a ferramenta.

Narrativa/Storytelling

A História da Desvalorização dos SUVs: Uma Jornada de 15 Meses

Nossa apresentação conta a história de três competidores no acirrado mercado de SUVs no Brasil: Jeep Compass, Honda HR-V e Toyota Corolla Cross, todos do modelo 2024. Estes veículos representam três das montadoras mais respeitadas no país, com propostas de valor e estratégias de mercado distintas.

Os slides podem ser encontrados na seção “Anexos”.

O Ponto de Partida: Janeiro de 2024

Nossa história começa em janeiro de 2024, com os três modelos recém-lançados e avaliados em valores próximos:

- Honda HR-V: R\$ 155.330,00
- Toyota Corolla Cross: R\$ 152.746,00
- Jeep Compass: R\$ 151.364,00

Neste momento inicial, o HR-V se posiciona como o mais valorizado, refletindo o prestígio da marca Honda no mercado brasileiro. Os três modelos estão relativamente próximos em valor, separados por apenas R\$ 3.966,00 (cerca de 2,5%).

A Divergência: Os Primeiros Seis Meses

À medida que avançamos para o meio de 2024, percebemos caminhos distintos:

- **Honda HR-V:** Mantém notável estabilidade, com oscilações mínimas de valor. Em junho de 2024, mantinha 98,7% do seu valor original.
- **Toyota Corolla Cross:** Demonstra moderada desvalorização, chegando a junho de 2024 com 98,2% do valor inicial.
- **Jeep Compass:** Apresenta a trajetória mais volátil, atingindo sua primeira queda significativa em abril (R\$ 136.019,00 - 89,9% do valor inicial), seguida de uma recuperação parcial em junho (R\$ 140.672,00 - 92,9%).

Esta fase revela o primeiro insight importante: a Honda consegue sustentar melhor o valor de seu modelo nos primeiros meses, enquanto o Jeep Compass mostra maior vulnerabilidade às flutuações de mercado.

O Ponto de Inflexão: Segundo Semestre de 2024

O segundo semestre marca um ponto crítico na nossa jornada, com desdobramentos significativos:

- **Jeep Compass:** Enfrenta quedas consistentes, atingindo seu valor mais baixo em setembro (R\$ 125.564,00 - apenas 82,9% do valor inicial).
- **Toyota Corolla Cross:** Apresenta queda mais acentuada em agosto (R\$ 145.210,00 - 95,1%), seguida de leve recuperação.
- **Honda HR-V:** Continua demonstrando estabilidade impressionante, mantendo-se acima de R\$ 152.000,00 até outubro.

Este período ilustra como políticas diferentes de precificação e posicionamento de mercado resultam em desempenhos distintos. O Jeep, que apostou em descontos e promoções mais agressivas, vê seu valor de tabela refletir essa estratégia.

O Desfecho: Primeiro Trimestre de 2025

Ao final de nossa jornada, em março de 2025, vemos o resultado de 15 meses de mercado:

- **Honda HR-V:** Termina avaliado em R\$ 150.812,00 - uma desvalorização de apenas 2,9%
- **Toyota Corolla Cross:** Fecha em R\$ 143.924,00 - uma queda de 5,8%
- **Jeep Compass:** Encerra em R\$ 118.654,00 - uma desvalorização expressiva de 21,6%

A conclusão de nossa história revela um fenômeno importante no mercado automotivo brasileiro: nem todos os SUVs se comportam da mesma forma no que tange à retenção de valor. O Honda HR-V demonstra uma excepcional capacidade de preservar seu valor, enquanto o Jeep Compass apresenta uma desvalorização substancialmente maior. Já o Toyota Corolla Cross apresenta comportamento semelhante ao Honda HR-V, ainda que com desvalorização um pouco maior.

Significado dos Dados e Ações Recomendadas

Para Consumidores:

- **Se você prioriza retenção de valor:** O Honda HR-V demonstra ser a escolha mais segura para minimizar perdas com desvalorização.
- **Se você busca oportunidades no mercado usado:** O Jeep Compass apresenta a maior desvalorização, podendo representar boas oportunidades de compra no mercado secundário.
- **Se planeja vender em curto prazo:** Considere que o Jeep Compass pode perder até um quinto de seu valor em apenas um ano.

Para Investidores e Revendedores:

- **Desenvolva estratégias de preço diferenciadas para cada marca,** reconhecendo que o Honda possui maior poder de retenção de valor.
- **Ajuste margens de negociação** considerando que o Jeep tende a desvalorizar mais rapidamente.
- **Considere que o Toyota mantém posição intermediária e estável,** podendo ser uma opção equilibrada para o portfólio.

