

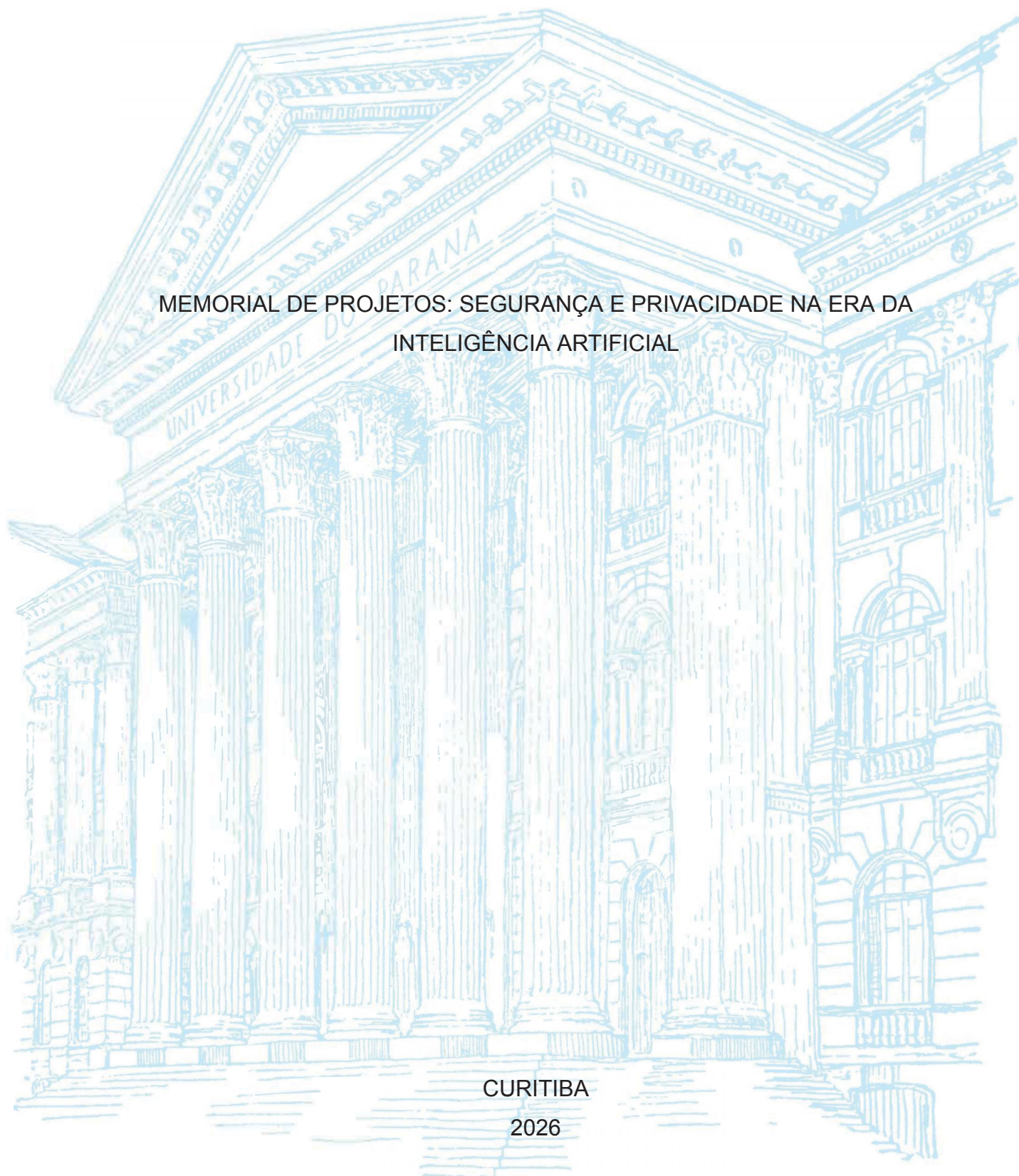
UNIVERSIDADE FEDERAL DO PARANÁ

IGOR BUESS ATALA Y MANSOUR

MEMORIAL DE PROJETOS: SEGURANÇA E PRIVACIDADE NA ERA DA
INTELIGÊNCIA ARTIFICIAL

CURITIBA

2026



IGOR BUESS ATALA Y MANSOUR

MEMORIAL DE PROJETOS: SEGURANÇA E PRIVACIDADE NA ERA DA
INTELIGÊNCIA ARTIFICIAL

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. Razer Anthom Nizer Rojas Montañó

CURITIBA

2026

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 **IGOR BUESS ATALA Y MANSOUR**, intitulada: **MEMORIAL DE PROJETOS: SEGURANÇA E PRIVACIDADE NA ERA DA INTELIGÊNCIA ARTIFICIAL**, 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, 06 de Março de 2026.



RAZER ANTHOMIZIER ROJAS MONTANO
Presidente da Banca Examinadora



RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

É notória a popularização das ferramentas de inteligência artificial para os mais variados objetivos tanto no grande público, quanto nas empresas. E devido a lógica de como os modelos utilizam dados, muitas vezes fornecidos pelos usuários, para constantemente se aprimorarem, muitas questões e preocupações quanto a proteção de dados pessoais vem surgindo nos últimos tempos. Vazamento de dados já é uma grande preocupação de muitas empresas desde de que sistemas de informação se tornaram o padrão em nossa sociedade, e muitas já têm restringido o uso interno de ferramentas de inteligência artificial apenas para aquelas devidamente homologadas e testadas. Entretanto ainda existem muitas dúvidas sobre as boas práticas de governança de dados que podem ser usados de forma segura nos treinamentos dos modelos. Outro aspecto relevante no mesmo contexto de privacidade é a constante extração de dados que determinados modelos executam na internet, colocando em risco aspectos como direitos autorais, ou mais especificamente, a violação deles. Sendo assim, é necessário planejar controles e limites para o *crawling* feito por estas ferramentas. Esses controles podem vir tanto via regulamentação quanto a criação de boas práticas de mercado sobre o desenvolvimento e operação destas ferramentas. Outra relevante preocupação é relacionada a eficiência de técnicas de anonimização que são aplicadas em bases que irão alimentar modelos durante suas fases de treinamento.

Palavras-chave: privacidade; segurança; vazamento; dados pessoais; violação.

ABSTRACT

The popularization of artificial intelligence tools for a wide variety of purposes, both among the general public and in companies, is well-known. Due to the logic of how these models use data, often provided by users, to constantly improve themselves, many questions and concerns regarding the protection of personal data have arisen recently. Data leaks have been a major concern for many companies since information systems became the standard in our society, and many have already restricted the internal use of artificial intelligence tools to only those that are properly certified and tested. However, many doubts remain about the best data governance practices that can be used safely in model training. Another relevant aspect in the same context of privacy is the constant extraction of data that certain models perform on the internet, putting aspects such as copyright, or more specifically, its violation, at risk. Therefore, it is necessary to plan controls and limits for the crawling done by these tools. These controls can come from regulation as well as the creation of best market practices regarding the development and operation of these tools. Another relevant concern relates to the efficiency of anonymization techniques applied to databases that will feed models during their training phases.

Keywords: privacy; security; leakage; personal data; disclosure.

SUMÁRIO

1 PARECER TÉCNICO.....	7
REFERÊNCIAS.....	9
APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA.....	10
APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA.....	18
APÊNDICE C - LINGUAGEM R.....	27
APÊNDICE D - ESTATÍSTICA APLICADA I.....	36
APÊNDICE E - ESTATÍSTICA APLICADA II.....	50
APÊNDICE F - ARQUITETURA DE DADOS.....	63
APÊNDICE G - APRENDIZADO DE MÁQUINA.....	86
APÊNDICE H - DEEP LEARNING.....	105
APÊNDICE I - BIG DATA.....	140
APÊNDICE J - VISÃO COMPUTACIONAL.....	145
APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	190
APÊNDICE L - GESTÃO DE PROJETOS DE IA.....	201
APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	204
APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING.....	219
APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	222

1 PARECER TÉCNICO

Os avanços da inteligência artificial e o ganho de produtividade gerada pelo uso de ferramentas deste tipo são inegáveis. Redigir textos, construir códigos de programas, buscar orientações sobre quaisquer temas: todas essas ações são muito facilitadas por estas ferramentas. Contudo, existe um custo que não deve ser ignorado pela sociedade como um todo, que é o risco aos dados pessoais e a privacidade tanto das pessoas que utilizam as ferramentas diretamente, quanto para pessoas que são autoras ou detêm direitos autorais sobre trabalhos publicados na internet.

Os modelos de Inteligência Artificial (IA) que são ofertados para o público via internet como, por exemplo, o ChatGPT, utilizam dados públicos em seus algoritmos durante a fase de treinamento para poder gerar respostas mais precisas o possível para o usuário (OPENAI, 2026). Essa consulta aos dados públicos é toda feita via internet, geralmente através dos chamados *crawlers*, usando das mais variadas formas de consumo de dados. A própria OpenAI (2025) explica a composição de sua base de dados:

Os modelos fundamentais da OpenAI, incluindo os modelos que alimentam o ChatGPT, são desenvolvidos usando três fontes principais de informação: (1) informação que está disponível publicamente na internet, (2) informação que acedemos em parceria com terceiros e (3) informação que os nossos utilizadores, treinadores humanos e investigadores fornecem ou geram. (OPENAI, 2025).

O problema do consumo de dados públicos abertos na internet para o treinamento de modelos de IA vai desde a questão do desrespeito a direitos autorais, até a diminuição do tráfego de internet nos websites ou fontes originais das informações. Esse problema ocorre pois a partir do momento que um modelo de IA consome e aprende determinada informação, o usuário perde grande parte do incentivo de acessar as fontes de dados originais. Na maioria das vezes ele, sequer, tem conhecimento de onde é a fonte de dados original, a não ser que, explicitamente, peça no *prompt* da ferramenta de IA que sejam citadas suas fontes.

Já é possível verificar um movimento da internet de fechamento ou controle de acesso a dados que, anteriormente, eram públicos. Isto é, diversos websites estão implementando controles como captchas e logins obrigatórios para evitar que *crawlers* de IA consumam suas informações de forma indiscriminada e gratuita. O

resultado disso é que haverá uma tendência futura de que a internet se torne um ambiente exclusivista e restrito, que é o contrário do seu propósito inicial quando foi criada (Longpre; Shayne, 2025) (The Economist, 2025).

Pode-se também destacar que muitas empresas estão tentando implementar internamente nos seus processos modelos de IA para aumento de produtividade e eficiência. Um caso de uso deste tipo são empresas de saúde que vêm tentando implementar ferramentas de diagnóstico baseadas em IA para, por exemplo, analisar imagens e exames para diagnosticar patologias. Segundo Murdoch (2021), a inteligência artificial traz desafios significativos para a privacidade das informações de saúde. Visto que as empresas têm acesso a uma massa de dados de saúde gigantesca e esforços relacionados a anonimização destes dados devem ser efetivos e não passíveis de identificação pessoal (Yadav *et al.* 2023).

Técnicas de anonimização ainda não são perfeitas no sentido de inviabilizar, por definitivo, a identificação das pessoas, e geralmente o que é alcançável atualmente é apenas a chamada pseudo anonimização. Ela assume que qualquer tentativa de anonimização pode ser desfeita se dados adicionais são adicionados ao conjunto de informações, por isso o termo *pseudo* é utilizado para indicar uma falsa anonimização (Kamp; Weinbaum, 2025).

Sendo assim, considera-se que é de extrema urgência que o desenvolvimento de um ecossistema de IA seja acompanhado de regras, boas práticas e princípios éticos na lida com as informações que serão usadas e consumidas por estas ferramentas. Os governos, as sociedades e os cidadãos devem estar plenamente cientes de que abrir mão da privacidade de seus dados em troca da comodidade de uma tecnologia de IA é um risco muito grande e não deve ser subestimado.

REFERÊNCIAS

KAMP, Roy; WEINBAUM, Noemie. *The myth of anonymization: why AI needs a new privacy paradigm*. IAPP, 26 mar. 2025. Disponível em: <https://iapp.org/news/a/the-myth-of-anonymization-why-ai-needs-a-new-privacy-paradigm>. Acesso em: 03 janeiro 2026.

LONGPRE, Shayne. *AI crawler wars threaten to make the web more closed for everyone*. 2025. Disponível em: <https://www.technologyreview.com/2025/02/11/1111518/ai-crawler-wars-closed-web/>. Acesso em: 05 janeiro 2026.

MURDOCH, B. *Privacy and artificial intelligence: challenges for protecting health information in a new era*. BMC Med Ethics 22, 122 (2021). <https://doi.org/10.1186/s12910-021-00687-3>

OPEN AI. *How ChatGPT and our foundation models are developed*. Disponível em: <https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-foundation-models-are-developed>. Acesso em: 30 janeiro 2026.

THE ECONOMIST. *AI is killing the web. Can anything save it?*. Disponível em: <https://www.economist.com/business/2025/07/14/ai-is-killing-the-web-can-anything-save-it>. Acesso em: 30 janeiro 2026.

YADAV N, *et al*. *Data Privacy in Healthcare: In the Era of Artificial Intelligence*. Indian Dermatol Online J. 2023 Oct 27;14(6):788-792. doi: 10.4103/idoj.idoj_543_23. PMID: 38099022; PMCID: PMC10718098.

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA

A – ENUNCIADO

1 ChatGPT

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

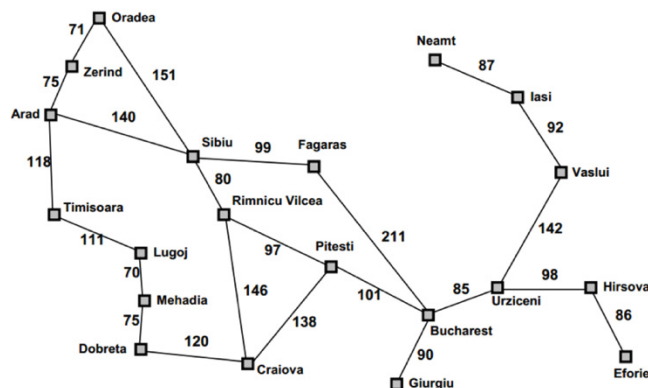
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

Se as uvas caem, então a raposa as come

Se a raposa as come, então estão maduras

As uvas estão verdes ou caem

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem

q: a raposa come as uvas

r: as uvas estão maduras

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

R1: $p \rightarrow q$

R2: $q \rightarrow r$

R3: $\neg r \vee p$

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

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

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

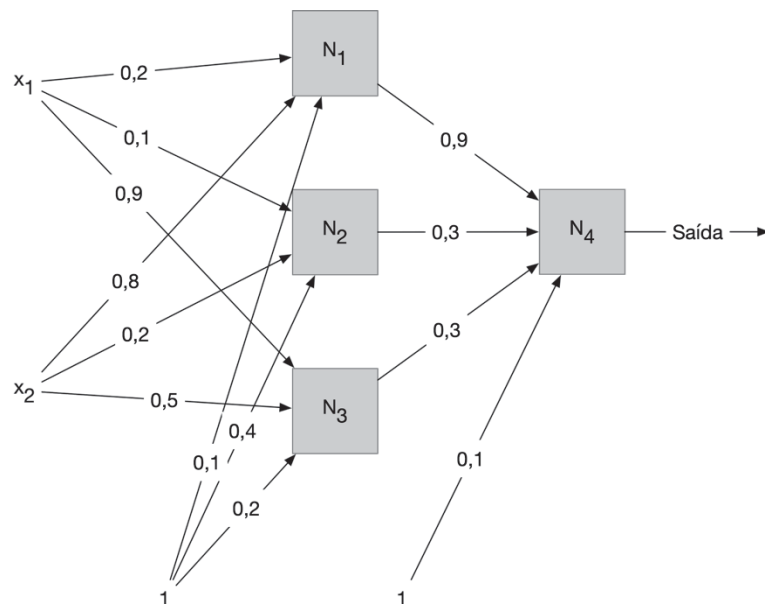
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

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

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

B – RESOLUÇÃO

1 - ChatGPT

a) “Inteligência Artificial (IA) refere-se à capacidade de uma máquina ou software imitar a inteligência humana. Em outras palavras, é a simulação de processos de pensamento humanos por meio de algoritmos e sistemas computacionais. A IA permite que as máquinas realizem tarefas que normalmente exigiriam inteligência humana, como aprendizado, raciocínio, reconhecimento de padrões, processamento de linguagem natural, visão computacional e tomada de decisões.

Existem dois tipos principais de IA: a estreita (ou fraca) e a geral (ou forte). A IA estreita é especializada em realizar tarefas específicas sem a capacidade de realizar atividades fora desse escopo, enquanto a IA geral busca replicar a ampla gama de habilidades cognitivas humanas.

As técnicas comuns usadas na implementação de IA incluem aprendizado de máquina (machine learning), redes neurais, algoritmos de otimização, lógica fuzzy e processamento de linguagem natural. A IA é aplicada em uma variedade de campos, como saúde, finanças, automação industrial, veículos autônomos, reconhecimento de voz, jogos, assistentes virtuais e muito mais. A pesquisa contínua nesse campo visa melhorar a eficiência e a adaptabilidade dos sistemas de IA, bem como explorar novas fronteiras na inteligência artificial.”

b) Na primeira frase do primeiro parágrafo, podemos classificar a resposta do ChatGPT como “Agir como Humanos”, pois ele define que IA como imitar a inteligência humana. Já na segunda frase, podemos classificar a resposta como “Pensar como Humanos”, pois ele afirma que seria a simulação do processo de pensamento humano.

No segundo parágrafo, podemos classificar a resposta como “Pensar racionalmente” quando se trata da IA estreita (ou fraca), enquanto a IA geral (ou forte) pode ser classificada como “Agir como Humano”.

No terceiro parágrafo, pode-se classificar os exemplos dados de aplicações da IA como IAs que se classificam em “Agir racionalmente”, pois são exemplos que foram desenvolvidos para tarefas específicas que respondem a ações e buscam o melhor resultado possível.

c) Os modelos GPT (incluindo o próprio ChatGPT), são projetados para gerar linguagem natural de maneira coerente e consistente com a linguagem humana. Esses modelos passam por um

processo de pré-treinamento, onde são treinados em grandes quantidades de dados de texto não supervisionados, como páginas da web ou livros. Posteriormente, esses modelos podem ser ajustados para tarefas específicas, como classificação de texto ou resposta a perguntas.

O desenvolvimento do ChatGPT envolveu uma série de marcos e melhorias, incluindo a introdução da arquitetura “Transformer” que possibilitou a criação de modelos de linguagem altamente eficientes e escaláveis. Uma característica-chave dos Transformers é a “atenção multi-cabeça” ou “attention heads”, onde o modelo pode ponderar diferentes partes do texto em relação a outras partes para capturar conexões significativas entre as palavras.

No caso do GPT-3.5, o modelo utiliza uma pilha de 13 blocos “Transformer”, cada um com 12 cabeças de atenção e 768 unidades ocultas. A entrada para o modelo é uma sequência de tokens usando uma camada de incorporação. Os tokens incorporados são então alimentados no primeiro bloco Transformer, que produz uma sequência de representações ocultas.

As representações ocultas são então passadas através dos 12 blocos “Transformer” restantes, cada um dos quais aplicam suas camadas. A saída do bloco “Transformer” final é uma sequência de representações ocultas, que são decodificadas em uma sequência de saída.

Além da arquitetura principal do “Transformer”, o GPT-3.5 também inclui vários componentes adicionais, como normalização de camadas, conexões residuais e incorporações posicionais. Esses componentes ajudam a estabilizar o treinamento e melhorar o desempenho do modelo em tarefas de modelagem de linguagem.

d) A partir do conhecimento adquirido acerca do funcionamento, treinamento e objetivos da OpenAI com o projeto do chatGPT, é possível observar duas abordagens: Agir como humanos e Agir Racionalmente.

Agir como humanos ao gerar linguagem natural de forma coerente e consistente com a linguagem humana, e, Agir Racionalmente ao buscar, por meio de seus modelos, a melhor forma possível de realizar alguma tarefa específica ou oferecer a melhor resposta à uma pergunta feita por seus usuários.