Para Fabricantes e Concessionárias:

- **Reavaliar estratégias de produção e oferta** para modelos como o Compass que apresentam desvalorização mais acentuada.
- **Comunicar o valor de revenda como diferencial competitivo** para modelos como o HR-V.
- **Explorar políticas de comercialização** que considerem as diferentes curvas de desvalorização.

Objetivo da Visualização

O objetivo principal desta visualização é permitir que os diversos stakeholders do mercado automotivo compreendam de forma clara e imediata como se comportam diferentes modelos de SUVs no que diz respeito à retenção de valor ao longo do tempo.

Ao contar esta história através dos dados, buscamos transformar números brutos em insights acionáveis que possam informar decisões financeiras importantes, desde a escolha do melhor momento para comprar ou vender um veículo até estratégias de precificação para empresas do setor.

Anexos

Dados Brutos

Dados obtidos da FIPE (<https://veiculos.fipe.org.br/#carro>):

index	month	month_full	jeep	honda	Toyota
0	Jan/24	janeiro de 2024	151364	155330	152746
1	Fev/24	fevereiro de 2024	146066	155889	151615
2	Mar/24	março de 2024	140953	154827	151084
3	Abr/24	abril de 2024	136019	153155	151053
4	Mai/24	maio de 2024	137745	152445	151415
5	Jun/24	junho de 2024	140672	153394	150012
6	Jul/24	julho de 2024	135748	152339	148462
7	Ago/24	agosto de 2024	130996	153557	145210
8	Set/24	setembro de 2024	125564	153445	148541
9	Out/24	outubro de 2024	129682	152490	146656
10	Nov/24	novembro de 2024	125143	152084	147900
11	Dez/24	dezembro de 2024	123998	149258	147134
12	Jan/25	janeiro de 2025	122015	151098	146194
13	Fev/25	fevereiro de 2025	119853	151033	144772
14	Mar/25	março de 2025	118654	150812	143924

Dados transformados:

index	month	month_full	jeep	honda	toyota	jeep_perc	honda_perc	toyota_perc
0	Jan/24	janeiro de 2024	151364	155330	152746	100.0	100.0	100.0
1	Fev/24	fevereiro de 2024	146066	155889	151615	96.5	100.4	99.3
2	Mar/24	março de 2024	140953	154827	151084	93.1	99.7	98.9
3	Abr/24	abril de 2024	136019	153155	151053	89.9	98.6	98.9
4	Mai/24	maio de 2024	137745	152445	151415	91.0	98.1	99.1
5	Jun/24	junho de 2024	140672	153394	150012	92.9	98.8	98.2
6	Jul/24	julho de 2024	135748	152339	148462	89.7	98.1	97.2
7	Ago/24	agosto de 2024	130996	153557	145210	86.5	98.9	95.1
8	Set/24	setembro de 2024	125564	153445	148541	83.0	98.8	97.2
9	Out/24	outubro de 2024	129682	152490	146656	85.7	98.2	96.0
10	Nov/24	novembro de 2024	125143	152084	147900	82.7	97.9	96.8
11	Dez/24	dezembro de 2024	123998	149258	147134	81.9	96.1	96.3
12	Jan/25	janeiro de 2025	122015	151098	146194	80.6	97.3	95.7
13	Fev/25	fevereiro de 2025	119853	151033	144772	79.2	97.2	94.8
14	Mar/25	março de 2025	118654	150812	143924	78.4	97.1	94.2

Apresentação (slides)



ANÁLISE DA DESVALORIZAÇÃO DE SUVS 2024



VALÉRIO WITTNER - VALERIOV@GMAIL.COM

PROCURANDO UM SUV PARA COMPRAR?

Qual SUV oferece a melhor
retenção de valor?

Onde estão as
oportunidades no mercado
de usados?



SUV PARA INVESTIR OU REVENDER?

Como ajustar estratégias
de preço e margens de
negociação?

Qual modelo é a escolha
mais equilibrada para o
portfólio?



É FABRICANTE OU CONCESSIONÁRIA?

Quais estratégias de
produção e oferta devem
ser reavaliadas?

Como comunicar o valor de
revenda como diferencial
competitivo?



PERÍODO DE ANÁLISE

- Período de Dados
 - Janeiro de 2024 a março de 2025
- Valores Mensais
 - Referência oficial de preços de veículos (FIPE)

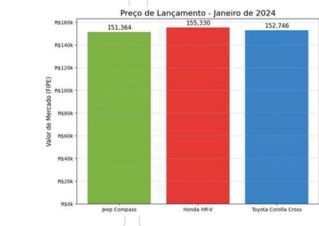


MODELOS DE SUVs ANALISADOS

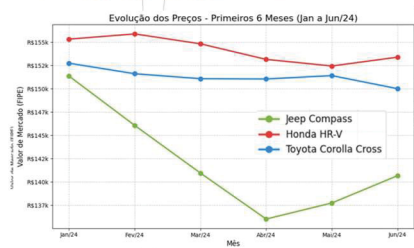
- Jeep Compass Sport T270 1.3 TB 4x2 Flex Aut.
- Honda HR-V EXL 1.5 Sensing 16V 5p Aut.
- Toyota Corolla Cross XR 2.0 16V Flex Aut.



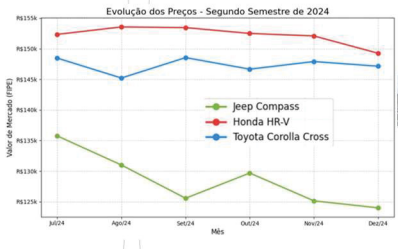
PREÇO JAN 2024



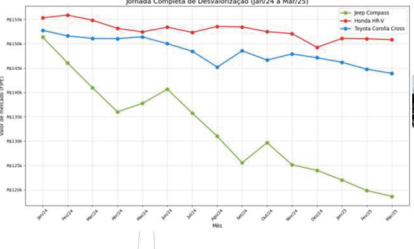
1º TRIMESTRE 2024



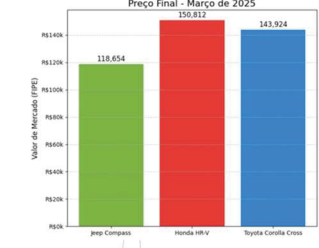
2º TRIMESTRE 2024



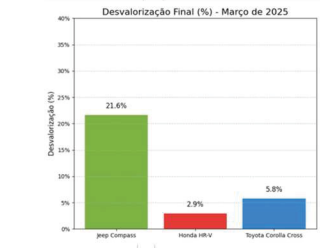
JAN/24 A MAR/25



FINAL: MARÇO 2025



FINAL (%): MARÇO 2025



PARA CONSUMIDORES



Retenção de Valor

Honda HR-V é a escolha mais segura para minimizar perdas com desvalorização



Oportunidades no Mercado Usado

Jeep Compass apresenta a maior desvalorização. Boas oportunidades de compra no mercado secundário



Venda em Curto Prazo

Jeep Compass pode perder até um quinto de seu valor em um ano



Desenvolvimento de Estratégias de Preço

Estratégias diferenciadas para cada marca. Reconhecimento do poder de retenção de valor da Honda



Ajuste de Margens de Negociação

Consideração da rápida desvalorização do Jeep



Posição Intermediária do Toyota

Opção equilibrada para o portfólio

PARA INVESTIDORES E REVENDEDORES

*PARA FABRICANTESE
CONCESSIONÁRIAS*



Reavaliar estratégias de produção e oferta

Focar em modelos como o Compass que apresentam desvalorização acentuada



Comunicar o valor de revenda

Utilizar como diferencial competitivo para modelos como o HRV



Explorar políticas de comercialização

Considerar as diferentes curvas de desvalorização

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);
- (6) Preserve sempre a melhor solução de uma geração para outra.