2- Busca Heurística

a)

FIGURA 1 - BUSCA HEURÍSTICA

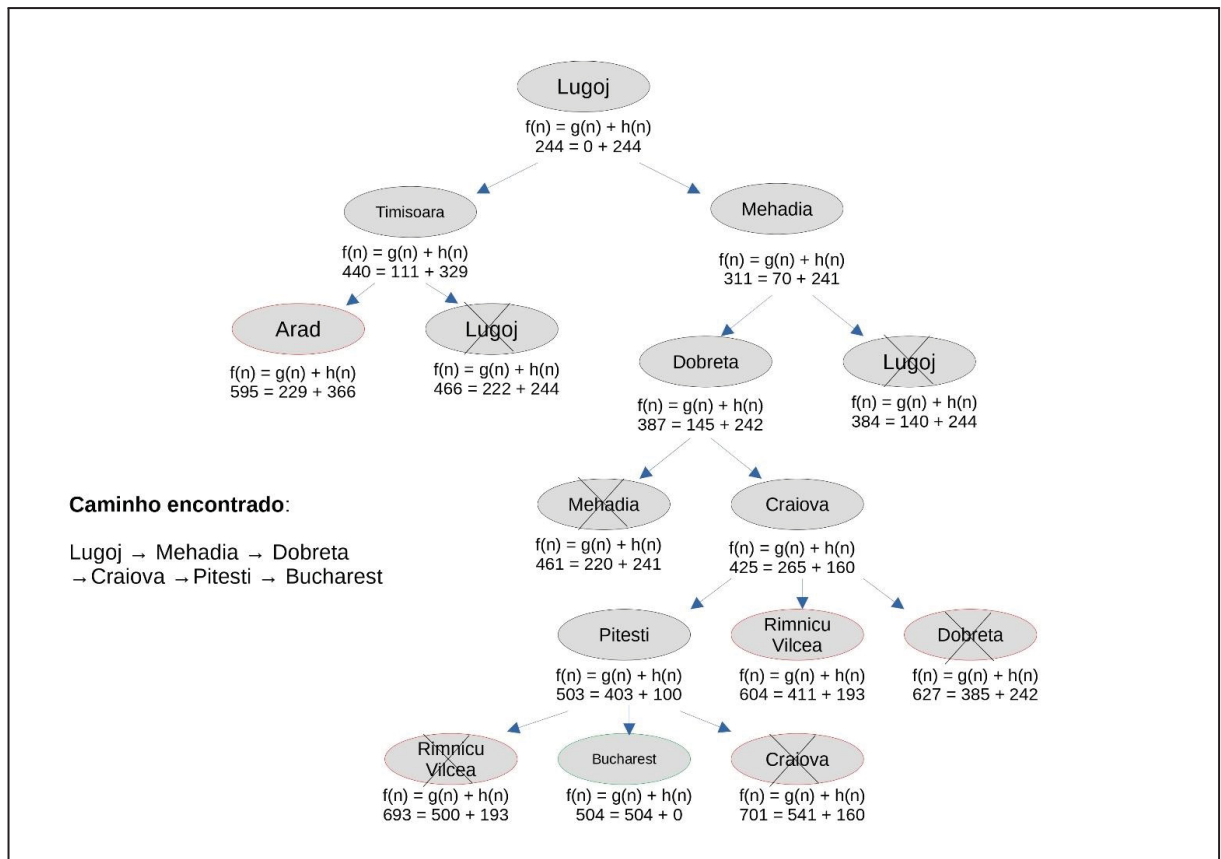


FIGURA 2 - RESULTADO DE BUSCA HEURÍSTICA

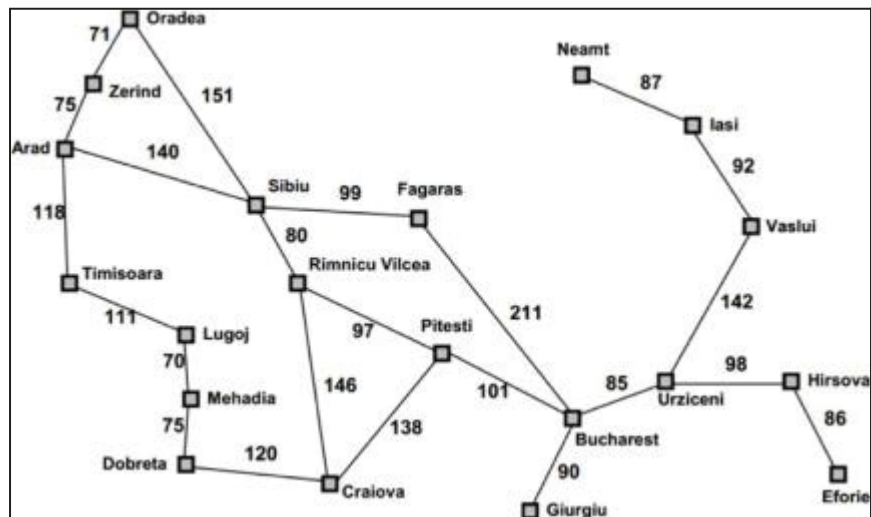


TABELA 1 - DISTÂNCIAS EM LINHA RETA PARA BUCARESTE

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

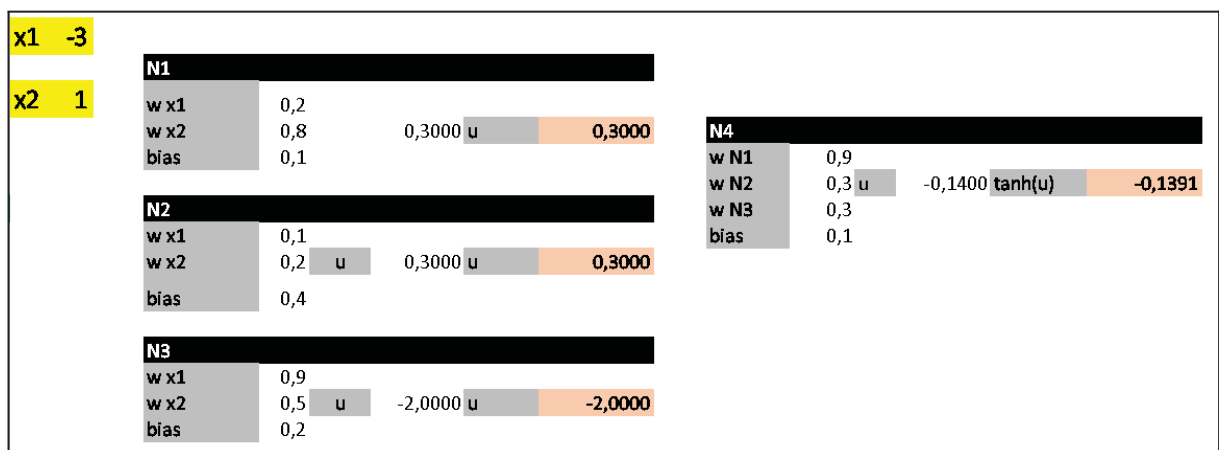
3- Lógica

a)

R4: $r \rightarrow p$ Equivalência condicional, R3R5: $q \rightarrow p$ Silogismo Hipotético, R2, R4R6: $(q \rightarrow p) \wedge (p \rightarrow q)$ Conjunção, R1, R5R7: $q \leftrightarrow p$ Equivalência Bicondicional, R6

4- Redes Neurais Artificiais

FIGURA 3 - CÁLCULOS DE SAÍDA DE CADA NEURÔNIO



a) Valor de saída do neurônio N1

VALOR: 0,3

b) Valor de saída do neurônio N2

VALOR: 0,3

c) Valor de saída do neurônio N3

VALOR: -2

d) Valor de saída da rede como um todo

VALOR: -0,1391

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

- 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 -

f) Existem valores faltantes e duplicados, removemos e fizemos as análises. A contagem de registros por marca, mostrou que a Fiat possui o maior número de registros, seguida pela Volkswagen e Chevrolet. Na contagem de modelos a marca mais presentes na base aparece em outra ordem sendo Fiat, Ford e Volkswagen.

2 -

e) Observou-se pouca variação na média de preços de carros automáticos por marca e manuais por marca. A exceção ocorre nos modelos da Renault, que apresentam valores significativamente inferiores tanto para os modelos automáticos quanto para os manuais. Os veículos de câmbio automático em geral apresentam maior preço médio.

g) É possível observar que veículos a diesel são mais caros. Principalmente veículos da marca VW que se destacam significativamente em relação aos valores dos veículos movidos a gasolina e álcool. Fabricantes Renault e Nissan não oferecem veículos movidos a álcool, porém, o preço dos veículos da Renault é comparável aos da Fiat movidos a gasolina.

3 -

f) Foi observado que em ambos os algoritmos de machine learning usados as variáveis de maior importância são a potência do motor 'engine_size' e o ano do modelo. Apesar da importância das variáveis no algoritmo XGBoost ter sido menor. A acurácia de ambos os modelos foi muito semelhante.

h) A avaliação utilizada foi a R^2 , ambos os modelos apresentaram resultados equivalentes, diante da métrica foi obtido 95,99% pelo modelo XGBoost e 96,03% pelo modelo Random Forest sem parâmetros configurados.

CÓDIGO

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

import warnings
warnings.filterwarnings('ignore')

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

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

dados.dtypes

dados.isna().any()

dados.isna().sum()

dados.dropna(how="all", inplace=True)

dados.duplicated().sum()

dados.drop_duplicates(inplace=True)

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

# Resumo das variáveis numéricas
dados[numericas_cols].describe()

# Resumo das variáveis categóricas
```

```
dados[categoricas_cols].describe()

dados.brand.value_counts()

dados.model.value_counts()

plt.figure(figsize=(10, 5))
plt.bar(dados.brand.unique(),
dados.brand.value_counts()[dados.brand.unique()])
plt.title('Distribuição de veículos por marca')
plt.ylabel('Total de veículos');

plt.figure(figsize=(5, 2.5))
plt.bar(dados.gear.unique(),
dados.gear.value_counts()[dados.gear.unique()])
plt.title('Distribuição de veículos por tipo de câmbio')
plt.ylabel('Total de veículos');

def atribui_numericos_mes(mes):
    if(mes == 'January'):
        return 1
    if(mes == 'February'):
        return 2
    if(mes == 'March'):
        return 3
    if(mes == 'April'):
        return 4
    if(mes == 'May'):
        return 5
    if(mes == 'June'):
        return 6
    if(mes == 'July'):
        return 7
    if(mes == 'August'):
        return 8
    if(mes == 'September'):
        return 9
    if(mes == 'October'):
        return 10
    if(mes == 'November'):
```

```

        return 11
    if(mes == 'December'):
        return 12

dados['month_of_reference'] =
dados['month_of_reference'].apply(atribui_numericos_mes)
dados['month_of_reference'] = dados['month_of_reference'].astype('Int64')

ano2022 = dados.loc[dados.year_of_reference == 2022]
y =
ano2022.groupby('month_of_reference').agg({'avg_price_brl':'mean'}).round(0)
)
print(y)

ano2022 = dados.loc[dados.year_of_reference == 2022]
y = ano2022.groupby('month_of_reference')['avg_price_brl'].mean().round(0)

#print(y)
plt.plot(y)
plt.grid()
plt.xticks(range(1,13))
plt.title('da evolução da média de preço dos carros ao longo dos meses de
2022')
plt.ylabel('Média de Preço em reais');
plt.xlabel('Meses de 2022')

plt.show()

media_marcas_cambio =
dados.groupby(['brand','gear'])['avg_price_brl'].mean().round(0)

media_marcas_cambio = media_marcas_cambio.reset_index(name = 'Média de
preços')
#print(media_marcas)

sns.barplot(x='brand', y='Média de preços', hue='gear',
data=media_marcas_cambio, hue_order=['manual', 'automatic'])
plt.grid()
plt.show()

```

```

media_marcas_combustivel =
dados.groupby(['brand', 'fuel'])['avg_price_brl'].mean().round(0)

media_marcas_combustivel = media_marcas_combustivel.reset_index(name =
'Média de preco')
#print(media_marcas)

sns.barplot(x='brand', y='Média de preco', hue='fuel',
data=media_marcas_combustivel, hue_order=['Diesel', 'Gasoline', 'Alcohol'])
plt.grid()
plt.xticks(rotation=45)
plt.show()

#teste da variável target
sns.boxplot(dados['avg_price_brl']).set_title("Distribuição dos preços");

dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])#transformando a
coluna do tipo de combustivel e cambio
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])
dados['brand'] = LabelEncoder().fit_transform(dados['brand'])
dados.head()

dados['engine_size'] = dados['engine_size'].astype('string')
dados['engine_size'] = dados['engine_size'].str.replace(',', '.')
dados['engine_size'] = dados['engine_size'].astype('float64')

#análise de média de preço em relação à potencia do motor
plt.figure(figsize=(15, 5))

potencia_preco =
dados.groupby('engine_size')['avg_price_brl'].mean().round(0)
potencia_preco = potencia_preco.reset_index(name = 'media_precos')
sns.barplot(x='engine_size', y='media_precos', data=potencia_preco)
plt.grid(axis="both")

#plt.xticks(range(1,7))
plt.show()
potencia_preco.head()

```

```

#criação de subdf contendo apenas os dados numericos
dados_num = dados.drop(['fipe_code','authentication','model'],axis = 1)
dados_num.head()

sns.heatmap(dados_num.corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()

#escolha das variaveis independentes e separação da variável target
X = dados_num.drop(['avg_price_brl','month_of_reference'],axis = 1)
Y = dados_num['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 = 42)

model_rf = RandomForestRegressor()
model_rf.fit(X_train, Y_train)

valores_preditos_rf = model_rf.predict(X_test)
valores_preditos_rf

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

mse = mean_squared_error(Y_test, valores_preditos_rf)
mae = mean_absolute_error(Y_test, valores_preditos_rf)
r2_score(Y_test, valores_preditos_rf)

model_xgboost = XGBRegressor()
model_xgboost.fit(X_train, Y_train)

valores_preditos_xgboost = model_xgboost.predict(X_test)
valores_preditos_xgboost

model_xgboost.feature_importances_
feature_importances = pd.DataFrame(model_xgboost.feature_importances_,

```

```
index = X_train.columns, columns=['importance']).sort_values('importance',
ascending = False)
feature_importances

mse = mean_squared_error(Y_test, valores_preditos_xgboost)
mae = mean_absolute_error(Y_test, valores_preditos_xgboost)
r2_score(Y_test, valores_preditos_xgboost)
```

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

- Coeficiente de determinação: R^2

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

onde y_i é o valor observado, $\hat{y}_i$ é o valor predito e $\bar{y}$ é a média dos valores y_i observados.

Quanto mais perto de 1 melhor é o modelo;

- Erro padrão da estimativa: S_{yx}

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

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

- $S_{yx}\%$

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1 - Pesquisa com Dados de Satélite (Satellite)

Base de dados Satellite da biblioteca mlbench. Treino 80% e Teste 20%. Colunas usadas no treino: x17, x18, x19, x20 e classes. Seed = 7.

Resultados obtidos:

FIGURA 4 - MATRIZ DE CONFUSÃO - RANDOM FOREST (ACURÁCIA: 0,8396)

Predição	red soil	cotton crop	grey soil	damp grey soil
red soil	296	1	5	2
cotton crop	0	123	0	0
grey soil	3	0	239	28
damp grey soil	1	0	20	60
vegetation stubble	6	10	0	0
very damp grey soil	0	6	7	35

FIGURA 5 - MATRIZ DE CONFUSÃO - SVM (ACURÁCIA: 0,8715)

Predição	red soil	cotton crop	grey soil	damp grey soil
red soil	299	1	4	2
cotton crop	1	120	0	0
grey soil	4	0	260	29
damp grey soil	0	1	7	69
vegetation stubble	2	14	0	2
very damp grey soil	0	4	0	23

FIGURA 6 - MATRIZ DE CONFUSÃO - RNA (ACURÁCIA: 0,6978)

Predição	red soil	cotton crop	grey soil	damp grey soil
red soil	291	2	1	0
cotton crop	0	118	0	0
grey soil	6	0	252	85
damp grey soil	0	0	0	0
vegetation stubble	0	1	0	0
very damp grey soil	9	19	18	18

O melhor modelo de predição foi obtido a partir do método SVM. Utilizamos a métrica de Acurácia para determinar esse modelo.

2 - Pesquisa com Dados de Satélite (Satellite)

Base de dados utilizada disponível no link <http://www.razer.net.br/datasets/Volumes.csv>, carregamento realizado por meio da função "read.csv2" indicando os parâmetros de separador "," e símbolo decimal ".". Coluna NR eliminada usando a função "subset" indicando os parâmetros do dataframe original e "-" a coluna indesejada. Treino 80% e Teste 20%.

Resultados obtidos com as funções UDF:

Random Forest

- R2: 0,8538019
- Sxy: 0,1445527
- Sxy%: 11,07658
- RMSE: 0,1371347

SVM

- R2: 0,849634
- Sxy: 0,1470481
- Sxy%: 11,2678
- RMSE: 0,1395021

Redes Neurais (NEURAL NET)

- R2: 0,8825988
- Sxy: 0,1295415
- Sxy%: 9,926323
- RMSE: 0,1228938

Redes Neurais (NNET)

- R2: 0
- Sxy: 0,49606
- Sxy%: 38,01139
- RMSE: 0,4706038

Observamos que não é possível executar no mesmo momento o NNET e NEURAL NET. Sendo assim, para obter cada resultado, tivemos que comentar as linhas de código de cada um e rodar em execuções diferentes.

Modelo alométrico de Spurr

- R2: 0,8271824
- Sxy: 0,1574296
- Sxy%: 12,0633
- RMSE: 0,1493509

O modelo escolhido é o de Redes Neurais (NEURALNET), pois apresenta os melhores resultados em todas as métricas propostas. Caso não levemos o modelo Redes Neurais (NEURALNET) em consideração, o melhor modelo observado passa a ser o Random Forest.

CÓDIGO - RESOLUÇÃO PARTE 1

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

data(Satellite) #carregamento da base Satellite do pacote
str(Satellite) #visualizacao dos dados principais

valores_centrais <- Satellite [,c(17,18,19,20,37)]
head(valores_centrais)

set.seed(7)

indices <- createDataPartition(valores_centrais$classes, p=0.8, list=FALSE)
#particao em base de treino 80%

treino <- valores_centrais[indices,]

teste <- valores_centrais[-indices,]

#RANDOM FOREST

rf <- caret::train(classes~., data=treino, method="rf")

predicoes.rf <- predict(rf, teste)

matriz.rf <- caret::confusionMatrix(predicoes.rf, teste$classes)
matriz.rf

#SVM
```

```

svm <- caret::train(classes ~ ., data=treino, method="svmRadial")

predicoes.svm <- predict(svm, teste)

matriz.svm <- caret::confusionMatrix(predicoes.svm, teste$classes)
matriz.svm

#RNA

rna <- caret::train(classes~., data=treino, method="nnet")
predicoes.rna <- predict(rna, teste)

matriz.rna <- caret::confusionMatrix(predicoes.rna, teste$classes)
matriz.rna

```

CÓDIGO - RESOLUÇÃO PARTE 2

```

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

df <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv", header=T,
sep=";", dec=",")

dados <- subset(df, select= -NR)

head(dados)

set.seed(7)

indices <- createDataPartition(dados$VOL, p=0.8, list=FALSE) #particao em
base de treino 80%

treino <- dados[indices,]
teste <- dados[-indices,]

#RANDOM FOREST

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

```

```

predicoes.rf <- predict(rf, teste)

#SVM

svm <- caret::train(VOL ~ ., data=treino, method="svmRadial")

predicoes.svm <- predict(svm, teste)

#REDES NEURAIAS

redeneural <- caret::train(VOL ~ ., data=treino, method="neuralnet")

predicoes.redeneural <- predict(redeneural, teste)

#MODELO ALOMETRICO DE SPURR !!!!!CONFERIR IMPLEMENTAÇÃO!!!!

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

predicoes.alom <- predict(alom, teste)

#####UDFs

#coeficiente de determinação

r2 <- function(valor_real, valor_predito){

  r2 <- 1 - (sum((valor_real - valor_predito)^2)/sum((valor_real -
mean(valor_predito))^2))

  return(r2)
}

#erro padrão da estimativa

syx <- function(valor_real, valor_predito){

  syx <- sqrt(sum((valor_real - valor_predito)^2)/(length(valor_real)-2))

  return(syx)
}

```

```

}

#porcentagem do erro padrao

porsyx <- function(valor_real, valor_predito){

  porsyx <- (syx(valor_real, valor_predito)/mean(valor_real))*100

  return(porsyx)
}

###comparação entre resultados

#rf
r2(teste$VOL, predicoes.rf)
syx(teste$VOL, predicoes.rf)
porsyx(teste$VOL, predicoes.rf)
RMSE(predicoes.rf, teste$VOL)

#svm
r2(teste$VOL, predicoes.svm)
syx(teste$VOL, predicoes.svm)
porsyx(teste$VOL, predicoes.svm)
RMSE(predicoes.svm, teste$VOL)

#redes neurais
r2(teste$VOL, predicoes.redeneural)
syx(teste$VOL, predicoes.redeneural)
porsyx(teste$VOL, predicoes.redeneural)
RMSE(predicoes.redeneural, teste$VOL)

#modelo alometrico de Spurr
r2(teste$VOL, predicoes.alom)
syx(teste$VOL, predicoes.alom)
porsyx(teste$VOL, predicoes.alom)
RMSE(predicoes.alom, teste$VOL)