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) COMPARE A REPRESENTAÇÃO DE DOIS MODELOS VETORIAIS

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de

palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

1) ALGORITMO GENÉTICO

```
import random
import math
def cria_cidades(qtde):
    """
    Gera cidades com coordenadas aleatórias.
    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.
    """
    cidades = []
    # Primeira cidade. Vai ser a última também. Posições 0 e 100.
    x1 = random.uniform(0, 100)
    y1 = random.uniform(0, 100)
    cidades.append([x1, y1])
    for i in range(1, qtde-1):
        x = random.uniform(0, 100)
        y = random.uniform(0, 100)
        cidades.append([x, y])
    # Última cidade. Igual a primeira
    cidades.append([x1, y1])
    return cidades

def populacaoInicial(tamanho, cidades):
    populacao = []
    for i in range(tamanho):
        lista_cidades = cidades[1:len(cidades)-1] # obtém somente cidades intermediárias, descartando início e
        fim que é a mesma cidade.
        random.shuffle(lista_cidades) # embaralha as cidades intermediárias
        rota = [cidades[0]] + lista_cidades + [cidades[0]] # Monta lista novamente com intermediárias
        embaralhadas e inicial e final fixas
        populacao.append(rota)
    return populacao

def distancia_euclidiana(cidade1, cidade2):
    res = math.sqrt((cidade1[0]-cidade2[0])**2 + (cidade1[1]-cidade2[1])**2)
    return res
```



```
def avaliacao(populacao):
```

```
    """
```

```
    A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
```

```
    """
```

```
    fit = []
```

```
    for rota in populacao:
```

```
        total = 0.0
```

```
        rotas = rota[:-1]
```

```
        for i in range(len(rotas)-1):
```

```
            total += distancia_euclidiana(rotas[i], rotas[i+1])
```

```
        fit.append(total)
```

```
    return fit
```

```
def mutacao(rota, tamanho_pop, qtde_mutacoes):
```

```
    """
```

```
    (4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação
```

```
    """
```

```
    qtde_mutacoes = math.floor(tamanho_pop * perc_mutacao)
```

```
    for i in range(qtde_mutacoes):
```

```
        nova_rota = rota.copy()
```

```
        tamanho = len(nova_rota)
```

```
        pos1, pos2 = random.sample(range(1, tamanho - 1), 2)
```

```
        nova_rota[pos1], nova_rota[pos2] = nova_rota[pos2], nova_rota[pos1]
```

```
    return nova_rota
```

```
def cruzamento(pai1, pai2):
```

```
    """Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox)"""
```

```
    """
```

```
    Operador de Cruzamento Ordenado (OX):
```

- Opera apenas sobre as posições intermediárias (índices 1 até n-2), pois as últimas posições(início e fim) são fixas.
- Seleciona dois pontos de corte e preserva o segmento do pai1; preenche as demais posições com as cidades do pai2, mantendo a ordem e ignorando as duplicidades.

```
    """
```

```
    tamanho = len(pai1)
```

```
    filho = [None] * tamanho
```

```
    # Seleciona os pontos de corte (garante que estejam entre 1 e n-2)
```

```
    ponto1 = random.randint(1, tamanho - 3)
```

```
    ponto2 = random.randint(ponto1 + 1, tamanho - 2)
```

```
    # Copia o segmento de pai1 para o filho
```

```
    filho[ponto1:ponto2+1] = pai1[ponto1:ponto2+1]
```

```
    # Preenche as posições restantes com as cidades de pai2 na ordem
```

```
    pos = (ponto2 + 1) if (ponto2 + 1) < tamanho - 1 else 1
```

```
    for cidade in pai2[1:tamanho-1]:
```

```
        if cidade not in filho:
```

```
            filho[pos] = cidade
```

```
            pos = pos + 1 if pos < tamanho - 2 else 1
```

```
    # As posições fixas (início e fim) recebem a cidade de início
```

```
    filho[0] = pai1[0]
```

```
    filho[-1] = pai1[0]
```

```
    return filho
```

```
# passo 1
```

```
qtde_cidades = 100
```

```
tamanho_pop = 100
```

```
geracoes = 1000
```



```

n_novos = tamanho_pop - 1 # novos indivíduos
perc_cruzamento = 0.9 # indivíduos gerados via cruzamento (90%)
n_cruzamento = math.floor(tamanho_pop*perc_cruzamento)
perc_mutacao = 0.1 # 10% mutacao
n_mutacao = math.floor(tamanho_pop*perc_mutacao)

cidades = cria_cidades(qtde_cidades)
p0 = populacaoInicial(tamanho_pop, cidades)
# avalia a pop. inicial
a0 = avaliacao(p0)

melhor_distancia = min(a0)
melhor_rota = p0[a0.index(melhor_distancia)]

todas_rotas = [melhor_rota]
todas_distancias = [melhor_distancia]

print("----- POPULAÇÃO INICIAL -----")
print(f"Pop inicial: {p0}")
print(f"Avaliação Pop inicial: {a0}")
print(f"Melhor distância na Pop inicial: {melhor_distancia:.2f}")
print(f"Melhor rota na Pop inicial: {melhor_rota}")
----- POPULAÇÃO INICIAL -----
Pop inicial: [[[45.001348343198686, 84.10243487528801], [18.502549217494714, 77.75014352699917], [33.4921558224041, 33.086434641099196], [52.64507185412718, 21.12151141851325], [50.0387425654707, 34.52572053576641], [48.314657751814906, 56.254707364271226], [22.12085534179128, 21.682852080760174], [38.808961956919234, 39.01122231426086], [13.576279114409061, 9.1274136
(...)
109181400011284], [43.38379483841683, 61.48995342069456], [3.2855269552518163, 69.33284374301894], [11.796136341198993, 2.0044999782873663], [62.70434195526142, 49.76588010418494], [45.001348343198686, 84.10243487528801]]]
def melhor_individuo(populacao):
    """
    Retorna o melhor indivíduo (rota com menor distância) da população atual.
    """
    valores_avaliacao = avaliacao(populacao)
    melhor_distancia = min(valores_avaliacao)
    indice_melhor_rota = valores_avaliacao.index(melhor_distancia)
    melhor_rota = populacao[indice_melhor_rota]
    return melhor_rota, melhor_distancia

def preserva_melhor(populacao, nova_pop):
    """
    Preserva o melhor indivíduo (rota com menor distância) da população atual.
    Insere na nova população e retorna esse melhor indivíduo e sua avaliação.
    """
    melhor_rota, melhor_distancia = melhor_individuo(populacao)
    nova_pop.append(melhor_rota)
    return melhor_rota, melhor_distancia

def seleciona_individuo(populacao, n=5):
    """
    - Sorteia n indivíduos aleatoriamente e retorna o que apresentar a menor distância.
    """
    candidatos = random.sample(populacao, n)
    valores_avaliacao = avaliacao(candidatos)

    melhor_distancia = min(valores_avaliacao)
    indice_melhor_rota = valores_avaliacao.index(melhor_distancia)
    melhor_rota = candidatos[indice_melhor_rota]
    return melhor_rota
populacao = p0