```

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

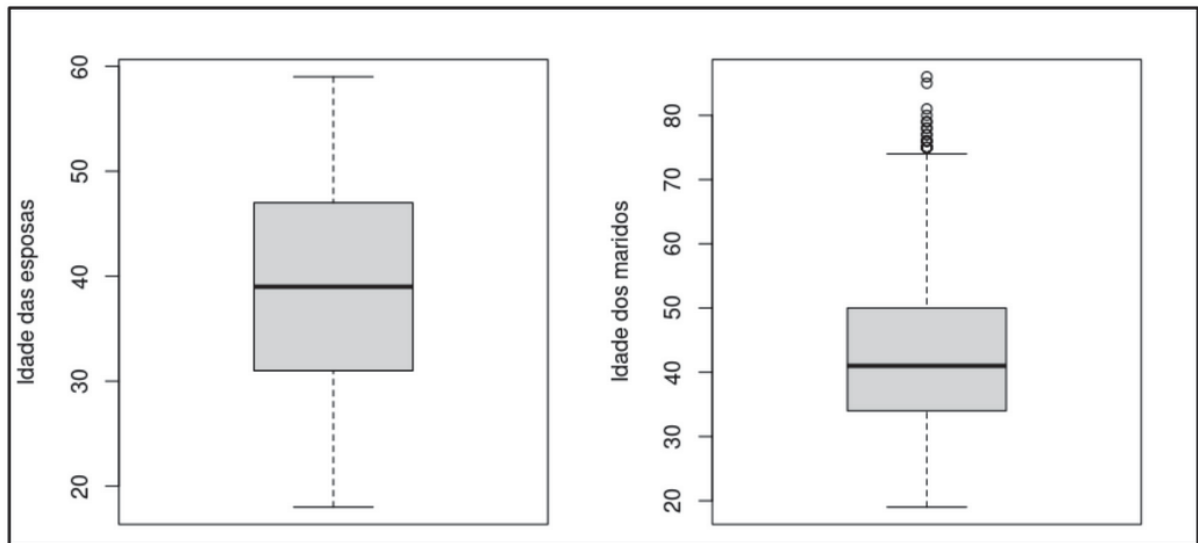
Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

B – RESOLUÇÃO

1 - Gráficos e tabelas

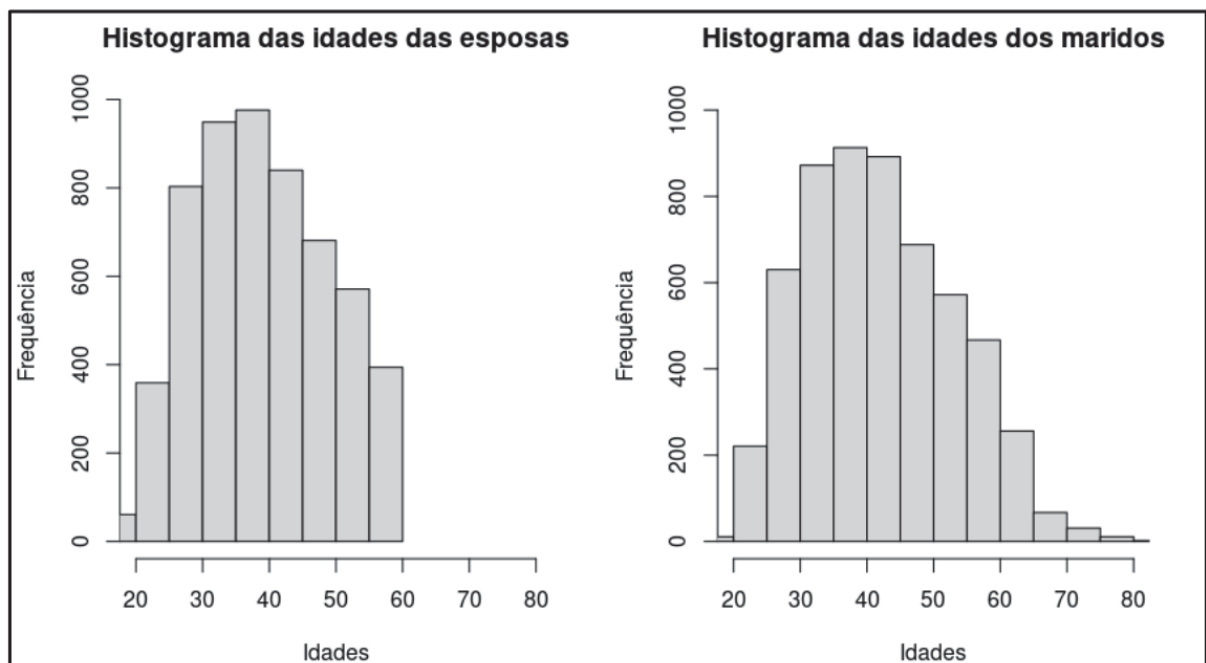
a)

FIGURA 7 - BOXPLOT DAS VARIÁVEIS "AGE" E "HUSAGE"



Ao observar os gráficos do tipo box plot, podemos observar que as idades dos maridos possuem outliers enquanto as idades das esposas não.

FIGURA 8 - HISTOGRAMAS DAS VARIÁVEIS "AGE" E "HUSAGE"



Observando os histogramas da frequência de idades das esposas e maridos podemos concluir que possuem diferentes limites em x.

b)

TABELA 2 - TABELA DE FREQUÊNCIAS DA VARIÁVEL "AGE"

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

TABELA 3 - TABELA DE FREQUÊNCIAS DA VARIÁVEL "HUSAGE"

Classes	Frequência	Frequência relativa	Frequência relativa (%)	Frequência acumulada	Frequência rel. acumulada (%)
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

2 - Medidas de posição e dispersão

a)

TABELA 4 - TABELA DE MÉDIA, MEDIANA E MODA DAS VARIÁVEIS

Medida	Idade da esposa	Idade do marido
Média	39.42758	42.45296
Mediana	39	41
Moda	37 (217 ocorrências)	44 (201 ocorrências)

b)

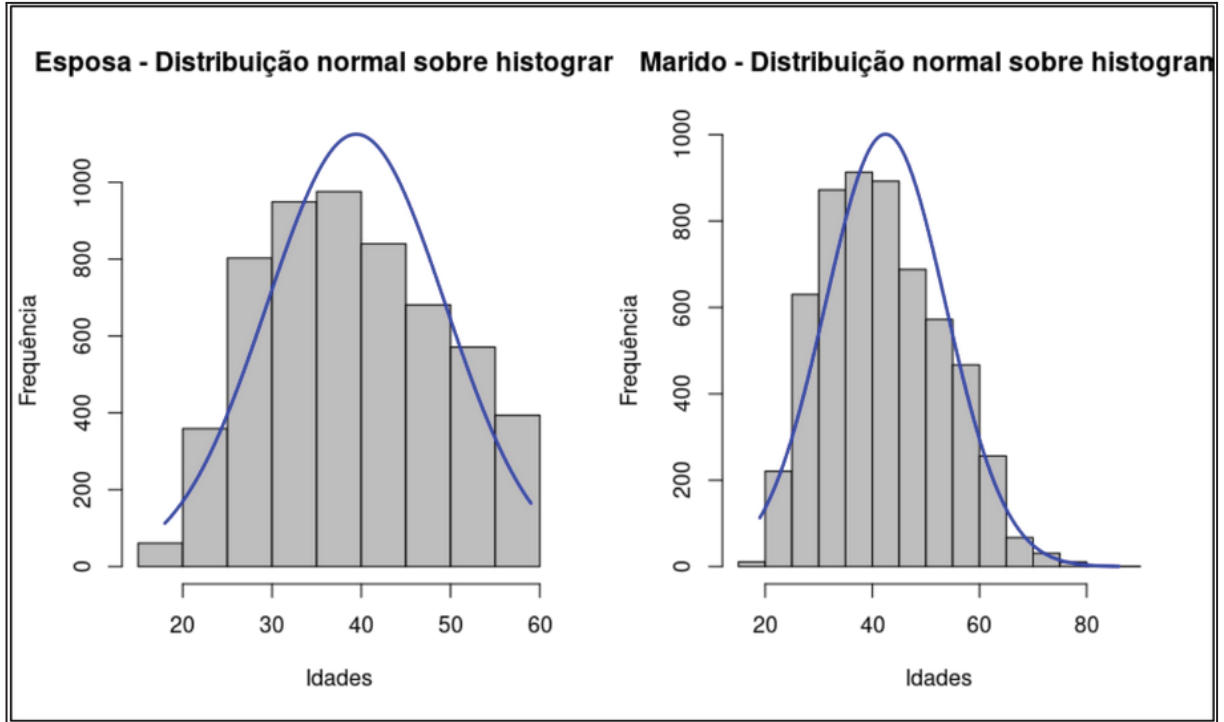
TABELA 5 - TABELA DE VARIÂNCIA, DESVIO PADRÃO E COEFICIENTE DE VARIAÇÃO DAS VARIÁVEIS

Medida	Idade da esposa	Idade do marido
Variância	99.75234	126.0717
Desvio Padrão	9.98761	11.22817
Coeficiente de variação	25.33153	26.44849

3 - Testes paramétricos ou não paramétricos

a)

FIGURA 9 - DISTRIBUIÇÃO NORMAL SOBRE O HISTOGRAMA



Para decidir se as médias serão testadas com testes paramétricos ou não paramétricos verificamos pelos métodos Kolmogorov-Smirnov e Shapiro-Wilk se a distribuição dos dados das variáveis analisadas são normais:

FIGURA 10 - KOLMOGOROV-SMIRNOV PARA "SALARIOS@AGE"

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  salarios$age
D = 0.058909, p-value < 0.00000000000000022
```

FIGURA 11 - KOLMOGOROV-SMIRNOV PARA "SALARIOS@HUSAGE"

```
Lilliefors (Kolmogorov-Smirnov) normality test
data:  salarios$husage
D = 0.059662, p-value < 0.00000000000000022
```

FIGURA 12 - SHAPIRO-WILK PARA NAME = "WIFE"

```

Shapiro-Wilk normality test

data:  age[name == "WIFE"][0:5000]
W = 0.97695, p-value < 0.000000000000000022

```

FIGURA 13 - SHAPIRO-WILK PARA NAME = "HUSBAND"

```

Shapiro-Wilk normality test

data:  age[name == "HUSBAND"][0:5000]
W = 0.98151, p-value < 0.000000000000000022

```

Por ambos os testes, é possível observar que p-value é inferior ao nível de significância 0,05, portanto os dados a serem comparados não são normais, demandando assim a abordagem por meio de testes não-paramétricos.

- H0: A idade mediana dos maridos é estatisticamente igual à mediana da idade das esposas.
- H1: A idade mediana dos maridos é estatisticamente diferente à mediana da idade das esposas.

O teste usado é o U de Mann-Whitney, que gerou o resultado abaixo:

FIGURA 14 - TESTE U DE MANN-WHITNEY - ESTATISTICAMENTE IGUAL

```

Wilcoxon rank sum test with continuity correction

data:  age by name
W = 18122044, p-value < 0.000000000000000022
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 2.000033 3.000024
sample estimates:
difference in location
      2.999966

```

Ao realizar o teste U de Mann-Whitney é possível verificar que as medianas das duas variáveis não são estatisticamente iguais, pois p-value inferior ao nível de significância 0,05, logo, rejeitamos H0.

O intervalo de confiança da diferença entre as medianas está entre 2,00 e 3,00, com uma mediana de 2,99.

- H0: A idade mediana dos maridos não é estatisticamente menor à mediana da idade das esposas.
- H1: A idade mediana dos maridos é estatisticamente menor à mediana da idade das esposas.

FIGURA 15 - TESTE U DE MANN-WHITNEY - ESTATISTICAMENTE MENOR

```
> #testando se a mediana das idades das esposas é estatisticamente menor
> wilcox.test(age ~ name, data = allAges,
+             exact = FALSE, alternative = "less",
+             conf.int=TRUE)

      wilcoxon rank sum test with continuity correction

data:  age by name
W = 18122044, p-value = 1
alternative hypothesis: true location shift is less than 0
95 percent confidence interval:
 -Inf 3.000034
sample estimates:
difference in location
      2.999966
```

Ao testar se a mediana do marido não é estatisticamente menor que a mediana das esposas, não rejeitamos H0, e verificamos que a mediana da idade dos maridos é estatisticamente maior que a mediana da idade das esposas, pois p-value foi superior a 0,05, com um intervalo de confiança entre -Inf e 3, e mediana 2,99.

Para confirmar a hipótese encontrada, realiza o teste para verificar se a mediana da idade dos maridos não é estatisticamente maior que a mediana da idade das esposas, caso o p-value fique abaixo de 0,05, confirmamos que, a mediana da idade dos maridos é estatisticamente maior à mediana da idade das esposas.

- H0: A idade mediana da idade dos maridos não é estatisticamente maior à mediana da idade das esposas.
- H1: A idade mediana da idade dos maridos é estatisticamente maior à mediana da idade das esposas.

FIGURA 16 - TESTE U DE MANN-WHITNEY - ESTATISTICAMENTE MAIOR

```

wilcoxon rank sum test with continuity correction

data:  age by name
w = 18122044, p-value < 0.000000000000000022
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 2.000046      Inf
sample estimates:
difference in location
      2.999966

```

Rejeitamos H_0 , e confirmamos que a mediana da idade dos maridos é estatisticamente maior que a mediana da idade das esposas.

CÓDIGO

```

library(fdth)
library(dplyr)
library(rstatix)

#setwd("D:/GIT/IAA004-TrabalhoFinal/")
load("salarios.RData")
View(salarios)

#####
#               1 - Graficos e Tabelas               #
#####
#

# Gerando o boxplot para a variavel "age" e "husage"
par(mfrow=c(1,2))

Boxplot( ~ age, data=salarios, id=list(method="identify"),
        ylab="Idade das esposas")

Boxplot( ~ husage, data=salarios, id=list(method="identify"),
        ylab="Idade dos maridos")

# Aqui vemos varios possiveis outliers na amostra, idades dos
# maridos muito elevadas

```

```
# Gerando o histograma para a variavel "age" e "husage"
par(mfrow=c(1,2))

hist(salarios$age, main = "Histograma das idades das esposas",
      xlim = c(20, 80), ylab = "Frequência", xlab="Idades")

hist(salarios$husage, main = "Histograma das idades dos maridos",
      xlim = c(20, 80), ylim = c(0, 1000), ylab = "Frequência",
      xlab="Idades")
```

```
#          Tabela de Distribuicao de frequencia          #
```

```
# Carregando pacote que calcula a distribuicao de frequencia
```

```
# Calculando a distribuicao de frequencia e guardando no objeto
```

```
# "table"
```

```
table_hus <- fdt(salarios$husage)
```

```
table <- fdt(salarios$age)
```

```
# Mostrando a distribuicao de frequencia
```

```
print (table)
```

```
print(table_hus)
```

```
#####
```

```
#          2 - Medidas de posição e dispersão          #
```

```
#####
```

```
#-----
```

```
#Calculo da media
```

```
mean(salarios$husage)
```

```
mean(salarios$age)

#Calculo da mediana
median(salarios$husage)
median(salarios$age)

#Calculo da frequencia da media
table(salarios$husage)
subset(table(salarios$husage),
        table(salarios$husage) ==
        max(table(salarios$husage)))

table(salarios$age)
subset(table(salarios$age),
        table(salarios$age) == max(table(salarios$age)))

#-----
#Letra B

# Calculando a variancia da idade das esposas
var(salarios$age)

# Calculando a variancia da idade do maridos
var(salarios$husage)

# Calculando o desvio padrao da idade das esposas:
sd(salarios$age)

# Calculando o desvio padrao da idade dos maridos:
sd(salarios$husage)

# Calculando a media das idades das esposas e guardando no
# objeto "meanE"
meanE <- mean(salarios$age)

# Calculando a media das idades dos maridos e guardando no
# objeto "meanM"
meanM <- mean(salarios$husage)
```

```

# Calculando o desvio padrao das idades das esposas e
# guardando no objeto sdE
sdE <- sd(salarios$age)

# Calculando o desvio padrao das idades dos maridos e
# guardando no objeto sdM
sdM <- sd(salarios$husage)

# Calculando o coeficiente de variacao das idades das
# esposas:
cvE <- (sdE/meanE)*100
cvE
# O coeficiente de variacao da idade das esposas eh 25,33%

# Calculando o coeficiente de variacao das idades dos
# maridos:
cvM <- (sdM/meanM)*100
cvM
# O coeficiente de variacao da idade dos maridos eh 26,45%

#####
#           3 - Testes paramétricos ou não paramétricos           #
#####
options(scipen = 999)

## CONSTRUINDO DATAFRAMES APENAS COM O TIPO E A IDADE
wifeAges <- data.frame(name='WIFE', age=salarios$age)
husAges <- data.frame(name='HUSBAND', age=salarios$husage)
allAges <- rbind(wifeAges, husAges)

#Ordenando os dados pela coluna age
allAges <- allAges[order(allAges$age, decreasing = FALSE),]

#Calcular a média da idade da mulher e do marido
mean(salarios$age)
mean(salarios$husage)

```

```

# Calculando as estatísticas descritivas por grupo, Homem e
# Mulher:

allAges %>% group_by(name) %>%
  get_summary_stats(age, type = "mean_sd")

#Calculando desvio padrao de toda a amostra

sd(allAges$age)

#hist(idades$idade)
hist(salarios$age)

# Vamos plotar a curva normal sobre o histograma:
plotNormalHistogram(allAges$age, prob = FALSE,
                     main = "Normal Distribution overlay on Histogram",
                     length = 1000 )

# Vamos plotar a curva normal sobre os histogramas de "age" e "husage":
par(mfrow=c(1,2))

plotNormalHistogram(salarios$age, prob = FALSE,
                     main = "Esposa - Distribuição normal sobre histograma",
                     length = 1000, ylab = "Frequência", xlab="Idades")

plotNormalHistogram(salarios$husage, prob = FALSE,
                     main = "Marido - Distribuição normal sobre histograma",
                     length = 1000 , ylab = "Frequência", xlab="Idades")

# Executando teste de normalidade de Kolmogorov-smirnov:
lillie.test(salarios$age) # p-value = 0,0000000000000022 ~ 0 - Não é uma
distribuição normal
lillie.test(salarios$husage) # p-value = 0,0000000000000022 ~ 0 - Não é uma
distribuição normal

lillie.test(allAges$age) # p-value = 0,0000000000000022 ~ 0 - Não é normal

# teste de normalidade Shapiro-Wilk
options(scipen = 999)

```

```

print(with(wifeAges, shapiro.test(age[name == "WIFE"][0:5000]))) )
# p-value < 0.000000000000000022
print(with(husAges, shapiro.test(age[name == "HUSBAND"][0:5000]))) )
# p-value < 0.000000000000000022

#JarqueBeraTest
jarque.bera.test(salarios$age)
jarque.bera.test(salarios$husage)
jarque.bera.test(allAges$age)

#O dataset não é normal
#Conclui-se que as variáveis não são normais, deve-se utilizar testes não
paramétricos

#H0: A idade mediana das esposas é estatisticamente igual à mediana da
idade dos maridos.
#H1: A idade mediana das esposas é estatisticamente diferente à mediana da
idade dos maridos.

# UTILIZANDO U DE MANN-WHITNEY POIS OS GRUPOS NAO POSSUEM DISTRIBUICAO
NORMAL

# sumario estatistico

allAges %>% group_by(name) %>%
  get_summary_stats(age, type = "median_iqr")

#teste U de mann-whitney

print(wilcox.test(age ~ name, data = allAges,
                  exact = FALSE, conf.int=TRUE))

# O p-value do teste eh 0.000000000000000022, que eh menor que o nivel de
# significancia 0,05. Podemos concluir que a idade mediana dos
# homens eh estatisticamente diferente da idade mediana das
# mulheres (rejeitamos H0).
# O intervalo de confianca da diferenca entre as medianas esta

```

```
# entre 2,00 e 3,00, com uma mediana de 2,99.

#testando se a mediana das idades das esposas e estatisticamente menor
wilcox.test(age ~ name, data = allAges,
             exact = FALSE, alternative = "less",
             conf.int=TRUE)

# O p-value do teste eh 0.1, que eh maior que o nivel de
# significancia 0,05. Podemos concluir que a idade mediana dos
# homens eh estatisticamente maior da idade mediana das
# mulheres (rejeitamos H0 e confirmamos H1).