```

```

melhor_geral = melhor_rota
melhor_geral_avaliacao = melhor_distancia

print(f"Geração 000 - Melhor distância: {melhor_distancia:.2f}")

# Evolução através das gerações
for ger in range(1, geracoes+1):
    nova_pop = []
    # Preserva o melhor indivíduo da geração atual
    melhor, aval_melhor = preserva_melhor(populacao, nova_pop)

    # Geração de novos indivíduos via cruzamento (90% dos novos indivíduos)
    for i in range(n_cruzamento):
        pai1 = seleciona_individuo(populacao)
        pai2 = seleciona_individuo(populacao)
        while pai1 == pai2:
            pai2 = seleciona_individuo(populacao)
        filho = cruzamento(pai1, pai2)
        nova_pop.append(filho)

    # Geração de novos indivíduos via mutação (mínimo 1% dos novos indivíduos)
    for i in range(n_mutacao):
        individuo = seleciona_individuo(populacao)
        mutante = mutacao(individuo, tamanho_pop, qtde_mutacoes=n_mutacao)
        nova_pop.append(mutante)

    # Atualiza a população e obtém melhor indivíduo
    populacao = nova_pop

    atual, atual_avaliacao = melhor_individuo(populacao)

    todas_rotas.append(atual)
    todas_distancias.append(atual_avaliacao)

    if atual_avaliacao < melhor_geral_avaliacao:
        melhor_geral = atual
        melhor_geral_avaliacao = atual_avaliacao

    # Exibe métrica a cada 100 gerações
    if ger % 100 == 0:
        print(f"Geração {ger} - Melhor distância: {atual_avaliacao:.2f}")

print("\nMelhor solução após 1000 gerações:")
print("Rota:", melhor_geral)
print("Distância:", melhor_geral_avaliacao)
Geração 000 - Melhor distância: 4345.21
Geração 100 - Melhor distância: 1967.14
Geração 200 - Melhor distância: 1547.10
Geração 300 - Melhor distância: 1435.07
Geração 400 - Melhor distância: 1390.88
Geração 500 - Melhor distância: 1328.13
Geração 600 - Melhor distância: 1303.79
Geração 700 - Melhor distância: 1235.56
Geração 800 - Melhor distância: 1170.38
Geração 900 - Melhor distância: 1150.00
Geração 1000 - Melhor distância: 1144.58

Melhor solução após 1000 gerações:
Rota: [[45.001348343198686, 84.10243487528801], [53.171843000171584, 91.8370920311527], [64.43488163595681, 79.99314
774959714], [84.83141190957919, 84.03585808241624], [72.86295389775104, 86.6559976115516], [65.94607167674565, 91.4
4935700424114], [74.00579725489122, 96.50683655302991], [68.19519383175965, 73.44157865404924], [67.5744812601174
8, 66.01472750174561], [68.54944563140747, 63.8440388522643], [71.70221898061794, 57.70648147472768], [77.436388257
58936, 64.58410387332121], [87.8489413457471, 61.42164274057623], [96.46600757209411, 57.83286799966616], [99.60654

```

```

119681105, 49.6338330236764], [88.2667132300605, 47.51016309879096], [91.76529436218948, 39.69507904474888], [88.61
857677328442, 36.913599638992665], [61.057938645984756, 54.24033779842233], [77.64043711563983, 38.35737413304966
], [86.02692692476076, 26.862978869388666], [95.66876575252302, 6.879456554528341], [87.01548443322942, 16.18527299
164877], [90.41382662120095, 28.000780329201504], [90.94629810600648, 21.64864118622415], [69.22833431368404, 31.47
9834718691315], [72.90059588430036, 43.85356524936574], [73.14833137612732, 50.87443286343073], [75.0640933870850
7, 52.17474685733989], [62.70434195526142, 49.76588010418494], [43.38379483841683, 61.48995342069456], [43.62008584
9107326, 58.77503859833261], [38.03371724069689, 60.9522626247174], [36.790285434718704, 68.2129585543359], [41.513
791061473306, 67.67709351768345], [57.13103023627956, 65.75559723386696], [60.175344358655266, 66.60026931485424],
[47.76598011632842, 60.85881386131997], [48.314657751814906, 56.254707364271226], [38.808961956919234, 39.01122231
426086], [33.4921558224041, 33.086434641099196], [39.55419725936494, 46.77995384374823], [49.27148784312112, 51.793
24135772577], [52.64507185412718, 21.12151141851325], [61.45474661290995, 43.59140082572298], [47.33054216182859, 3
7.17398576647268], [50.0387425654707, 34.52572053576641], [61.67571961660621, 28.734527716722134], [60.87146903785
112, 21.787225907616016], [59.79153988044364, 21.57791947794181], [70.47057022861704, 17.384491706406113], [71.6814
8770543817, 0.2381843761620317], [76.84220454153609, 1.0247461085245324], [81.61158084964939, 3.7211532303244033],
[75.95018641062148, 18.144406078414278], [58.9451628918882, 8.135894082410488], [58.27677440297235, 9.336093538215
195], [59.72352004690163, 10.745355442413118], [47.724937851556426, 8.705069101670448], [41.02038086486402, 10.9401
38800593092], [31.752885945871522, 18.728002551577326], [27.903027557054973, 16.827889118016405], [28.83866121566
211, 21.192950849665515], [31.58840814772923, 11.251159336199922], [9.808787716549162, 14.588233036602983], [1.1148
377164914236, 5.532332369798388], [13.576279114409061, 9.12741365937907], [0.9926022171756865, 10.532515433201517
], [11.796136341198993, 2.0044999782873663], [28.567876405378367, 3.109694944151764], [31.81996351632719, 3.8822947
81385931], [37.8357267961998, 3.971019769822437], [25.019905261506338, 4.624092533031643], [22.994105856987957, 6.2
20932041097827], [31.39453544351146, 14.813484218467677], [22.12085534179128, 21.682852080760174], [23.8870886758
03835, 26.109181400011284], [34.61904535248941, 31.449783938015628], [37.60550565578127, 34.6223534581274], [7.1711
53935583963, 41.68559298403769], [10.547812810610912, 45.5740204464395], [0.9645220378936226, 21.182256264395495],
[14.257124094530326, 16.9334862240891], [11.51803584180432, 40.77726971920117], [24.763701077459565, 48.0540104653
24296], [26.205485566040398, 66.1197214375517], [27.86835252097921, 67.49851317770727], [28.71600039360638, 63.6304
705896267], [18.502549217494714, 77.75014352699917], [12.332733342981438, 70.7814665203206], [1.6599637301893067, 7
5.61701697063843], [6.150648138935377, 72.9514175787701], [3.2855269552518163, 69.33284374301894], [6.799177508857
568, 91.9365194876732], [3.1985636553024, 93.17079958809829], [19.650227831348865, 97.44729603260141], [21.87608167
9934178, 89.94476272156129], [32.03993630836464, 77.32795306720213], [25.48906985224304, 74.52633617304723], [45.00
1348343198686, 84.10243487528801]]