#Para confirmar o resultado obtido na etapa anterior, testar se a idade das
#esposas é estatisticamente maior que a dos maridos
#H0 é Estatisticamente Maior
#H1 Não é estatisticamente Maior
#P-value esperado = abaixo de 0,05
wilcox.test(age ~ name, data = allAges,
             exact = FALSE, alternative = "greater",
             conf.int=TRUE)
```

APÊNDICE E - ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

TABELA 6 - TABELA DE REGRESSÕES

	RMSE	R^2
Treino Ridge	0.8487758	0.2792296
Teste Ridge	0.8489331	0.3002941
Treino Lasso	0.8498015	0.2774865
Teste Lasso	0.8491485	0.2999389
Treino Elasticnet	0.8499146	0.2772942
Teste Elasticnet	0.8497128	0.2990082

Observamos que todas as regressões tiveram valores parecidos, mas mesmo assim escolhemos o padrão Ridge por apresentar a menor diferença dos erros entre o treino e teste e melhores valores nos erros se comparados com as outras regressões.

O resultado da predição usando o Ridge para os seguintes valores passados no exercício e depois de aplicar o antilog são:

- Resultado da predição: 8.346
- Intervalos de confiança: Entre 8.324 e 8.369

CÓDIGO

```
# install.packages("plyr")
# install.packages("readr")
# install.packages("dplyr")
# install.packages("ggplot2")
# install.packages("repr")
# install.packages("glmnet")
# install.packages("caret")
# install.packages("psych")

library(plyr)
library(readr)
library(dplyr)
library(ggplot2)
library(repr)
library(glmnet)
library(caret)
library(psych)

# Carregando o dataset
load("trabalhosalarios.RData")
```

```

dat <- trabalhosalarios

# retirar a coluna "earns" da variável dat
dat <- dat[,-which(names(dat) == "earns")]

# Vamos visualizar parte do dataset
glimpse(sal)
set.seed(42)

#####
## PREPARAÇÃO DA BASE DE DADOS ##
#####

# Vamos criar um índice para particionar o dataset em
# 80% para treinamento
index = sample(1:nrow(dat),0.8*nrow(dat))

# Vamos criar a base de dados de treinamento e teste
train = dat[index,]
test = dat[-index,]
dim(train)
dim(test)

# Vamos padronizar as variáveis não binárias
cols = c('husage', 'husearns', 'huseduc', 'hushrs',
'age', 'educ', 'exper','lwage')

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

# Vamos ver o sumário estatístico das variáveis padronizadas de cada
dataset
summary(train)
summary(test)

#####
## PREPARAÇÃO COMUM A TODAS AS TÉCNICAS ##

```

```
#####

# Vamos criar um objeto com as variaveis que usaremos no
# modelo
cols_reg = c('husage','husunion','husearns','huseduc','husblck','hushisp',
'hushrs','kidge6','age','black','educ','hispanic','union',
'exper','kidlt6','lwage')

# Vamos gerar variaveis dummies para organizar os datasets
# em objetos tipo matriz
# Estamos interessados em estimar o salario-hora da esposa (lwage)
dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+
husblck+hushisp+hushrs+kidge6+age+black+
educ+hispanic+union+exper+kidlt6,
data = dat[,cols_reg])
train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])
print(dim(train_dummies)); print(dim(test_dummies))

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

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

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

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

#####
## Regressão Ridge ##
```

```
#####

Vamos calcular o valor otimo de lambda;
# alpha = "0", eh para regressao Ridge
# Vamos testar os lambdas de 10^-3 ate 10^2, a cada 0.1
lambdas <- 10^seq(2, -3, by = -.1)

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

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

# Estimando o modelo Ridge
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
family = 'gaussian',
lambda = best_lambda_ridge)
# Vamos ver o resultado (valores) da estimativa
# (coeficientes)
ridge_reg[["beta"]]

# 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
)
}

# 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:
res_treino_ridge <- eval_results(y_train, predictions_train, train)
res_treino_ridge

# 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:
res_teste_ridge <- eval_results(y_test, predictions_test, test)
res_teste_ridge

#####
## Regressão Lasso ##
#####

# Vamos criar o intervalo para o tuning do melhor lambda
lambdas <- 10^seq(2, -3, by = -.1)

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

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

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

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

```

```

# Perceba que alguns coeficientes estao zerados pois
# nao sao significativos
# Vamos fazer as predicoes na base de treinamento e
# avaliar a regressao Lasso
predictions_train <- predict(lasso_model,
s = best_lambda_lasso,
newx = x)

# Vamos calcular o R^2 dos valores verdadeiros e
# preditos conforme a seguinte funcao:
eval_results <- function(true, predicted, df) {
SSE <- sum((predicted - true)^2)
SST <- sum((true - mean(true))^2)
R_square <- 1 - SSE / SST
RMSE = sqrt(SSE/nrow(df))
# As metricas de performace do modelo:
data.frame(
RMSE = RMSE,
Rsquare = R_square
)
}

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

# 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:
res_teste_lasso <- eval_results(y_test, predictions_test, test)
res_teste_lasso

#####
## 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 = 10,
repeats = 5,
search = "random",
verboseIter = TRUE)

# Vamos treinar o modelo
elastic_reg <- train(lwage~husage+husunion+husearns+huseduc+
husblk+hushisp+hushrs+kidge6+age+black+
educ+hispanic+union+exper+kidlt6,
data = train,
method = "glmnet",
tuneLength = 10,
trControl = train_cont)

# O melhor parametro alpha escolhido eh:
elastic_reg$bestTune

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

# Vamos fazer as predicoes e avaliar a performance do
# modelo
# Vamos fazer as predicoes no modelo de treinamento:
predictions_train <- predict(elastic_reg, x)

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

```

```

data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)
}

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

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

# As metricas de performance na base de teste sao:
res_teste_elastic <-eval_results(y_test, predictions_test, test)
res_teste_elastic

#####
## Predições com o nosso exemplo ##
#####

# Vamos fazer uma predicao para:
# husage = 40 anos (idade do marido)
husage = (40-pre_proc_val[["mean"]][["husage"]])/
pre_proc_val[["std"]][["husage"]]

#husunion = 0 (marido não possui união estável)
husunion = 0

# husearns = 600 (rendimento do marido em US$)
husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
pre_proc_val[["std"]][["husearns"]]

# huseduc = 13 (anos de estudo do marido)
huseduc = (13-pre_proc_val[["mean"]][["huseduc"]])/
pre_proc_val[["std"]][["huseduc"]]

# husblck = 1 (o marido eh preto)
husblck = 1

```

```

# hushisp = 0 (o marido nao eh hispanico)
hushisp = 0

# hushrs = 40 (o marido trabalha 40 horas semanais)
hushrs = (40-pre_proc_val[["mean"]][["hushrs"]])/
pre_proc_val[["std"]][["hushrs"]]

# kidge6 = 1 (possui filhos maiores de 6 anos)
kidge6 = 1

# age = 38 anos (idade da esposa)
age = (38-pre_proc_val[["mean"]][["age"]])/
pre_proc_val[["std"]][["age"]]

# black = 0 (esposa nao eh preta)
black = 0

# educ = 13 (esposa possui 13 anos de estudo)

educ = (13-pre_proc_val[["mean"]][["educ"]])/
pre_proc_val[["std"]][["educ"]]

# hispanic = 1 (esposa eh hispanica)
hispanic = 1

# union = 0 (o casal nao possui uniao registrada)
union = 0

# exper = 18 (anos de experiência de trabalho da esposa)
exper = (18-pre_proc_val[["mean"]][["exper"]])/
pre_proc_val[["std"]][["exper"]]

# kidlt6 = 1 (possui filhos com menos de 6 anos)
kidlt6 = 1

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

```

```

huseduc=huseduc,
husblk=husblk,
hushisp=hushisp,
hushrs=hushrs,
kidge6=kidge6,
age=age,
black=black,
educ=educ,
hispanic=hispanic,
union=union,
exper=exper,
kidlt6=kidlt6))

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

# O resultado da predicao eh:
predict_our_ridge
# O resultado eh um valor padronizado, vamos converte-lo
# para o valor nominal, consistente com o dataset original
log_pred_ridge=(predict_our_ridge*pre_proc_val[["std"]][["lwage"]])+
pre_proc_val[["mean"]][["lwage"]]
# depois reverteremos o log:
wage_pred_ridge <- exp(log_pred_ridge)

# O resultado eh:
wage_pred_ridge

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

# Os valores sao:

```

```

CIlwr_ridge
CIupr_ridge

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

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

# depois revertemos o log:
wage_pred_lasso <- exp(log_pred_lasso)
wage_pred_lasso

# Vamos criar o intervalo de confianca para o nosso
# exemplo
n <- nrow(train)
m <- log_pred_lasso
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_lasso <- exp(m) + (qnorm(0.025))*dam
CIupr_lasso <- exp(m) - (qnorm(0.025))*dam

# O intervalo de confianca eh:
CIlwr_lasso
CIupr_lasso

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

# Novamente, o resultado eh padronizado, nos temos que

```

```

# reverte-lo para o nivel dos valores originais do
# dataset, vamos fazer isso:
log_pred_elastic=(predict_our_elastic*
pre_proc_val[["std"]][["lwage"]])+
pre_proc_val[["mean"]][["lwage"]]
wage_pred_elastic <- exp(log_pred_elastic)
wage_pred_elastic

# Vamos criar o intervalo de confianca para o nosso
# exemplo
n <- nrow(train)
m <- log_pred_elastic
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_elastic <- exp(m) + (qnorm(0.025))*dam
CIupr_elastic <- exp(m) - (qnorm(0.025))*dam

# Os valores minimo e maximo sao:
CIlwr_elastic
CIupr_elastic

# criando um dataframe com as respostas dos testes:
df_resumo <- rbind(res_treino_ridge, res_teste_ridge,
res_treino_lasso, res_teste_lasso,
res_treino_elastic, res_teste_elastic)
rownames(df_resumo) <- c("Treino_ridge", "Teste_ridge",
"Treino_lasso", "Teste_lasso",
"Treino_elastic", "Teste_elastic")

df_resumo

```

APÊNDICE F - ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

Tarefa 1 - Construção de Características: Identificador automático de idioma

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)
Identificador automático de idioma

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

```

"I enjoy going to the beach.",
"Where can I find a taxi?",
"I'm sorry for the inconvenience.",
"I'm studying for my exams.",
"I like to cook dinner at home.",
"Do you have any recommendations for restaurants?",
]
espanhol = [
"Hola, ¿cómo estás?",
"Me encanta leer libros.",
"El clima está agradable hoy.",
"¿Dónde está el restaurante más cercano?",
"¿Qué hora es?",
"Voy al parque todos los días.",
"¿Puedes ayudarme con esto?",
"Me gustaría ir de vacaciones.",
"Este es mi libro favorito.",
"Me gusta bailar salsa.",
"¿Hablas español?",
"¿Cuál es tu comida favorita?",
"Estoy aprendiendo a tocar el piano.",
"¡Que tengas un buen día!",
"Necesito comprar algunas frutas.",
"Vamos a dar un paseo.",
"¿Cómo estuvo tu fin de semana?",
"Estoy emocionado por el concierto.",
"¿Me pasas la sal, por favor?",
"Tengo una reunión a las 2 PM.",
"Estoy planeando unas vacaciones.",
"Ella canta hermosamente.",
"El perro está jugando.",

El perro está jugando. ,
"Quiero aprender italiano.",
"Disfruto ir a la playa.",
"¿Dónde puedo encontrar un taxi?",
"Lamento las molestias.",
"Estoy estudiando para mis exámenes.",
"Me gusta cocinar la cena en casa.",
"¿Tienes alguna recomendación de restaurantes?",

```

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

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

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

```
import random

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

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

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

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

[['Estou aprendendo a cozinhar pratos novos.', 'português'], ['Vamos fazer um churrasco no final de semana.', 'português'], ['Tenho uma reunião importante amanhã.', 'português'], ['Me encanta leer libros.', 'espanhol'], ['¿Qué hora es?', 'espanhol'], ["I'm studying for my exams.", 'inglês'], ['Gosto de ouvir música clássica.', 'português'], ['Hola, ¿cómo estás?', 'espanhol'], ['Preciso comprar leite e pão.', 'português'], ['I enjoy going to the beach.', 'inglês'], ['Hello, how are you?', 'inglês'], ['Do you speak English?', 'inglês'], ['Adoro explorar novos lugares.', 'português'], ['A comida estava deliciosa!', 'português'], ['Vamos fazer uma viagem de carro.', 'português'], ['Me gusta bailar salsa.', 'espanhol'], ['O filme que assisti ontem foi ótimo.', 'português'], ['Adoro passar tempo com minha família.', 'português'], ['Preciso resolver esse problema o mais rápido possível.', 'português'], ['Necesito comprar algunas frutas.', 'espanhol'], ['¿Tienes alguna recomendación de restaurantes?', 'espanhol'], ['Você já visitou o Rio de Janeiro?', 'português'], ['I need to buy some groceries.', 'inglês'], ['How was your weekend?', 'inglês'], ["I'm sorry for the inconvenience.", 'inglês'], ['Lamento las molestias.', 'espanhol'], ['I enjoy playing soccer.', 'inglês'], ["Let's go for a walk.", 'inglês'], ['¿Dónde puedo encontrar un taxi?', 'espanhol'], ['A festa começa às 20h.', 'português'], ['Estou indo para o trabalho agora.', 'português'], ['Estou ansioso para as férias de verão.', 'português'], ['Preciso fazer exercícios

físicos regularmente.', 'português'], ['O parque fica cheio aos finais de semana.', 'português'], ['I'm excited for the concert.', 'inglês'], ['¿Que tengas un buen día!', 'espanhol'], ['Where is the nearest restaurant?', 'inglês'], ['The cat is sleeping.', 'inglês'], ['Vamos a dar un paseo.', 'espanhol'], ['I'm going to the movies tonight.', 'inglês'], ['Vamos ao cinema no sábado.', 'português'], ['Quiero aprender italiano.', 'espanhol'], ['Quero aprender a tocar violão.', 'português'], ['I like listening to music.', 'inglês'], ['Me gusta cocinar la cena en casa.', 'espanhol'], ['Have a great day!', 'inglês'], ['Do you have any recommendations for restaurants?', 'inglês'], ['Estou cansado depois de um longo dia de trabalho.', 'português'], ['El perro está jugando.', 'espanhol'], ['Gosto de praticar esportes ao ar livre.', 'português'], ['Estoy emocionado por el concierto.', 'espanhol'], ['I want to learn French.', 'inglês'], ['O trânsito está terrível hoje.', 'português'], ['Gosto de fazer caminhadas na natureza.', 'português'], ['Can you help me with this?', 'inglês'], ['I like to cook dinner at home.', 'inglês'], ['Voy al parque todos los días.', 'espanhol'], ['I'm learning to play the guitar.', 'inglês'], ['Tengo una reunión a las 2 PM.', 'espanhol'], ['The weather is nice today.', 'inglês'], ['¿Cuál es tu comida favorita?', 'espanhol'], ['Hoje é meu aniversário!', 'português'], ['I love to read books.', 'inglês'], ['What is your favorite color?', 'inglês'], ['She sings beautifully.', 'inglês'], ['Vamos sair para jantar no sábado.', 'português'], ['O livro que estou lendo é muito interessante.', 'português'], ['Este es mi libro favorito.', 'espanhol'], ['¿Puedes ayudarme con esto?', 'espanhol'], ['I'm planning a vacation.', 'inglês'], ['Vou visitar meus avós no domingo.', 'português'], ['Disfruto ir a la playa.', 'espanhol'], ['Estou estudando para o vestibular.', 'português'], ['Vou viajar para o exterior nas férias.', 'português'], ['Me gustaría ir de vacaciones.', 'espanhol'], ['I have a meeting at 2 PM.', 'inglês'], ['Could you pass me the salt, please?', 'inglês'], ['Estoy estudiando para mis exámenes.', 'espanhol'], ['Estoy aprendiendo a tocar el piano.', 'espanhol'], ['This is a beautiful place.', 'inglês'], ['¿Cómo estuvo tu fin de semana?', 'espanhol'], ['Meu time de futebol favorito ganhou o jogo.', 'português'], ['Estoy planeando unas vacaciones.', 'espanhol'], ['¿Hablas español?', 'espanhol'], ['El clima está agradable hoy.', 'espanhol'], ['¿Dónde está el restaurante más cercano?', 'espanhol'], ['Você gosta de dançar?', 'português'], ['Ella canta hermosamente.', 'espanhol'], ['¿Me pasas la sal, por favor?', 'espanhol'], ['O restaurante tem uma vista incrível.', 'português'], ['What time is it?', 'inglês'], ['Where can I find a taxi?', 'inglês']]

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

FIGURA 17 - DATAFRAME DOS PRÉ-PADRÕES

	0	1
0	Do you speak English?	inglês
1	Gosto de ouvir música clássica.	português
2	É importante manter a mente aberta.	português
3	Have a great day!	inglês
4	O que você está fazendo?	português
...	...	...
115	Onde você quer ir hoje?	português
116	Me encanta leer libros.	espanhol
117	Meu time de futebol favorito ganhou o jogo.	português
118	I'm studying for my exams.	inglês
119	Você acha que está quente demais para ir camin...	português
120 rows x 2 columns		

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

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

```
# a entrada é o vetor pre_padroes e a saída desse passo deverá ser
"padrões"
import re

def tamanhoMedioFrases(texto):
    palavras = re.split(r"\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 contar_consoantes(texto):
    # retorna a quantidade de consoantes em uma frase
    padrao_consoantes = r'[bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ]'
    #padrão regex dos elementos que serão contados

    consoantes = re.findall(padrao_consoantes, texto)

    return len(consoantes)

def contar_acentuados(texto):

    #retorna o numero de caracteres com acentos gráficos
    padrao_acentos =
r'[áéíóúàèìòûâêîôûãõäëïöüñçÁÉÍÓÚÀÈÌÒÛÂÊÎÔÛÃÕÄËÏÖÜÇÑ]'#padrão regex dos
elementos que serão contados

    acentos = re.findall(padrao_acentos, texto)

    return len(acentos)

def contar_dupla_vogal(texto):
    padrao_duplas_vogais = r'(?i)[aeiouy]{2}'

    duplas_vogais = re.findall(padrao_duplas_vogais, texto)

    return len(duplas_vogais)

def contar_termino_palavras(texto):
    padrao_terminos = r'\b\w*(?:m|an|ón|ing)\b'
    terminos = re.findall(padrao_terminos, texto, re.IGNORECASE)
    return len(terminos)

```

```

def contar_pontuacao(texto):
    padrao_pontuacao = r'\w(?:\'|ç|!|)'
    pontuacao = re.findall(padrao_pontuacao, texto)

    return len(pontuacao)

def caracteristicas_ingles(texto):
    padrao_ingles = r'\w(?:\'|wh|th|ing|oo|ee|nn)'
    pd_ingles = re.findall(padrao_ingles, texto)

    return len(pd_ingles)

def caracteristicas_espanhol(texto):
    padrao_espanhol = r'\w(?:ç|el|la|ue|ie|ón|an|oy|ay)'
    pd_espanhol = re.findall(padrao_espanhol, texto)

    return len(pd_espanhol)

def caracteristicas_portugues(texto):
    padrao_portugues = r'\w(?:nh|lh|rr|ss|am)'
    pd_portugues = re.findall(padrao_portugues, texto)

    return len(pd_portugues)

def contar_ngramas(texto):
    #Retorna um dicionário com a frequência de bigramas de letras.
    ngramas = {}
    texto_limpo = texto.lower().replace(" ", "")

    for i in range(len(texto_limpo) - 1):
        bigrama = texto_limpo[i:i+2]
        if bigrama in ngramas:
            ngramas[bigrama] += 1
        else:
            ngramas[bigrama] = 1

    return ngramas

def contar_sufixos(texto):

```

```

#Conta a quantidade de sufixos comuns em inglês e espanhol.

padrao_sufixos_ingles = r'\b\w*(?:tion|ment|ness|ly)\b'
sufixos_ingles = len(re.findall(padrao_sufixos_ingles, texto,
re.IGNORECASE))

padrao_sufixos_espanhol = r'\b\w*(?:mente|dad|cion)\b'
sufixos_espanhol = len(re.findall(padrao_sufixos_espanhol, texto,
re.IGNORECASE))

return sufixos_ingles + sufixos_espanhol

def comprimentoMedioPalavras(texto):
    #Calcula e retorna o comprimento médio das palavras em um texto.

    palavras = re.findall(r'\b\w+\b', texto)
    if not palavras:
        return 0
    tamanhos = [len(palavra) for palavra in palavras]
    return sum(tamanhos) / len(tamanhos)

def extraiCaracteristicas(frase):
    # frase é um vetor [ 'texto', 'lingua' ]
    texto = frase[0]
    pattern_regex = re.compile(r'^\w+', re.UNICODE)
    texto = re.sub(pattern_regex, ' ', texto)
    #print(texto)
    caracteristica1=tamanhoMedioFrases(texto)
    caracteristica2=contar_consoantes(texto)
    caracteristica3=contar_acentuados(texto)
    caracteristica4=caracteristicas_ingles(texto)
    caracteristica5=caracteristicas_espanhol(texto)
    caracteristica6=caracteristicas_portugues(texto)
    #caracteristica7=contar_dupla_vogal(texto)
    #caracteristica8=contar_termino_palavras(texto)
    caracteristica9=contar_pontuacao(texto)
    caracteristica10=comprimentoMedioPalavras(texto)
    #caracteristica11=contar_ngramas(texto)
    caracteristica12=contar_sufixos(texto)
    # acrescente as suas funcoes no vetor padrao

```

```

    padrao = [caracteristica1, caracteristica2, caracteristica3,
caracteristica4, caracteristica5,caracteristica6, caracteristica9,
caracteristica10, caracteristica12, frase[1]]
    return padrao

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

import numpy as np

padroes_brutos = geraPadroes(pre_padroes)

# Converte a lista de listas para um array numpy para facilitar os cálculos
padroes_array = np.array([p[:-1] for p in padroes_brutos])
classes = [p[-1] for p in padroes_brutos]

minimos = np.min(padroes_array, axis=0)
maximos = np.max(padroes_array, axis=0)

padroes_normalizados = []
for i, padrao in enumerate(padroes_array):
    padrao_normalizado = []
    for j in range(len(padrao)):
        if (maximos[j] - minimos[j]) != 0:
            normalized_value = (padrao[j] - minimos[j]) / (maximos[j] -
minimos[j])
        else:
            normalized_value = 0 # Or another appropriate value, like 0
        padrao_normalizado.append(normalized_value)
    padroes_normalizados.append(padrao_normalizado + [classes[i]])

# Agora a variável 'padroes_normalizados' contém o conjunto de dados pronto
para o modelo

```

Treinando o modelo com SVM.

```

from sklearn.model_selection import train_test_split
import numpy as np

#from sklearn.metrics import confusion_matrix

vet = np.array(padroes_normalizados)
classes = vet[:, -1]          # classes = [p[-1] for p in padroes]
#print(classes)
padroes_sem_classe = vet[:, 0:-1]
#print(padroes_sem_classe)
X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe,
                                                    classes, test_size=0.2, stratify=classes)

from sklearn import svm
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, classification_report

treinador = svm.SVC()  # algoritmo escolhido

# Realiza validação cruzada com 5 folds (você pode ajustar o número de
folds)
scores = cross_val_score(treinador, X_train, y_train, cv=8)

print("Acurácia por fold na validação cruzada:", scores)
print("Acurácia média na validação cruzada: {:.2f}%".format(scores.mean() *
100))

# Treina o modelo com todo o conjunto de treinamento para a avaliação final
no conjunto de teste
modelo = treinador.fit(X_train, y_train)

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

# melhor avaliar com a matriz de confusão nos dados de treinamento
(opcional)