```

Distância: 1144.5773328700104

```
import matplotlib.pyplot as plt
```

```
# Gráfico rota
```

```
def grafico_rota(cidades, rota, titulo):
```

```
    rotas = [i for i in rota]
```

```
    xs, ys = zip(*rotas)
```

```
    plt.figure(figsize=(6,6))
```

```
    plt.scatter(*zip(*cidades), c='blue')
```

```
    plt.plot(xs, ys, c='red', lw=2)
```

```
# Destaque a primeira cidade
```

```
plt.scatter(xs[0], ys[0], c='green', s=100, label='Início/Fim') # Aumenta o tamanho e muda a cor
```

```
# Posiciona a legenda embaixo do gráfico
```

```
plt.legend(loc='lower center', bbox_to_anchor=(0.5, -0.15), ncol=3) # Ajusta a posição e o número de  
colunas
```

```
plt.title(titulo)
```

```
plt.xlabel("X")
```

```
plt.ylabel("Y")
```

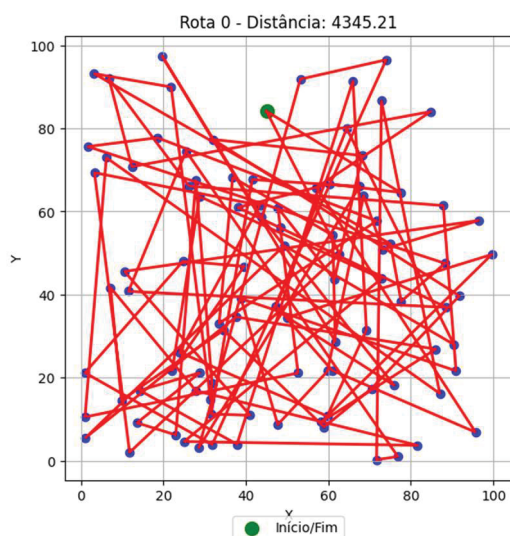
```
plt.grid(True)
```

```
plt.show()
```

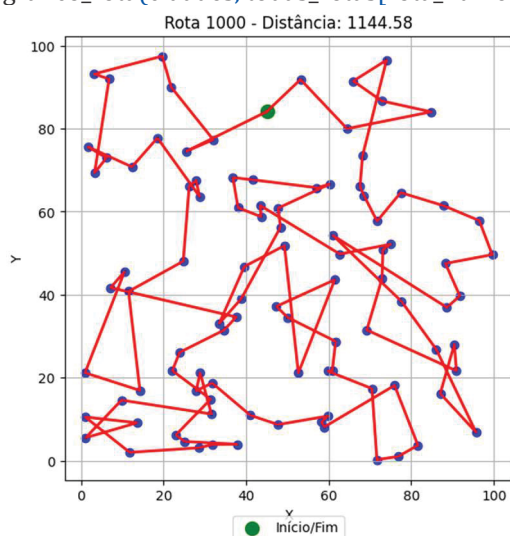
```
rota_numero = 0
```

```
titulo = f'Rota {rota_numero} - Distância: {todas_distancias[rota_numero]:.2f}'
```

```
grafico_rota(cidades, todas_rotas[rota_numero], titulo)
```



```
rota_numero = ger
titulo = f'Rota {rota_numero} - Distância: {todas_distancias[rota_numero]:.2f}'
grafico_rota(cidades, todas_rotas[rota_numero], titulo)
```



Visualizar vários gráficos

```
"""
for rota_numero in range(len(todas_rotas)):
    if rota_numero % 100 == 0:
        titulo = f'Rota {rota_numero} - Distância: {todas_distancias[rota_numero]:.2f}'
        grafico_rota(cidades, todas_rotas[rota_numero], titulo)
"""

'\nfor rota_numero in range(len(todas_rotas)):\n if rota_numero % 100 == 0:\n  titulo = f'Rota {rota_numero} - Distância: {todas_distancias[rota_numero]:.2f}'\n  grafico_rota(cidades, todas_rotas[rota_numero], titulo)\n'
```