```

```
# y_pred = modelo.predict(X_train)
# cm = confusion_matrix(y_train, y_pred)
# print(cm)
# print(classification_report(y_train, y_pred))

# com dados de teste que não foram usados no treinamento
print('\nmétricas mais confiáveis (avaliadas no conjunto de teste)')
y_pred2 = modelo.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
# acuracia do modelo na previsão dos dados de teste
acuracia = modelo.score(X_test, y_test)
print("Acurácia nos dados de teste: {:.2f}%".format(acuracia * 100))
print(cm)
print(classification_report(y_test, y_pred2))
```

FIGURA 18 - ACURÁCIA TREINAMENTO COM SVM

```
Acurácia por fold na validação cruzada: [0.83333333 0.58333333 0.58333333 0.66666667 0.66666667 0.83333333
0.5      0.75      ]
Acurácia média na validação cruzada: 67.71%

métricas mais confiáveis (avaliadas no conjunto de teste)
Acurácia nos dados de teste: 58.33%
[[7 1 0]
 [3 4 1]
 [5 0 3]]
      precision    recall  f1-score   support

    espanhol       0.47      0.88      0.61         8
      inglês       0.80      0.50      0.62         8
    português       0.75      0.38      0.50         8

 accuracy                   0.58         24
 macro avg       0.67      0.58      0.57         24
weighted avg       0.67      0.58      0.57         24
```

Tarefa 2 - Melhore uma base de dados ruim

A atividade 02 visa trabalhar com um conjunto de dados pré-construído, onde as opções que o desenvolvedor tem, são de aplicar as técnicas de pré-processamento abaixo relacionadas:

- Seleção
- Limpeza
- Codificação
- Enriquecimento
- Normalização
- Construção de Atributos
- Correção de Prevalência
- Partição do Conjunto de Dados

Busque uma base de dados na UCI Machine Learning que seja indicada para problemas de classificação. (<https://archive.ics.uci.edu/datasets>)

Para esse exemplo, vamos usar a base Secondary Mushroom (<https://archive.ics.uci.edu/static/public/848/data.csv>)

```
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd

from sklearn import svm
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report, accuracy_score

from sklearn.model_selection import train_test_split
import numpy as np

import matplotlib.pyplot as plt

url = "https://archive.ics.uci.edu/static/public/848/data.csv"
colunas = ["class", "cap-diameter", "cap-shape", "cap-surface",
"cap-color", "does-bruise-or-bleed",
"    gill-attachment", "gill-spacing", "gill-color",
"stem-height", "stem-width", "stem-root",
"    stem-surface", "stem-color", "veil-type",
"veil-color", "has-ring", "ring-type",
"    spore-print-color", "habitat", "season"]

colunas_categoricas = [ "cap-shape", "cap-surface", "cap-color",
"does-bruise-or-bleed",
"    gill-attachment", "gill-spacing", "gill-color", "stem-root",
"    stem-surface", "stem-color", "veil-type",
"veil-color", "has-ring", "ring-type",
"    spore-print-color", "habitat", "season"]

colunas_numericas = ["cap-diameter", "stem-height", "stem-width"]

sm = pd.read_csv(url, header=None, low_memory=False, names=colunas)
sm = sm.drop(index=0)
```

```
print(type(sm))
print(sm.head())
```

FIGURA 19 - CARREGAMENTO PARA DATAFRAME

```
<class 'pandas.core.frame.DataFrame'>
  class cap-diameter cap-shape cap-surface cap-color does-bruise-or-bleed \
1      p      15.26      x      g      o      f
2      p      16.6      x      g      o      f
3      p      14.07      x      g      o      f
4      p      14.17      f      h      e      f
5      p      14.64      x      h      o      f

  gill-attachment gill-spacing gill-color stem-height ... stem-root \
1      e      NaN      w      16.95 ... s
2      e      NaN      w      17.99 ... s
3      e      NaN      w      17.8 ... s
4      e      NaN      w      15.77 ... s
5      e      NaN      w      16.53 ... s

  stem-surface stem-color veil-type veil-color has-ring ring-type \
1      y      w      u      w      t      g
2      y      w      u      w      t      g
3      y      w      u      w      t      g
4      y      w      u      w      t      p
5      y      w      u      w      t      p

  spore-print-color habitat season
1      NaN      d      w
2      NaN      d      u
3      NaN      d      w
4      NaN      d      w
5      NaN      d      w

[5 rows x 21 columns]
```

Treinamentos com normalização de colunas, remoção de coluna de identificação e separação de classe dos atributos.

```
sm_dummies = pd.get_dummies(sm[colunas_categoricas])
X = pd.concat([sm[colunas_numericas], sm_dummies], axis=1)

#X = sm.iloc[:,1:]
cols = sm[0:]
print(X.head())
Y = sm['class']
Y_orig = sm['class']
print(Y.unique())
```

FIGURA 20 - PRÉ-PROCESSAMENTO DOS DADOS PARA MODELAGEM

	cap-diameter	stem-height	stem-width	cap-shape_b	cap-shape_c	cap-shape_f	\
1	15.26	16.95	17.09	False	False	False	
2	16.6	17.99	18.19	False	False	False	
3	14.07	17.8	17.74	False	False	False	
4	14.17	15.77	15.98	False	False	True	
5	14.64	16.53	17.2	False	False	False	

	cap-shape_o	cap-shape_p	cap-shape_s	cap-shape_x	...	habitat_h	\
1	False	False	False	True	...	False	
2	False	False	False	True	...	False	
3	False	False	False	True	...	False	
4	False	False	False	False	...	False	
5	False	False	False	True	...	False	

	habitat_l	habitat_m	habitat_p	habitat_u	habitat_w	season_a	season_s	\
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
5	False	False	False	False	False	False	False	

	season_u	season_w
1	False	True
2	True	False
3	False	True
4	False	True
5	False	True

[5 rows x 119 columns]
['p' 'e']

Tratamento de dados, visando a melhoria no desempenho do classificador (SVM).

```
X_orig = X.copy()
print(X_orig.head())

print(Y_orig.unique() )

# normalização min-max
X = pd.DataFrame( minmax_scale(X), columns=X.columns)

#retirada de dados faltantes

X.dropna(axis = 1, how ='any')

print(X_orig.head())
print(X.head())
```


FIGURA 23 - CÓPIA DOS DADOS ORIGINAIS E NORMALIZAÇÃO MIN-MAX DAS VARIÁVEIS DE ENTRADA (PARTE 3)

	cap-diameter	stem-height	stem-width	cap-shape_b	cap-shape_c	\
0	0.240155	0.499705	0.164469	0.0	0.0	
1	0.261782	0.530366	0.175055	0.0	0.0	
2	0.220949	0.524764	0.170725	0.0	0.0	
3	0.222563	0.464917	0.153787	0.0	0.0	
4	0.230148	0.487323	0.165528	0.0	0.0	

	cap-shape_f	cap-shape_o	cap-shape_p	cap-shape_s	cap-shape_x	...	\
0	0.0	0.0	0.0	0.0	1.0	...	
1	0.0	0.0	0.0	0.0	1.0	...	
2	0.0	0.0	0.0	0.0	1.0	...	
3	1.0	0.0	0.0	0.0	0.0	...	
4	0.0	0.0	0.0	0.0	1.0	...	

	habitat_h	habitat_l	habitat_m	habitat_p	habitat_u	habitat_w	season_a	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

	season_s	season_u	season_w
0	0.0	0.0	1.0
1	0.0	1.0	0.0
2	0.0	0.0	1.0
3	0.0	0.0	1.0
4	0.0	0.0	1.0

[5 rows x 119 columns]

Construção do modelo, dos testes e das métricas da matriz de confusão.

```
# com os dados originais
X_oring_train, X_orig_test, y_orig_train, y_orig_test =
train_test_split(X_orig,
                  Y_orig, test_size=0.25,
                  stratify=Y_orig, random_state=10)

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

treinador = svm.SVC() #algoritmo escolhido

modelo_orig = treinador.fit(X_oring_train, y_orig_train)

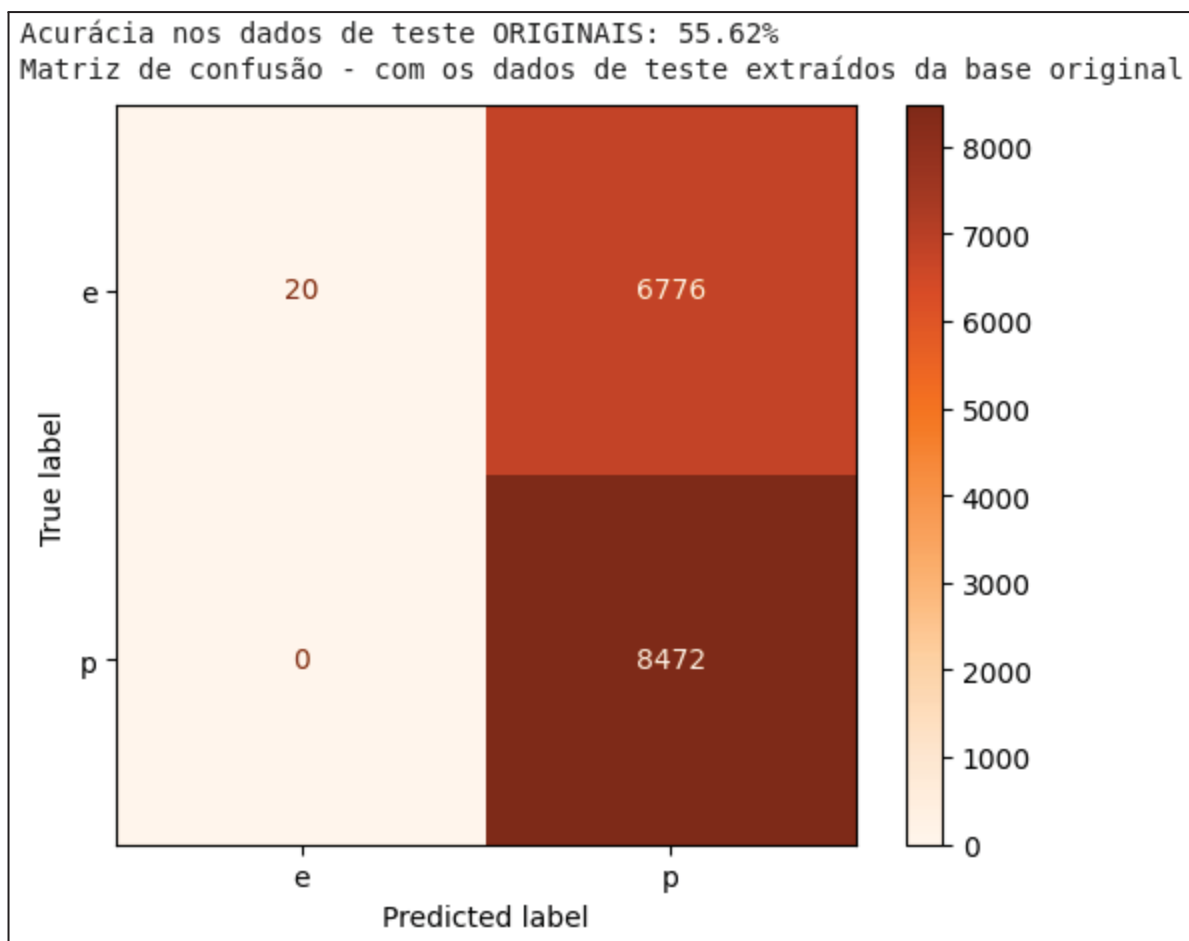
modelo = treinador.fit(X_train, y_train)

# predição com os dados de teste da base original
#print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_orig_pred = modelo_orig.predict(X_orig_test)
acuracia_orig_test = accuracy_score(y_orig_test, y2_orig_pred)
print("Acurácia nos dados de teste ORIGINAIS:
{:.2f}%".format(acuracia_orig_test * 100))

# gerando a matriz de confusão
print('Matriz de confusão - com os dados de teste extraídos da base
original')
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_orig_test,
                              display_labels= modelo_orig.classes_)
disp.plot(cmap="Oranges")

plt.show()
```

FIGURA 24 - ANÁLISE DOS RESULTADOS

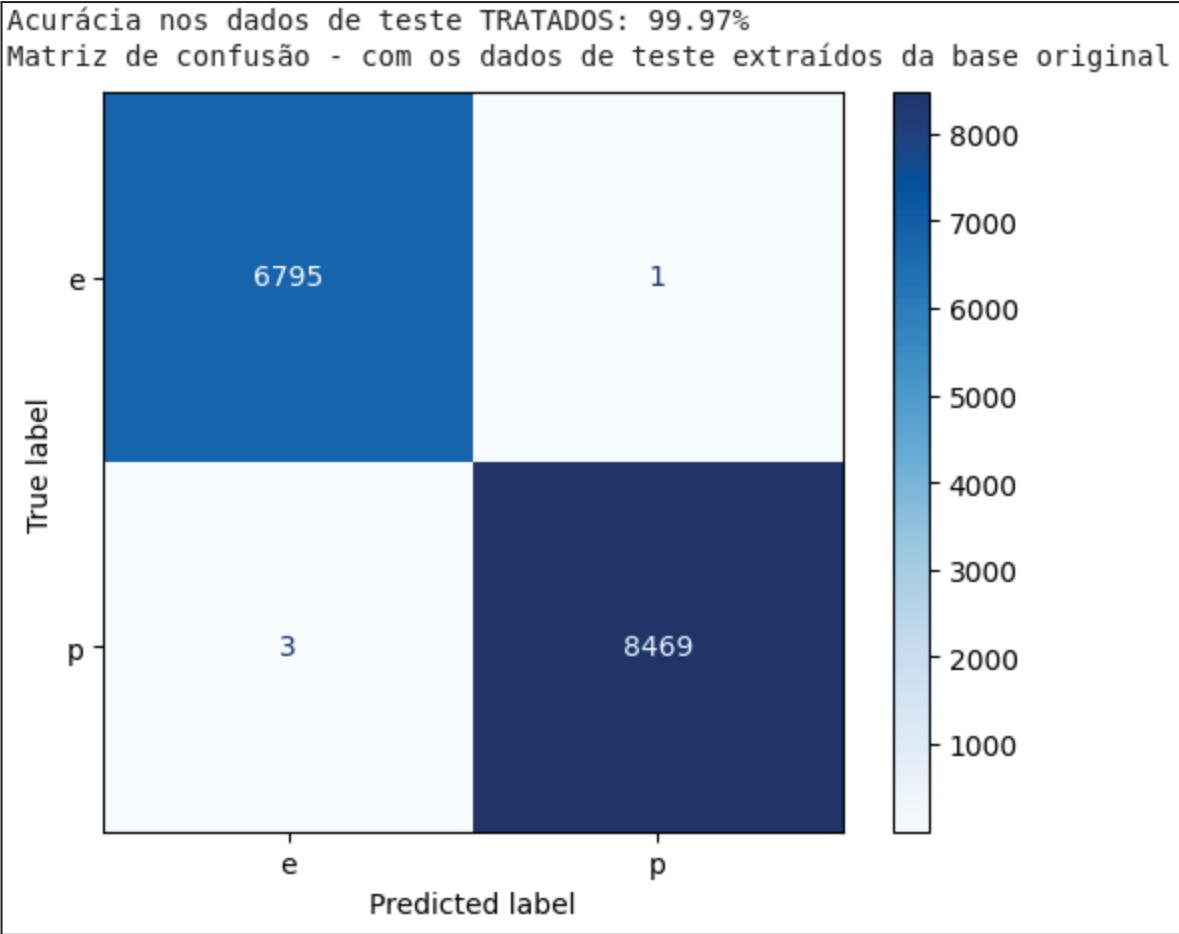


Dados após os processos de tratamento de dados.

```
y_pred = modelo.predict(X_test)
acuracia_test = accuracy_score(y_test, y_pred)
print("Acurácia nos dados de teste TRATADOS: {:.2f}%".format(acuracia_test
* 100))

# gerando a matriz de confusão
print('Matriz de confusão - com os dados de teste extraídos da base
original')
cm_test = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=cm_test, display_labels=
modelo.classes_)
disp.plot(cmap='Blues')
plt.show()
```

FIGURA 25 - ANÁLISE DOS RESULTADOS APÓS TRATAMENTO DE DADOS



APÊNDICE G - APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

Os comandos descritos são executados no RStudio.

1 - Classificação

a) Veículos

```
#carregamento da base
dados <- read.csv("databases/6 - Veiculos - Dados.csv")
head(dados)
dados$a <- NULL
#separando em bases de treino e teste
set.seed(2034)
indice <- createDataPartition(dados$tipo, p = 0.8, list =
FALSE)
treino <- dados[indice, ]
teste <- dados[-indice, ]
#treinando o modelo usando svm com crossvalidarion e mostrando
as metricas
set.seed(2034)
ctrl <- trainControl(method = "cv", number = 10)
rf <- train(tipo ~ ., data = treino, method = "rf", trControl =
ctrl)

rf
#fazendo a predicaõ na base de teste
predito_rf <- predict(rf, teste)
```

```
#apresentando a matriz de confusão
confusionMatrix(predito_rf, as.factor(teste$tipo))
```

TABELA 7 - EXPERIMENTO DE CLASSIFICAÇÃO - VEÍCULOS - ORDENAMENTO POR ACURÁCIA
(PARTE 1)

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	C=50 Sigma=0,021	0,8802	Reference Prediction bus opel saab van bus 43 0 0 0 opel 0 35 12 0 saab 0 6 30 0 van 0 1 1 39
SVM – Hold-out	C = 100 Sigma = 0,021	0,8743	Reference Prediction bus opel saab van bus 43 0 0 0 opel 0 34 11 1 saab 0 8 31 0 van 0 0 1 38
RF – CV	mtry=18	0,7784	Reference Prediction bus opel saab van bus 42 0 3 0 opel 0 27 15 0 saab 1 13 23 1 van 0 2 2 38
RNA – CV	size=05 decay=0,1	0,7725	Reference Prediction bus opel saab van bus 39 1 3 0 opel 1 32 20 1 saab 2 9 20 0 van 1 0 0 38

TABELA 8 - EXPERIMENTO DE CLASSIFICAÇÃO - VEÍCULOS - ORDENAMENTO POR ACURÁCIA
(PARTE 2)

RF – Hold-out	mtry=02	0,7605	Reference Prediction bus opel saab van bus 43 0 3 0 opel 0 22 14 0 saab 0 17 24 1 van 0 3 2 38
RNA – Hold-out	size=05 decay=0,1	0,7006	Reference Prediction bus opel saab van bus 42 2 1 0 opel 1 36 40 0 saab 0 2 1 1 van 0 2 1 38
KNN	k=01	0,6467	Reference Prediction bus opel saab van bus 37 4 8 3 opel 2 18 15 0 saab 3 18 18 1 van 1 2 2 35

TABELA 9 - EXPERIMENTO DE CLASSIFICAÇÃO - VEÍCULOS - NOVOS CASOS

a	Co mp	Circ	DCir c	Rad Ra	PrA xisR a	Max LRa	Scat Ra	Elong	PrA xisR ect	Max LRe ct	ScV arM axis	ScV arm axis	RaG yr	Ske wM axis	Ske wm axis	Kurt max is	Kurt Max is	Holl Ra	pred ict_s vm
1	76	45	69	306	112	42	174	59	24	182	222	301	200	96	6	1	204	198	van
2	100	54	106	288	118	17	181	48	28	167	209	971	210	65	2	39	178	198	opel
3	103	51	75	283	95	10	182	50	24	140	132	519	146	83	11	7	179	205	bus

b) Diabetes

```
##Leitura da database
setwd("D:/GIT/IAA008-AprendizadoMaquina/")
dados <- read.csv("databases/10 - Diabetes - Dados.csv", header
= T)
dados$num <- NULL
#View(dados)

#(Ano atual com 4 dígitos + 2 algarismos do dígito verificador
do CPF de um dos integrantes)
set.seed(2034)

ind <-createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <-dados[ind,]
teste <-dados[-ind,]

##executa o RNA com esse grid
rna <- train(diabetes~.,data = treino, method =
"nnet",trace=FALSE)
rna
##Aplica o modelo no arquivo de teste
predict.rna <- predict(rna,teste)
##mostra as métricas
confusionMatrix(predict.rna, as.factor(teste$diabetes))
```

TABELA 10 - EXPERIMENTO DE CLASSIFICAÇÃO - DIABETES - ORDENAMENTO POR ACURÁCIA

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – Hold-out	size=03 decay=0,1	0,817	<pre> Reference Prediction neg pos neg 88 16 pos 12 37 </pre>
SVM – Hold-out	C=0,25 Sigma=0,1090209	0,7908	<pre> Reference Prediction neg pos neg 93 25 pos 7 28 </pre>
SVM – CV	C=0,25 Sigma=0,1090209	0,7908	<pre> Reference Prediction neg pos neg 93 25 pos 7 28 </pre>
RNA – CV	size=05 decay=0,1	0,7843	<pre> Reference Prediction neg pos neg 91 24 pos 9 29 </pre>
RF – CV	mtry=02	0,7843	<pre> Reference Prediction neg pos neg 85 18 pos 15 35 </pre>
RF – Hold-out	mtry=02	0,7712	<pre> Reference Prediction neg pos neg 85 20 pos 15 33 </pre>
KNN	k=10	0,7451	<pre> Reference Prediction neg pos neg 88 27 pos 12 26 </pre>

TABELA 11 - EXPERIMENTO DE CLASSIFICAÇÃO - DIABETES - NOVOS CASOS

num	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes	predict.rna
1	16	148	16	86	756	202	2025	24	?	pos
2	15	197	81	25	80	665	1351	30	?	pos
3	16	128	4	50	134	321	1738	36	?	pos

2 - Regressão

c) Admissão

```

dados <- read.csv("databases/9 - Admissao - Dados.csv", header
= T)
dados$num <- NULL
#View(dados)

#(Ano atual com 4 dígitos + 2 algarismos do dígito verificador

```

```

do CPF de um dos integrantes)
set.seed(2034)

ind <-createDataPartition(dados$ChanceOfAdmit, p=0.80,
list=FALSE)
treino <-dados[ind,]
teste <-dados[-ind,]

##Cross-Validation
control <- trainControl(method='cv', number = 10)

##executa o RF com esse grid
rf <- train(ChanceOfAdmit~.,data = treino, method = "rf",
trControl = control)
rf
##Aplica o modelo no arquivo de teste
predict.rf <- predict(rf,teste)
##mostra as métricas
rmse(teste$ChanceOfAdmit, predict.rf)
mae(teste$ChanceOfAdmit, predict.rf)
cor(teste$ChanceOfAdmit, predict.rf, method = "pearson")
Syx <- function(predito, observado, p) {
return(sqrt(sum((observado - predito)^2) /
(length(observado) - p)))
}
Syx(predict.rf,teste$ChanceOfAdmit,ncol(teste) - 1)

r2 <- function(predito, observado){ return (1 - (sum((predito-
observado)^2)/sum((observado-mean(observado))^2)))}

r2(predict.rf, teste$ChanceOfAdmit)

```

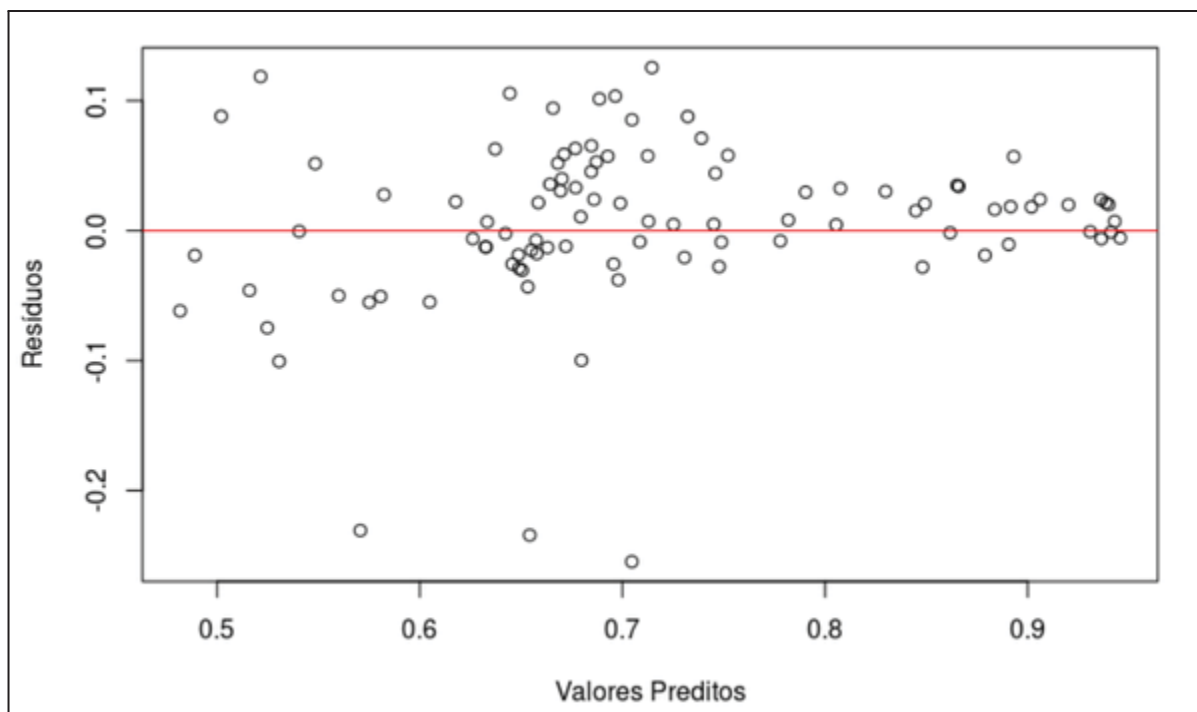
TABELA 12 - EXPERIMENTO DE REGRESSÃO - ADMISSÃO - ORDENAMENTO POR R2

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – CV	mtry=02	0,8138969	0,06462985	0,9064265	0,06227888	0,04290906
RF – Hold-out	mtry=02	0,8133655	0,06472206	0,9057414	0,06236773	0,04237825
SVM – CV	C=01 Sigma=0,1599276	0,8107899	0,06516711	0,9046141	0,0627966	0,04506664
SVM – Hold-out	C=0,5 Sigma=0,1599276	0,8034357	0,0664215	0,9032343	0,06400535	0,0452094
RNA – Hold-out	size=05 decay=0,1	0,76	0,073	0,89	0,07	0,054
RNA – CV	size=05 decay=0,1	0,76	0,073	0,89	0,07	0,054
KNN	K=10	0,7337591	0,07730253	0,8578415	0,07449058	0,05307143

TABELA 13 - EXPERIMENTO DE REGRESSÃO - ADMISSÃO - NOVOS CASOS

num	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	ChanceOfAdmit	predict.rf
1	338	96	1	43	3	865	0	?	0,7
2	299	108	4	36	37	704	0	?	0,58
3	324	109	2	4	42	6	0	?	0,66

FIGURA 26 - EXPERIMENTO DE REGRESSÃO - ADMISSÃO - GRÁFICOS DE RESÍDUOS - MODELO RF CROSSVALIDATION



d) Biomassa

```

#carregamento da base
dados <- read.csv("databases/5 - Biomassa - Dados.csv")
head(dados)
#separando em bases de treino e teste
set.seed(2034)
indice <- createDataPartition(dados$biomassa, p = 0.8, list =
FALSE)
treino <- dados[indice, ]
teste <- dados[-indice, ]
#funcoes das metricas
syx <- function(observado, predito, p) {
return(sqrt(sum((observado - predito)^2) / (length(observado)
- p)))
}
r2 <- function(observado, predito){
return(1 - (sum((predito - observado)^2) / sum((observado -
mean(observado))^2)))
}
metricas <- function(observado, predito, p) {
#calculando as metricas
r2_valor <- r2(observado, predito)
syx_valor <- syx(observado, predito, p)
pearson_valor <- cor(observado, predito, method = "pearson")
rmse_valor <- rmse(observado, predito)
mae_valor <- mae(observado, predito)
#apresentando as metricas
cat("R2:", r2_valor, "\n")
cat("Syx:", syx_valor, "\n")
cat("Pearson:", pearson_valor, "\n")
cat("RMSE:", rmse_valor, "\n")
cat("MAE:", mae_valor, "\n")
}
#treinando o modelo usando svm com hold-out e mostrando as
metricas
set.seed(2034)
tune_grid <- expand.grid(C = c(1, 2, 10, 50, 100), sigma =
c(.021, .015, 0.2))

```

```

svml <- train(biomassa ~ ., data = treino, method =
"svmRadial", tuneGrid = tune_grid)
svml
#fazendo a predicaõ na base de teste
predito_svml <- predict(svml, teste)
#apresentando as metricas
metricas(teste$biomassa, predito_svml, ncol(teste) - 1)

```

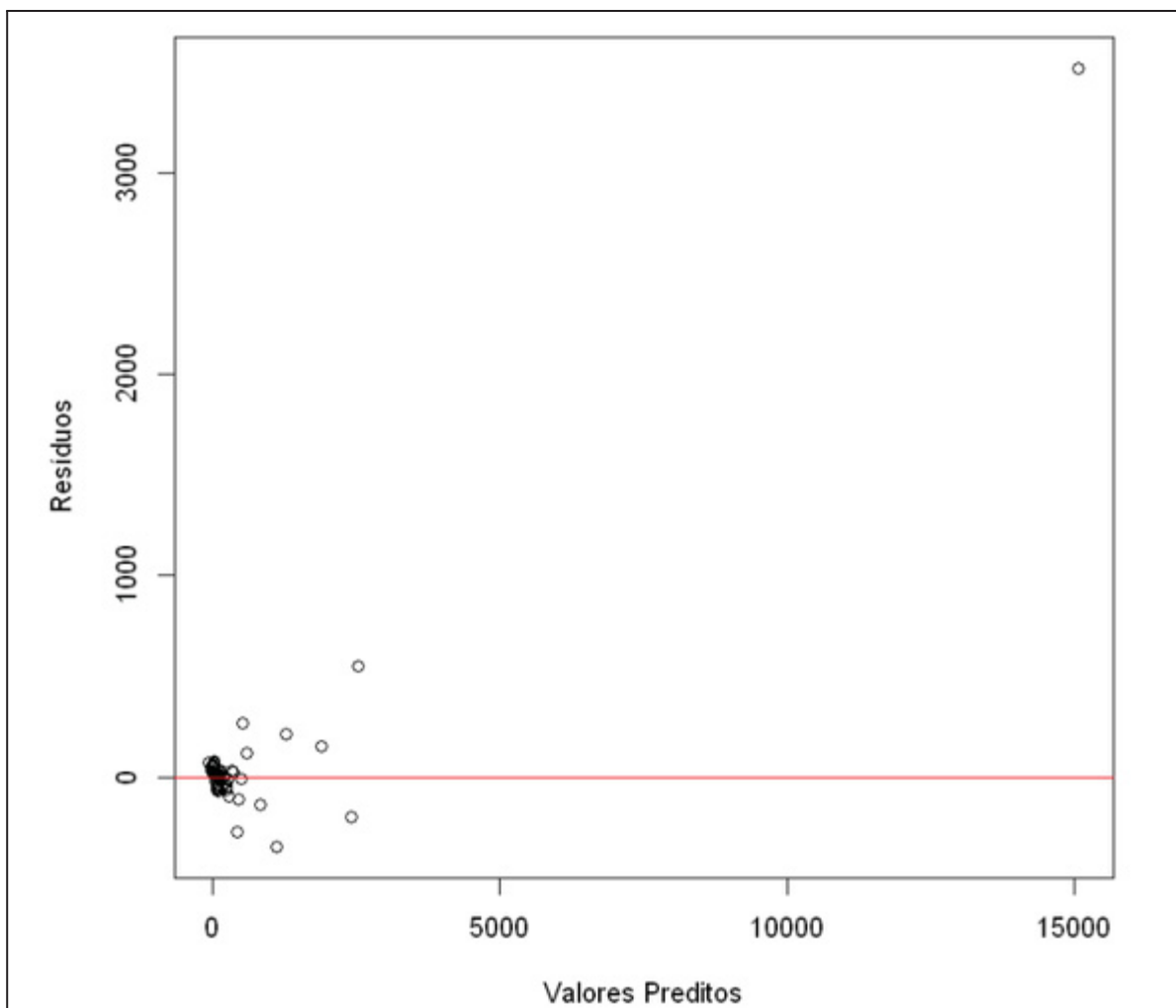
TABELA 14 - EXPERIMENTO DE REGRESSÃO - BIOMASSA - ORDENAMENTO POR R2

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – Hold-out	C=100 Sigma=0,021	0,9622214	481,2785	0,9981336	469,0922	127,2661
SVM – CV	C=100 Sigma=0,021	0,9622214	481,2785	0,9981336	469,0922	127,2661
RNA – Hold-out	size=05 decay=0,1	0,71	1324	0,97	1290	227
RNA – CV	size=05 decay=0,1	0,71	1324	0,97	1290	227
RF – Hold-out	mtry=02	0,6397223	1486,252	0,9529809	1448,619	241,0877
RF – CV	mtry=03	0,6501568	1464,571	0,9574702	1427,487	238,9709
KNN	K=03	0,6579974	1448,066	0,959401	1411,4	233,0013

TABELA 15 - EXPERIMENTO DE REGRESSÃO - BIOMASSA - NOVOS CASOS

Novos casos				
dap	h	Me	biomassa	Predict_svm
12.1	20.5	0.467	?	75.95294
7.2	12.3	0.514	?	22.16095
19.1	14.9	0.617	?	170.43377

FIGURA 27 - EXPERIMENTO DE REGRESSÃO - BIOMASSA - GRÁFICOS DE RESÍDUOS -
MODELO SVM HOLDOUT



3 - Agrupamento

```
### Pacotes necessários
library("caret")
library(Metrics)
##Leitura da database
setwd("D:/GIT/IAA008-AprendizadoMaquina/")
dados <- read.csv("databases/6 - Veiculos - Dados.csv", header
= T)
dados$a <- NULL
View(dados)
#(Ano atual com 4 dígitos + 2 algarismos do dígito verificador
do CPF de um dos integrantes)
```

```

set.seed(2034)
km.res = kmeans(dados[, 0:(ncol(dados) - 1)],10)
print(km.res)
table(km.res$cluster, dados$tipo)
resultado <- cbind(dados,km.res$cluster)
resultado

```

FIGURA 28 - AGRUPAMENTO - VEÍCULO - LISTA DE CLUSTERS GERADOS

K-means clustering with 10 clusters of sizes 56, 24, 78, 112, 71, 135, 47, 77, 184, 62

Cluster means:

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis
1	102	49	99	206	64	9.2	199	33	23	156	217	596
198		69		6.1	13.6		192	199				
2	107	56	103	190	56	5.8	249	27	27	168	271	911
247		84		6.5	14.2		183	184				
3	104	53	103	197	61	10.2	212	31	24	165	224	664
212		71		7.2	16.4		188	197				
4	90	46	79	154	62	9.1	156	43	20	152	175	359
180		73		6.6	9.1		187	195				
5	94	43	83	187	64	7.5	171	38	21	142	193	443
164		68		5.9	14.7		195	201				
6	88	38	62	130	57	6.4	128	53	18	132	149	243
139		73		6.6	11.5		188	193				
7	98	47	93	199	64	8.6	186	35	22	151	207	523
182		68		7.0	12.7		194	200				
8	104	54	103	207	63	10.6	221	30	25	171	229	717
212		72		7.9	16.0		188	198				
9	87	42	71	144	60	7.4	146	46	19	141	168	318
163		75		5.9	10.4		187	192				
10	92	39	77	187	71	12.8	155	42	19	135	187	363
143		73		4.3	14.9		194	201				

TABELA 16 - AGRUPAMENTO - VEÍCULO - 10 PRIMEIRAS LINHAS DO ARQUIVO COM O CLUSTER CORRESPONDENTE.

Cluster	bus	opel	saab	van
1	10	24	22	0
2	24	0	0	0
3	5	33	40	0
4	28	11	9	64
5	22	22	27	0
6	0	27	26	82
7	10	18	19	0
8	6	41	30	0
9	91	25	25	43
10	22	11	19	10