2) COMPARE A REPRESENTAÇÃO DE DOIS MODELOS VETORIAIS

```
!pip install --upgrade gensim
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import PCA
import re
import numpy as np
from gensim.models import Word2Vec
```

```

# Textos
textos = [
    "Eu gosto de maçãs e bananas.",
    "Eu adoro maçãs, são muito saborosas.",
    "O cachorro corre no parque.",
    "A gata dorme no sofá.",
    "A gata dorme no sofá.",
    "O parque é bonito e espaçoso.",
    "A natureza inspira paz e harmonia."
]

# Função de pré-processamento para remover pontuação e converter para minúsculas
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[\^\w\s]', '', text)
    return text.split()

# Utilizando CountVectorizer
def count_vectorizer(textos):
    count_vectorizer = CountVectorizer()
    count_matrix = count_vectorizer.fit_transform(textos).toarray()
    return count_matrix

# Utilizando TfidfVectorizer
def tfidf_vectorizer(textos):
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform(textos).toarray()
    return tfidf_matrix

# Função para obter a média dos vetores das palavras de cada sentença
def sentence_vector(w2v_model, sentence):
    vectors = [w2v_model.wv[word] for word in sentence if word in w2v_model.wv]
    if vectors:
        return np.mean(vectors, axis=0)
    else:
        return np.zeros(w2v_model.vector_size)

def word2vec(textos):
    sentences = [preprocess(t) for t in textos]

    w2v_model = Word2Vec(sentences, vector_size=50, window=2, min_count=1, workers=2, seed=42)

    # Calcula os vetores para cada sentença
    sentence_vectors = np.array([sentence_vector(w2v_model, sentence) for sentence in sentences])
    return sentence_vectors

count_matrix = count_vectorizer(textos)
tfidf_matrix = tfidf_vectorizer(textos)
sentence_vectors = word2vec(textos)

# PCA
# CountVectorizer
pca_count = PCA(n_components=2)
pca_count_res = pca_count.fit_transform(count_matrix)

# TF-IDF
pca_tfidf = PCA(n_components=2)
pca_tfidf_res = pca_tfidf.fit_transform(tfidf_matrix)

# Word2Vec
pca_w2v = PCA(n_components=2)

```

```
pca_w2v_res = pca_w2v.fit_transform(sentence_vectors)
```

```
# Gráficos
```

```
def grafico(textos, pca_res, titulo):
```

```
    #plt.figure(figsize=(18, 5))
```

```
    for i, point in enumerate(pca_res):
```

```
        plt.scatter(point[0], point[1], color='blue')
```

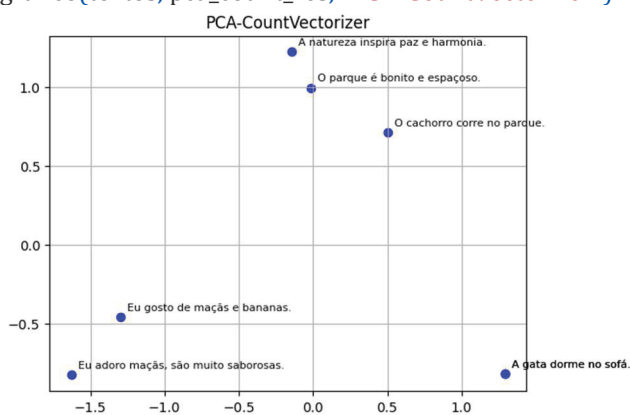
```
        plt.annotate(textos[i], (point[0], point[1]), textcoords="offset points", xytext=(5,5), ha='left', fontsize=8)
```

```
    plt.title(titulo)
```

```
    plt.grid(True)
```

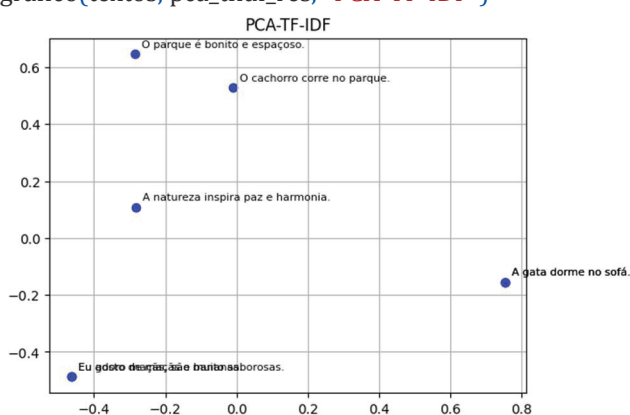
```
# PCA-CountVectorizer
```

```
grafico(textos, pca_count_res, "PCA-CountVectorizer")
```



```
# PCA-TF-IDF
```

```
grafico(textos, pca_tfidf_res, "PCA-TF-IDF")
```



```
# Word2Vec
```

```
grafico(textos, pca_tfidf_res, "PCA-Word2Vec")
```