3 - Regras de associação

```
### Pacotes necessários
library(arules)
library(datasets)
##Leitura da database
setwd("D:/GIT/IAA008-AprendizadoMaquina/")
dados <- read.csv2("databases/2 - Musculacao - Dados.csv",
header = F)
View(dados)
summary(dados)
set.seed(2034)
rules <- apriori(dados, parameter = list(supp = 0.001, conf =
0.7, minlen = 2))
summary(rules)

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

TABELA 17 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 1).

lhs	_1	rhs	support	confidence
{V2= Gemeos}	=>	{V3=Bicicleta}	0.02564	1
{V2= Gemeos}	=>	{V4=Esteira}	0.02564	1
{V2= Gemeos}	=>	{V1=Extensor}	0.02564	1
{V3=Gemeos}	=>	{V2=AgachamentoSmith}	0.05128	1
{V3=Gemeos}	=>	{V4=Esteira}	0.05128	1
{V4=Crucifixo}	=>	{V3=Afundo}	0.05128	1
{V4=Crucifixo}	=>	{V2=Gemeos}	0.05128	1
{V4=Crucifixo}	=>	{V1=LegPress}	0.05128	1
{V2=Flexor}	=>	{V3=Bicicleta}	0.05128	1
{V2=Flexor}	=>	{V4=Esteira}	0.05128	1
{V2=Flexor}	=>	{V1=Extensor}	0.05128	1
{V4=Gemeos}	=>	{V2=AgachamentoSmith}	0.05128	1
{V4=Gemeos}	=>	{V3=Bicicleta}	0.05128	1
{V4=Gemeos}	=>	{V1=Extensor}	0.05128	1
{V4=Bicicleta}	=>	{V3=Afundo}	0.05128	1
{V4=Agachamento}	=>	{V2=LegPress}	0.05128	1
{V4=Agachamento}	=>	{V1=Gemeos}	0.05128	1
{V4=Agachamento}	=>	{V3=Afundo}	0.05128	1
{V4=Adutor}	=>	{V3=LegPress}	0.07692	1
{V3=LegPress}	=>	{V4=Adutor}	0.07692	1
{V4=Adutor}	=>	{V2=Agachamento}	0.07692	1
{V2=Agachamento}	=>	{V4=Adutor}	0.07692	1
{V4=Adutor}	=>	{V1=Adutor}	0.07692	1
{V1=Adutor}	=>	{V4=Adutor}	0.07692	1
{V3=LegPress}	=>	{V2=Agachamento}	0.07692	1
{V2=Agachamento}	=>	{V3=LegPress}	0.07692	1
{V3=LegPress}	=>	{V1=Adutor}	0.07692	1
{V1=Adutor}	=>	{V3=LegPress}	0.07692	1
{V2=Agachamento}	=>	{V1=Adutor}	0.07692	1
{V1=Adutor}	=>	{V2=Agachamento}	0.07692	1
{V2=LegPress}	=>	{V1=Gemeos}	0.07692	1

TABELA 18 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 2).

{V2=LegPress}	=>	{V3=Afundo}	0.07692	1
{V3=Agachamento}	=>	{V2=Gemeos}	0.07692	1
{V3=Agachamento}	=>	{V1=LegPress}	0.07692	1
{V3=Agachamento}	=>	{V4=}	0.07692	1
{V3=Bicicleta}	=>	{V1=Extensor}	0.30769	1
{V2=}	=>	{V3=}	0.33333	1
{V3=}	=>	{V2=}	0.33333	1
{V2=}	=>	{V4=}	0.33333	1
{V3=}	=>	{V4=}	0.33333	1
{V2= Gemeos, V3=Bicicleta}	=>	{V4=Esteira}	0.02564	1
{V2= Gemeos, V4=Esteira}	=>	{V3=Bicicleta}	0.02564	1
{V2= Gemeos, V3=Bicicleta}	=>	{V1=Extensor}	0.02564	1
{V1=Extensor, V2= Gemeos}	=>	{V3=Bicicleta}	0.02564	1
{V2= Gemeos, V4=Esteira}	=>	{V1=Extensor}	0.02564	1
{V1=Extensor, V2= Gemeos}	=>	{V4=Esteira}	0.02564	1
{V1=Afundo, V3=Gemeos}	=>	{V2=AgachamentoSmith}	0.02564	1
{V1=Afundo, V2=AgachamentoSmith}	=>	{V3=Gemeos}	0.02564	1
{V1=Afundo, V3=Gemeos}	=>	{V4=Esteira}	0.02564	1
{V1=Afundo, V4=Esteira}	=>	{V3=Gemeos}	0.02564	1
{V2=AgachamentoSmith, V3=Gemeos}	=>	{V4=Esteira}	0.05128	1
{V3=Gemeos, V4=Esteira}	=>	{V2=AgachamentoSmith}	0.05128	1
{V1=Extensor, V3=Gemeos}	=>	{V2=AgachamentoSmith}	0.02564	1
{V1=Extensor, V3=Gemeos}	=>	{V4=Esteira}	0.02564	1
{V3=Afundo, V4=Crucifixo}	=>	{V2=Gemeos}	0.05128	1
{V2=Gemeos, V4=Crucifixo}	=>	{V3=Afundo}	0.05128	1
{V3=Afundo, V4=Crucifixo}	=>	{V1=LegPress}	0.05128	1
{V1=LegPress, V4=Crucifixo}	=>	{V3=Afundo}	0.05128	1
{V2=Gemeos, V4=Crucifixo}	=>	{V1=LegPress}	0.05128	1
{V1=LegPress, V4=Crucifixo}	=>	{V2=Gemeos}	0.05128	1
{V2=Flexor, V3=Bicicleta}	=>	{V4=Esteira}	0.05128	1
{V2=Flexor, V4=Esteira}	=>	{V3=Bicicleta}	0.05128	1
{V2=Flexor, V3=Bicicleta}	=>	{V1=Extensor}	0.05128	1
{V1=Extensor, V2=Flexor}	=>	{V3=Bicicleta}	0.05128	1
{V2=Flexor, V4=Esteira}	=>	{V1=Extensor}	0.05128	1
{V1=Extensor, V2=Flexor}	=>	{V4=Esteira}	0.05128	1

TABELA 19 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 3).

{V2=AgachamentoSmith, V4=Gemeos}	=>	{V3=Bicicleta}	0.05128	1
{V3=Bicicleta, V4=Gemeos}	=>	{V2=AgachamentoSmith}	0.05128	1
{V2=AgachamentoSmith, V4=Gemeos}	=>	{V1=Extensor}	0.05128	1
{V1=Extensor, V4=Gemeos}	=>	{V2=AgachamentoSmith}	0.05128	1
{V3=Bicicleta, V4=Gemeos}	=>	{V1=Extensor}	0.05128	1
{V1=Extensor, V4=Gemeos}	=>	{V3=Bicicleta}	0.05128	1
{V2=LegPress, V4=Bicicleta}	=>	{V1=Gemeos}	0.02564	1
{V1=Gemeos, V4=Bicicleta}	=>	{V2=LegPress}	0.02564	1
{V2=LegPress, V4=Bicicleta}	=>	{V3=Afundo}	0.02564	1
{V1=Gemeos, V4=Bicicleta}	=>	{V3=Afundo}	0.02564	1
{V2=Gemeos, V4=Bicicleta}	=>	{V3=Afundo}	0.02564	1
{V1=LegPress, V4=Bicicleta}	=>	{V3=Afundo}	0.02564	1
{V2=Gemeos, V4=Bicicleta}	=>	{V1=LegPress}	0.02564	1
{V1=LegPress, V4=Bicicleta}	=>	{V2=Gemeos}	0.02564	1
{V2=LegPress, V4=Agachamento}	=>	{V1=Gemeos}	0.05128	1
{V1=Gemeos, V4=Agachamento}	=>	{V2=LegPress}	0.05128	1
{V2=LegPress, V4=Agachamento}	=>	{V3=Afundo}	0.05128	1
{V3=Afundo, V4=Agachamento}	=>	{V2=LegPress}	0.05128	1
{V1=Gemeos, V4=Agachamento}	=>	{V3=Afundo}	0.05128	1
{V3=Afundo, V4=Agachamento}	=>	{V1=Gemeos}	0.05128	1
{V1=Afundo, V2=AgachamentoSmith}	=>	{V4=Esteira}	0.02564	1
{V1=Afundo, V4=Esteira}	=>	{V2=AgachamentoSmith}	0.02564	1
{V1=Afundo, V2=}	=>	{V3=}	0.05128	1
{V1=Afundo, V3=}	=>	{V2=}	0.05128	1
{V1=Afundo, V2=}	=>	{V4=}	0.05128	1
{V1=Afundo, V4=}	=>	{V2=}	0.05128	1
{V1=Afundo, V3=}	=>	{V4=}	0.05128	1
{V1=Afundo, V4=}	=>	{V3=}	0.05128	1
{V3=LegPress, V4=Adutor}	=>	{V2=Agachamento}	0.07692	1
{V2=Agachamento, V4=Adutor}	=>	{V3=LegPress}	0.07692	1
{V2=Agachamento, V3=LegPress}	=>	{V4=Adutor}	0.07692	1
{V3=LegPress, V4=Adutor}	=>	{V1=Adutor}	0.07692	1

TABELA 20 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 4).

{V1=Adutor, V4=Adutor}	=>	{V3=LegPress}	0.07692	1
{V1=Adutor, V3=LegPress}	=>	{V4=Adutor}	0.07692	1
{V2=Agachamento, V4=Adutor}	=>	{V1=Adutor}	0.07692	1
{V1=Adutor, V4=Adutor}	=>	{V2=Agachamento}	0.07692	1
{V1=Adutor, V2=Agachamento}	=>	{V4=Adutor}	0.07692	1
{V2=Agachamento, V3=LegPress}	=>	{V1=Adutor}	0.07692	1
{V1=Adutor, V3=LegPress}	=>	{V2=Agachamento}	0.07692	1
{V1=Adutor, V2=Agachamento}	=>	{V3=LegPress}	0.07692	1
{V1=Gemeos, V2=LegPress}	=>	{V3=Afundo}	0.07692	1
{V2=LegPress, V3=Afundo}	=>	{V1=Gemeos}	0.07692	1
{V1=Gemeos, V3=Afundo}	=>	{V2=LegPress}	0.07692	1
{V2=Gemeos, V3=Agachamento}	=>	{V1=LegPress}	0.07692	1
{V1=LegPress, V3=Agachamento}	=>	{V2=Gemeos}	0.07692	1
{V2=Gemeos, V3=Agachamento}	=>	{V4=}	0.07692	1
{V3=Agachamento, V4=}	=>	{V2=Gemeos}	0.07692	1
{V2=Gemeos, V4=}	=>	{V3=Agachamento}	0.07692	1
{V1=LegPress, V3=Agachamento}	=>	{V4=}	0.07692	1
{V3=Agachamento, V4=}	=>	{V1=LegPress}	0.07692	1
{V1=Gemeos, V2=}	=>	{V3=}	0.05128	1
{V1=Gemeos, V3=}	=>	{V2=}	0.05128	1
{V1=Gemeos, V2=}	=>	{V4=}	0.05128	1
{V1=Gemeos, V4=}	=>	{V2=}	0.05128	1
{V1=Gemeos, V3=}	=>	{V4=}	0.05128	1
{V1=Gemeos, V4=}	=>	{V3=}	0.05128	1
{V2=Gemeos, V3=Afundo}	=>	{V1=LegPress}	0.07692	1
{V1=LegPress, V3=Afundo}	=>	{V2=Gemeos}	0.07692	1
{V2=Gemeos, V3=Bicicleta}	=>	{V4=Esteira}	0.02564	1
{V2=Gemeos, V4=Esteira}	=>	{V3=Bicicleta}	0.02564	1
{V2=Gemeos, V3=Bicicleta}	=>	{V1=Extensor}	0.02564	1
{V1=Extensor, V2=Gemeos}	=>	{V3=Bicicleta}	0.02564	1
{V2=Gemeos, V4=Esteira}	=>	{V1=Extensor}	0.02564	1
{V1=Extensor, V2=Gemeos}	=>	{V4=Esteira}	0.02564	1
{V2=Gemeos, V4=}	=>	{V1=LegPress}	0.07692	1
{V2=AgachamentoSmith, V3=Bicicleta}	=>	{V1=Extensor}	0.20512	1

TABELA 21 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 5).

{V3=Bicicleta, V4=Esteira}	=>	{V1=Extensor}	0.25641	1
{V1=LegPress, V2=}	=>	{V3=}	0.23076	1
{V1=LegPress, V3=}	=>	{V2=}	0.23076	1
{V2=, V3=}	=>	{V4=}	0.33333	1
{V2=, V4=}	=>	{V3=}	0.33333	1
{V3=, V4=}	=>	{V2=}	0.33333	1
{V1=LegPress, V2=}	=>	{V4=}	0.23076	1
{V1=LegPress, V3=}	=>	{V4=}	0.23076	1
{V2= Gemeos, V3=Bicicleta, V4=Esteira}	=>	{V1=Extensor}	0.02564	1
{V1=Extensor, V2= Gemeos, V3=Bicicleta}	=>	{V4=Esteira}	0.02564	1
{V1=Extensor, V2= Gemeos, V4=Esteira}	=>	{V3=Bicicleta}	0.02564	1
{V1=Afundo, V2=AgachamentoSmith, V3=Gemeos}	=>	{V4=Esteira}	0.02564	1
{V1=Afundo, V3=Gemeos, V4=Esteira}	=>	{V2=AgachamentoSmith}	0.02564	1
{V1=Afundo, V2=AgachamentoSmith, V4=Esteira}	=>	{V3=Gemeos}	0.02564	1
{V1=Extensor, V2=AgachamentoSmith, V3=Gemeos}	=>	{V4=Esteira}	0.02564	1
{V1=Extensor, V3=Gemeos, V4=Esteira}	=>	{V2=AgachamentoSmith}	0.02564	1
{V2=Gemeos, V3=Afundo, V4=Crucifixo}	=>	{V1=LegPress}	0.05128	1
{V1=LegPress, V3=Afundo, V4=Crucifixo}	=>	{V2=Gemeos}	0.05128	1
{V1=LegPress, V2=Gemeos, V4=Crucifixo}	=>	{V3=Afundo}	0.05128	1
{V2=Flexor, V3=Bicicleta, V4=Esteira}	=>	{V1=Extensor}	0.05128	1
{V1=Extensor, V2=Flexor, V3=Bicicleta}	=>	{V4=Esteira}	0.05128	1
{V1=Extensor, V2=Flexor, V4=Esteira}	=>	{V3=Bicicleta}	0.05128	1

TABELA 22 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 6).

{V2=AgachamentoSmith, V3=Bicicleta, V4=Gemeos}	=>	{V1=Extensor}	0.05128	1
{V1=Extensor, V2=AgachamentoSmith, V4=Gemeos}	=>	{V3=Bicicleta}	0.05128	1
{V1=Extensor, V3=Bicicleta, V4=Gemeos}	=>	{V2=AgachamentoSmith}	0.05128	1
{V1=Gemeos, V2=LegPress, V4=Bicicleta}	=>	{V3=Afundo}	0.02564	1
{V2=LegPress, V3=Afundo, V4=Bicicleta}	=>	{V1=Gemeos}	0.02564	1
{V1=Gemeos, V3=Afundo, V4=Bicicleta}	=>	{V2=LegPress}	0.02564	1
{V2=Gemeos, V3=Afundo, V4=Bicicleta}	=>	{V1=LegPress}	0.02564	1
{V1=LegPress, V3=Afundo, V4=Bicicleta}	=>	{V2=Gemeos}	0.02564	1
{V1=LegPress, V2=Gemeos, V4=Bicicleta}	=>	{V3=Afundo}	0.02564	1
{V1=Gemeos, V2=LegPress, V4=Agachamento}	=>	{V3=Afundo}	0.05128	1
{V2=LegPress, V3=Afundo, V4=Agachamento}	=>	{V1=Gemeos}	0.05128	1
{V1=Gemeos, V3=Afundo, V4=Agachamento}	=>	{V2=LegPress}	0.05128	1
{V1=Afundo, V2=, V3=}	=>	{V4=}	0.05128	1
{V1=Afundo, V2=, V4=}	=>	{V3=}	0.05128	1
{V1=Afundo, V3=, V4=}	=>	{V2=}	0.05128	1
{V2=Agachamento, V3=LegPress, V4=Adutor}	=>	{V1=Adutor}	0.07692	1
{V1=Adutor, V3=LegPress, V4=Adutor}	=>	{V2=Agachamento}	0.07692	1
{V1=Adutor, V2=Agachamento, V4=Adutor}	=>	{V3=LegPress}	0.07692	1
{V1=Adutor, V2=Agachamento, V3=LegPress}	=>	{V4=Adutor}	0.07692	1
{V1=LegPress, V2=Gemeos, V3=Agachamento}	=>	{V4=}	0.07692	1

TABELA 23 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 7).

{V2=Gemeos, V3=Agachamento, V4=}	=>	{V1=LegPress}	0.07692	1
{V1=LegPress, V3=Agachamento, V4=}	=>	{V2=Gemeos}	0.07692	1
{V1=LegPress, V2=Gemeos, V4=}	=>	{V3=Agachamento}	0.07692	1
{V1=Gemeos, V2=, V3=}	=>	{V4=}	0.05128	1
{V1=Gemeos, V2=, V4=}	=>	{V3=}	0.05128	1
{V1=Gemeos, V3=, V4=}	=>	{V2=}	0.05128	1
{V2=Gemeos, V3=Bicicleta, V4=Esteira}	=>	{V1=Extensor}	0.02564	1
{V1=Extensor, V2=Gemeos, V3=Bicicleta}	=>	{V4=Esteira}	0.02564	1
{V1=Extensor, V2=Gemeos, V4=Esteira}	=>	{V3=Bicicleta}	0.02564	1
{V2=AgachamentoSmith, V3=Bicicleta, V4=Esteira}	=>	{V1=Extensor}	0.15384	1
{V1=LegPress, V2=, V3=}	=>	{V4=}	0.23076	1
{V1=LegPress, V2=, V4=}	=>	{V3=}	0.23076	1
{V1=LegPress, V3=, V4=}	=>	{V2=}	0.23076	1
{V1=Extensor}	=>	{V3=Bicicleta}	0.30769	9,23077E+14
{V4=Esteira}	=>	{V1=Extensor}	0.28205	9,16667E+14
{V1=Extensor, V4=Esteira}	=>	{V3=Bicicleta}	0.25641	9,09091E+14
{V2=AgachamentoSmith}	=>	{V1=Extensor}	0.23076	9
{V1=Extensor, V2=AgachamentoSmith}	=>	{V3=Bicicleta}	0.20512	8,88889E+14
{V2=AgachamentoSmith, V4=Esteira}	=>	{V1=Extensor}	0.17948	875
{V2=Gemeos}	=>	{V1=LegPress}	0.15384	8,57143E+14
{V1=Extensor, V2=AgachamentoSmith, V4=Esteira}	=>	{V3=Bicicleta}	0.15384	8,57143E+14
{V1=Extensor}	=>	{V4=Esteira}	0.28205	8,46154E+14
{V3=Bicicleta}	=>	{V4=Esteira}	0.25641	8,33333E+14
{V4=Esteira}	=>	{V3=Bicicleta}	0.25641	8,33333E+14
{V1=Extensor, V3=Bicicleta}	=>	{V4=Esteira}	0.25641	8,33333E+14
{V4=}	=>	{V2=}	0.33333	8125
{V4=}	=>	{V3=}	0.33333	8125
{V2=AgachamentoSmith}	=>	{V3=Bicicleta}	0.20512	8
{V2=AgachamentoSmith}	=>	{V4=Esteira}	0.20512	8

TABELA 24 - REGRAS DE ASSOCIAÇÃO - MUSCULAÇÃO (PARTE 8).

{V1=LegPress}	=>	{V4=}	0.30769	8
{V1=Extensor, V2=AgachamentoSmith}	=>	{V4=Esteira}	0.17948	7,77778E+14
{V4=}	=>	{V1=LegPress}	0.30769	75
{V2=AgachamentoSmith, V3=Bicicleta}	=>	{V4=Esteira}	0.15384	75
{V2=AgachamentoSmith, V4=Esteira}	=>	{V3=Bicicleta}	0.15384	75
{V1=LegPress, V4=}	=>	{V2=}	0.23076	75
{V1=LegPress, V4=}	=>	{V3=}	0.23076	75
{V1=Extensor, V2=AgachamentoSmith, V3=Bicicleta}	=>	{V4=Esteira}	0.15384	75

APÊNDICE H - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

1 - Classificação de Imagem (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
cifar10 = tf.keras.datasets.cifar10
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

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

FIGURA 29 - CNN - SAÍDA DA NORMALIZAÇÃO DE DADOS.

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

```
K = len(set(y_train))
#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)
#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 do modelo
model.summary()
```

FIGURA 30 - CNN - SAÍDA CRIAÇÃO DA CNN.

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)
```

FIGURA 31 - CNN - SAÍDA COMPILAÇÃO E TREINAMENTO DO MODELO.

```

Epoch 1/15
1563/1563 [=====] - 16s 6ms/step - loss: 1.5600 - accuracy: 0.4304 - val_loss: 1.2886 - val_accuracy: 0.5242
Epoch 2/15
1563/1563 [=====] - 8s 5ms/step - loss: 1.2627 - accuracy: 0.5457 - val_loss: 1.1283 - val_accuracy: 0.6040
Epoch 3/15
1563/1563 [=====] - 9s 6ms/step - loss: 1.1300 - accuracy: 0.5968 - val_loss: 1.0290 - val_accuracy: 0.6394
Epoch 4/15
1563/1563 [=====] - 8s 5ms/step - loss: 1.0288 - accuracy: 0.6361 - val_loss: 0.9900 - val_accuracy: 0.6446
Epoch 5/15
1563/1563 [=====] - 9s 6ms/step - loss: 0.9595 - accuracy: 0.6611 - val_loss: 0.9217 - val_accuracy: 0.6778
Epoch 6/15
1563/1563 [=====] - 9s 6ms/step - loss: 0.9048 - accuracy: 0.6792 - val_loss: 0.8967 - val_accuracy: 0.6889
Epoch 7/15
1563/1563 [=====] - 8s 5ms/step - loss: 0.8526 - accuracy: 0.6975 - val_loss: 0.8644 - val_accuracy: 0.6992
Epoch 8/15
1563/1563 [=====] - 9s 6ms/step - loss: 0.8123 - accuracy: 0.7104 - val_loss: 0.8678 - val_accuracy: 0.7015
Epoch 9/15
1563/1563 [=====] - 9s 5ms/step - loss: 0.7682 - accuracy: 0.7258 - val_loss: 0.8473 - val_accuracy: 0.7053
Epoch 10/15
1563/1563 [=====] - 8s 5ms/step - loss: 0.7398 - accuracy: 0.7369 - val_loss: 0.8433 - val_accuracy: 0.7045
Epoch 11/15
1563/1563 [=====] - 10s 6ms/step - loss: 0.7067 - accuracy: 0.7503 - val_loss: 0.8249 - val_accuracy: 0.7194
Epoch 12/15
1563/1563 [=====] - 9s 6ms/step - loss: 0.6781 - accuracy: 0.7588 - val_loss: 0.8043 - val_accuracy: 0.7199
Epoch 13/15
1563/1563 [=====] - 8s 5ms/step - loss: 0.6549 - accuracy: 0.7677 - val_loss: 0.8259 - val_accuracy: 0.7160
Epoch 14/15
1563/1563 [=====] - 9s 6ms/step - loss: 0.6351 - accuracy: 0.7740 - val_loss: 0.8092 - val_accuracy: 0.7202
Epoch 15/15
1563/1563 [=====] - 8s 5ms/step - loss: 0.6138 - accuracy: 0.7830 - val_loss: 0.8124 - val_accuracy: 0.7175

```

```

#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()

```

FIGURA 32 - CNN - GRÁFICO DE TREINO: LOSS.

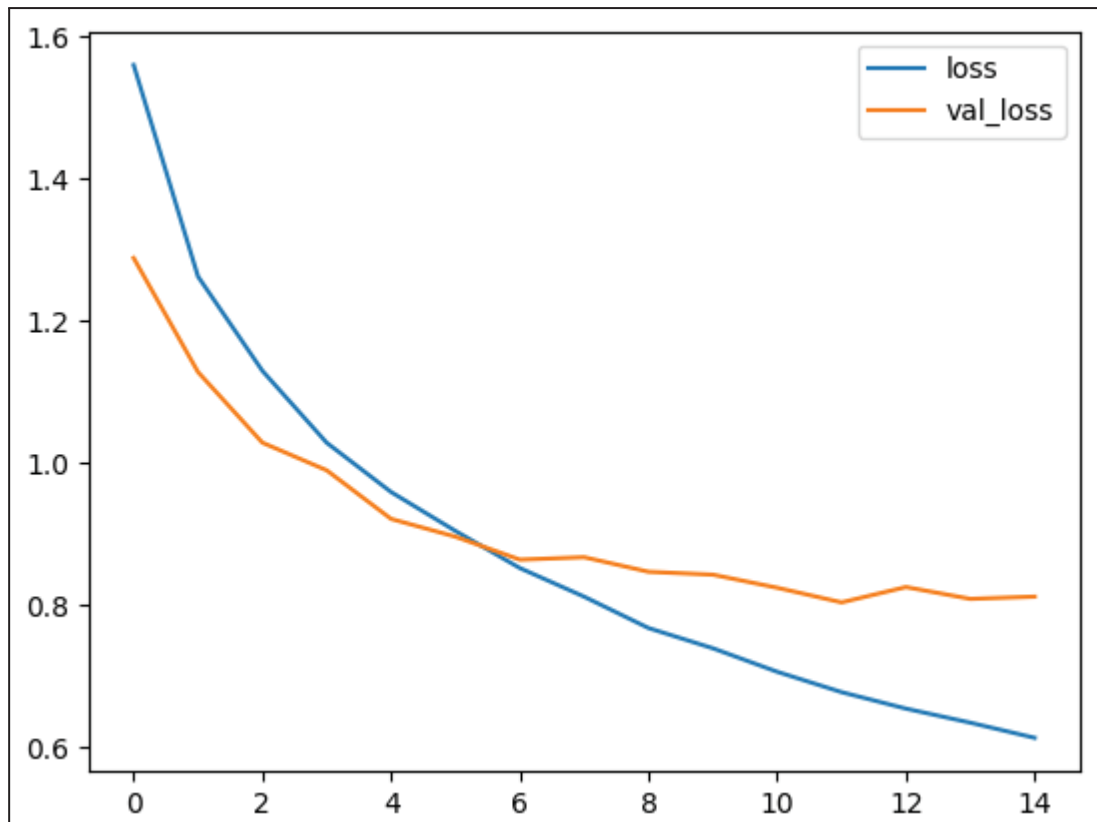
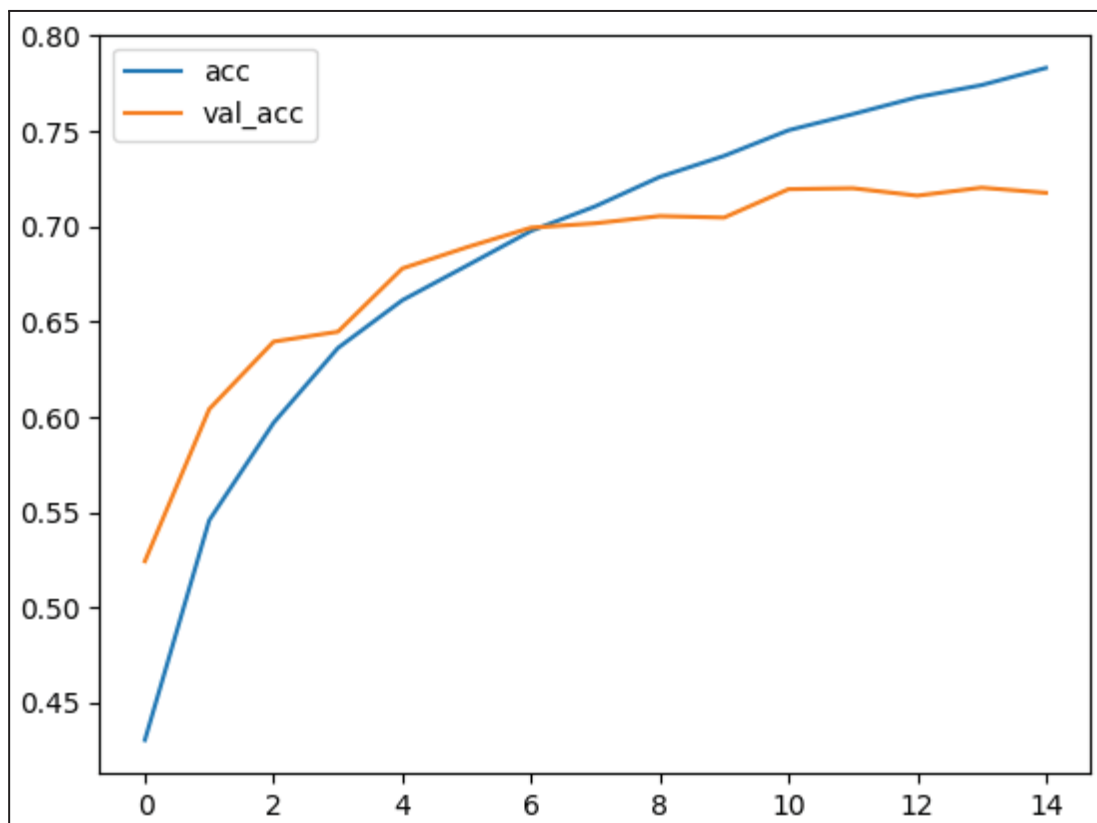
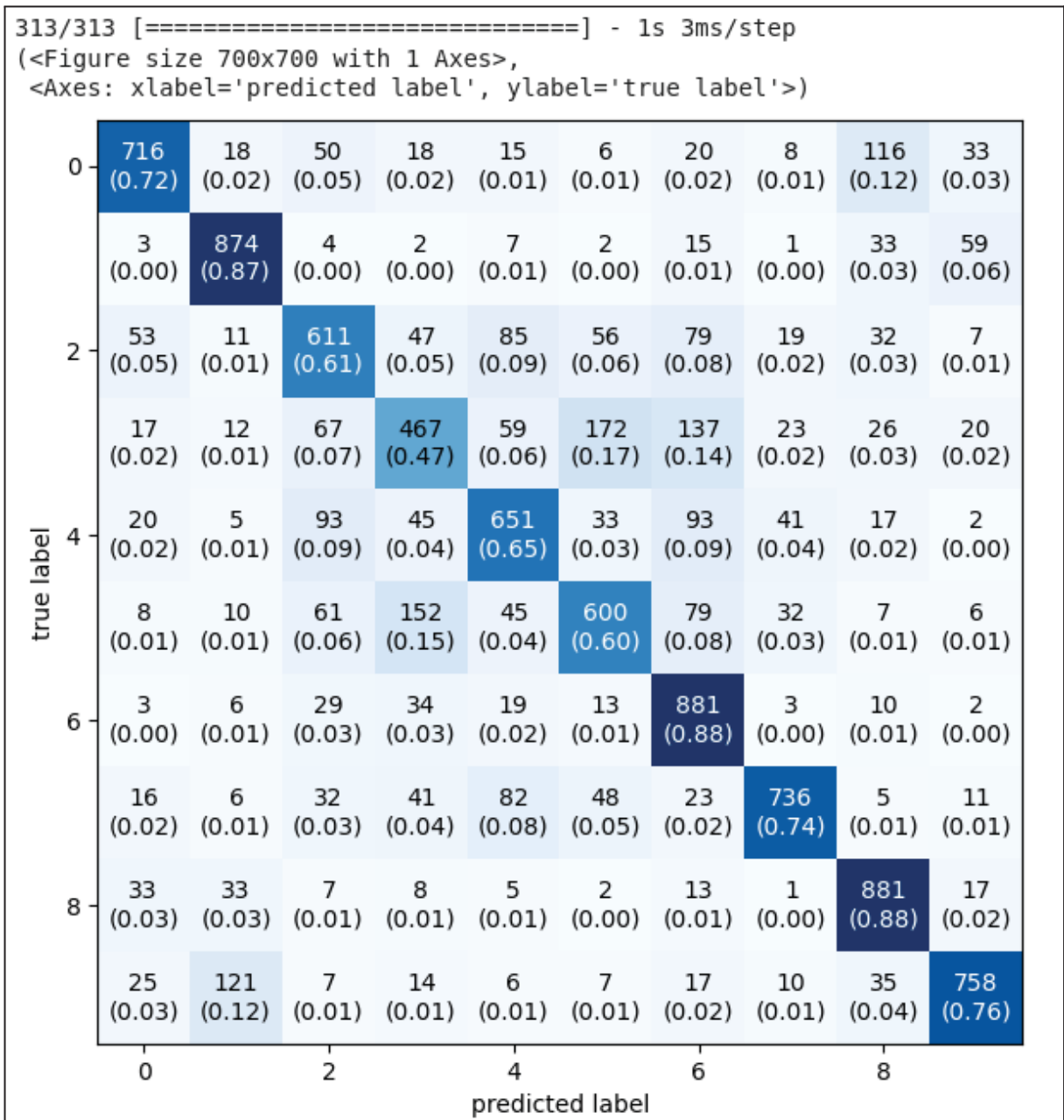


FIGURA 33 - CNN - GRÁFICO DE TREINO: ACURÁCIA.



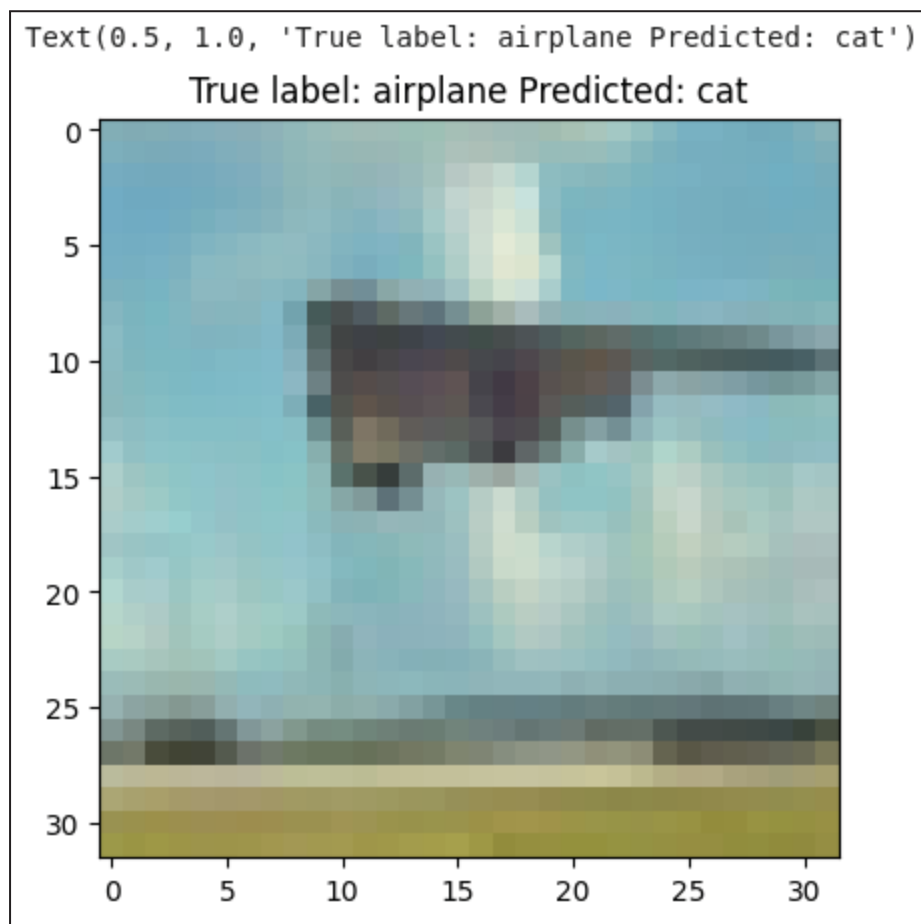
```
#Efetuar predições na base de teste
y_pred = model.predict(x_test).argmax(axis=1)
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7,7), show_normed=True)
```

FIGURA 34 - CNN - PREDIÇÃO E MATRIZ DE CONFUSÃO.



```
# 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]]))
```

FIGURA 35 - CNN - CLASSIFICAÇÕES ERRADAS.



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

```

FIGURA 36 - RNN - SAÍDA CARGA DA BASE.

```

--2024-07-27 14:42:04--  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  1.24MB/s   in 0.4s

2024-07-27 14:42:05 (1.24 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)
```

FIGURA 37 - RNN - TOKENIZAÇÃO DOS DADOS.

7122 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)
```

FIGURA 38 - RNN - *PADDING*.

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

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

FIGURA 39 - RNN - DEFINIÇÃO E VISUALIZAÇÃO DO MODELO.

Model: "model"		
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 189)]	0
embedding (Embedding)	(None, 189, 20)	142460
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6
Total params: 142986 (558.54 KB)		
Trainable params: 142986 (558.54 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))
```

FIGURA 40 - RNN - COMPILAÇÃO E TREINAMENTO DO MODELO.

```
Epoch 1/5
117/117 [=====] - 20s 142ms/step - loss: 0.5134 - accuracy: 0.8495 - val_loss: 0.3332 - val_accuracy: 0.8651
Epoch 2/5
117/117 [=====] - 6s 50ms/step - loss: 0.2217 - accuracy: 0.9445 - val_loss: 0.1696 - val_accuracy: 0.9723
Epoch 3/5
117/117 [=====] - 6s 49ms/step - loss: 0.1309 - accuracy: 0.9863 - val_loss: 0.1243 - val_accuracy: 0.9821
Epoch 4/5
117/117 [=====] - 3s 28ms/step - loss: 0.0908 - accuracy: 0.9933 - val_loss: 0.0993 - val_accuracy: 0.9853
Epoch 5/5
117/117 [=====] - 3s 24ms/step - loss: 0.0652 - accuracy: 0.9965 - val_loss: 0.0809 - val_accuracy: 0.9869
```

```
# 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()
```

FIGURA 41 - RNN - FUNÇÃO DE PERDA.

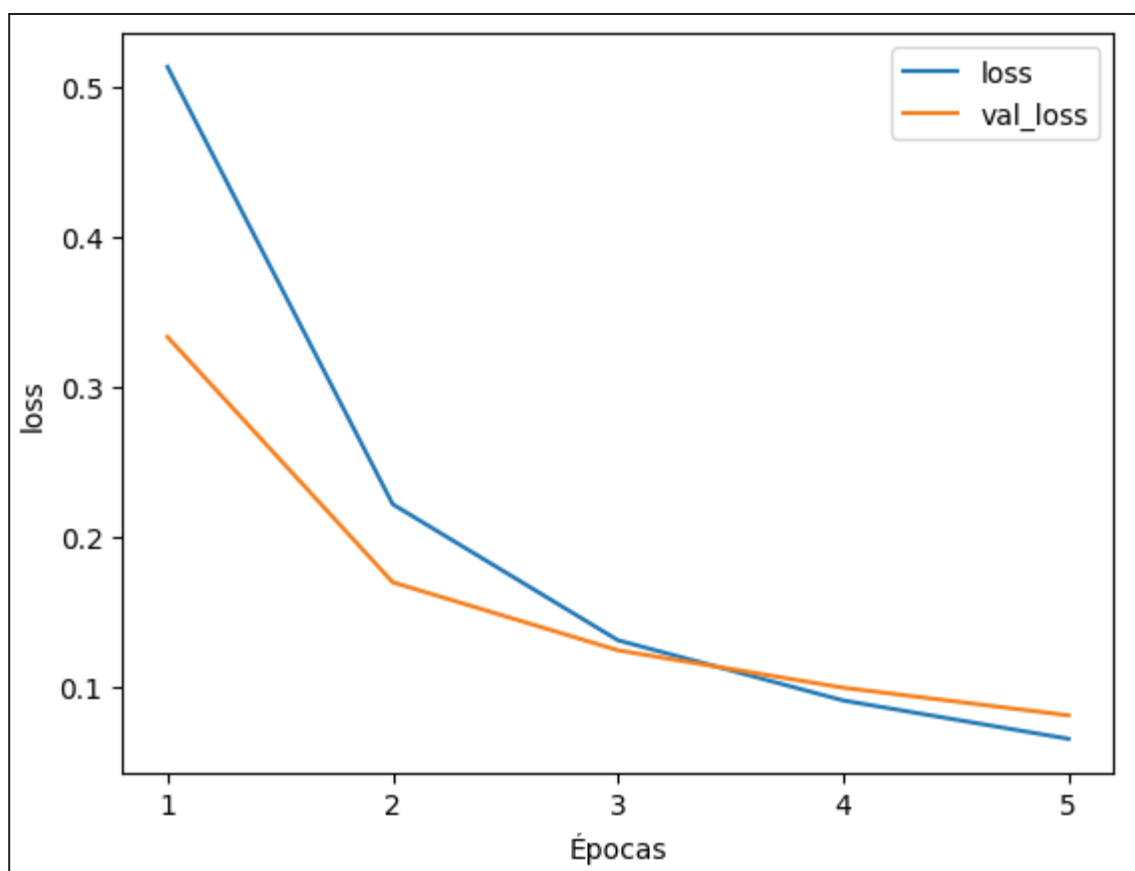
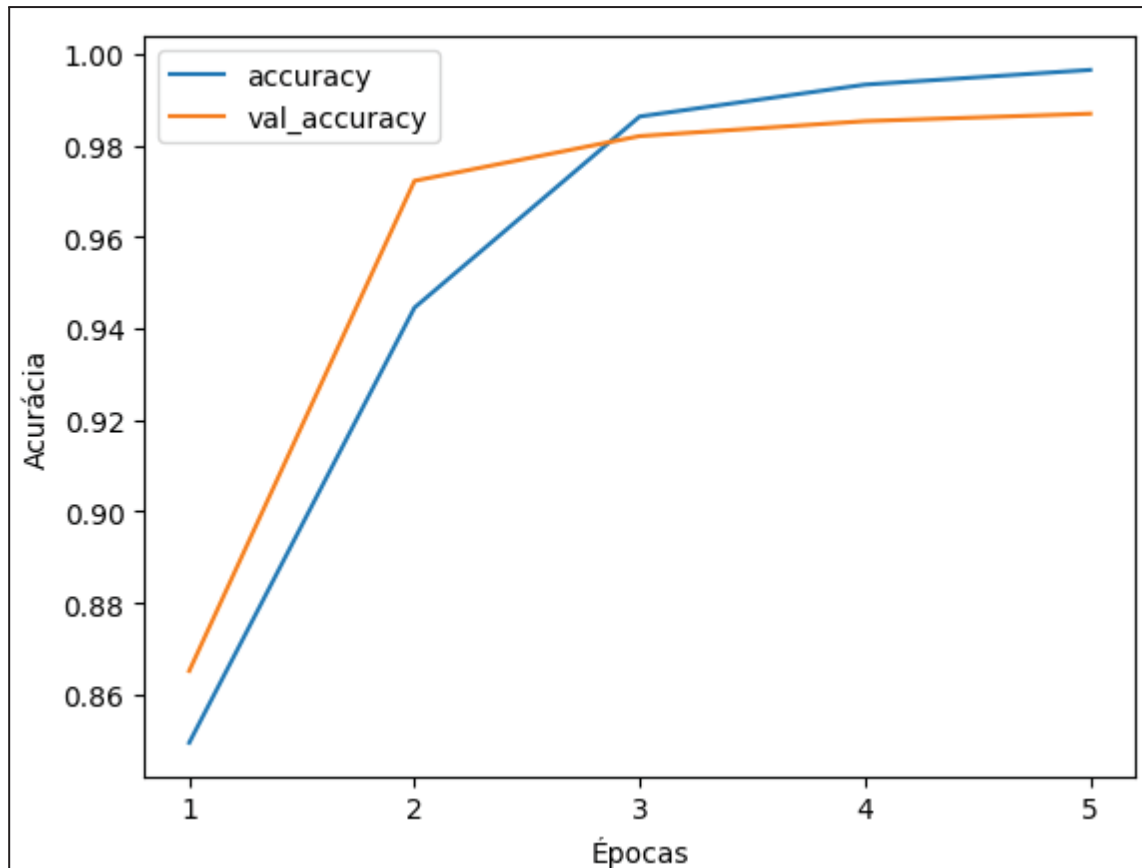


FIGURA 42 - RNN - FUNÇÃO DE ACURÁCIA.



```
# 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")
```

FIGURA 43 - RNN - PREDIÇÃO DE UM NOVO TEXTO.

```
1/1 [=====] - 1s 548ms/step
[[0.6523792]]
SPAM
```

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

# Carrega 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)
# shuffle: embaralha os dados
# Combina elementos consecutivos em lotes (batch)
train_dataset =
tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).batch
(BATCH_SIZE)

# Cria o gerador
def make_generator_model():
```

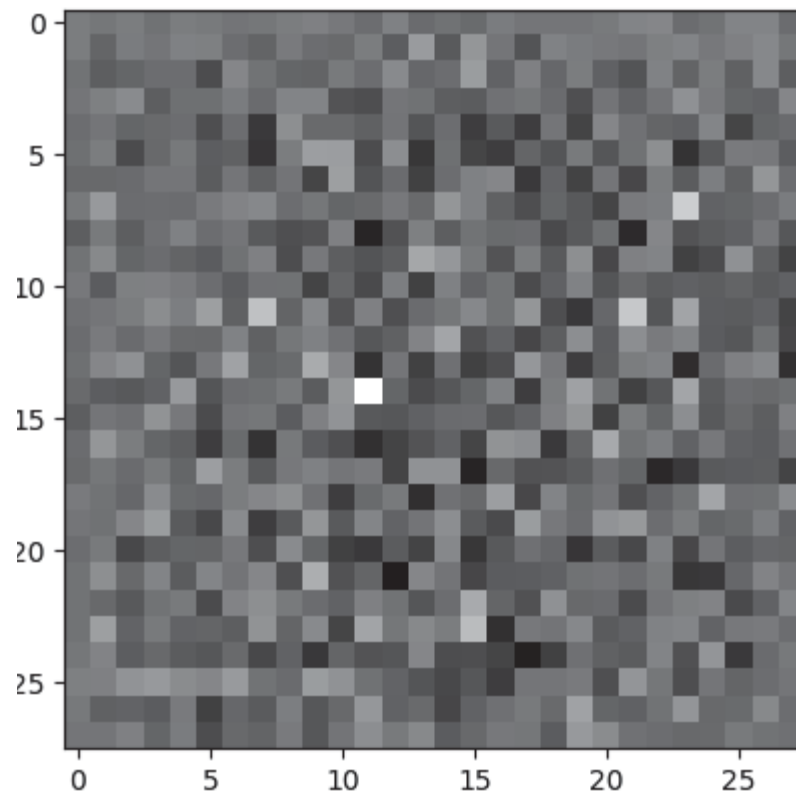
```

model = tf.keras.Sequential()
model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU())
model.add(layers.Reshape((7, 7, 256)))
assert model.output_shape == (None, 7, 7, 256) # None é o 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')

```

FIGURA 44 - GAN - TESTE DO GERADOR.



```
# 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

discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print (decision)
```

```

# Definição das funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

# Função de 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

# Função de 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 em um desligamento
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,
discriminator_optimizer=discriminator_optimizer,
                                generator=generator,
                                discriminator=discriminator)

# Configura o loop de treinamento
EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16

# Seed
seed = tf.random.normal([num_examples_to_generate, noise_dim])

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

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)

```

FIGURA 45 - GAN - VERIFICAÇÃO DOS GRADIENTES

```
[<tf.Tensor: shape=(), dtype=float32, numpy=30.0>, <tf.Tensor: shape=(), dtype=float32, numpy=12.0>]
```

```

# 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)

    # Produz imagens do GIF
    display.clear_output(wait=True)
    generate_and_save_images(generator, epoch + 1, seed)

    # Salva o modelo a cada 15 epocas
    if (epoch + 1) % 15 == 0:
        checkpoint.save(file_prefix = checkpoint_prefix)

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

# Gera depois da epoca final
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):
    predictions = model(test_input, training=False)
    fig = plt.figure(figsize=(4,4))

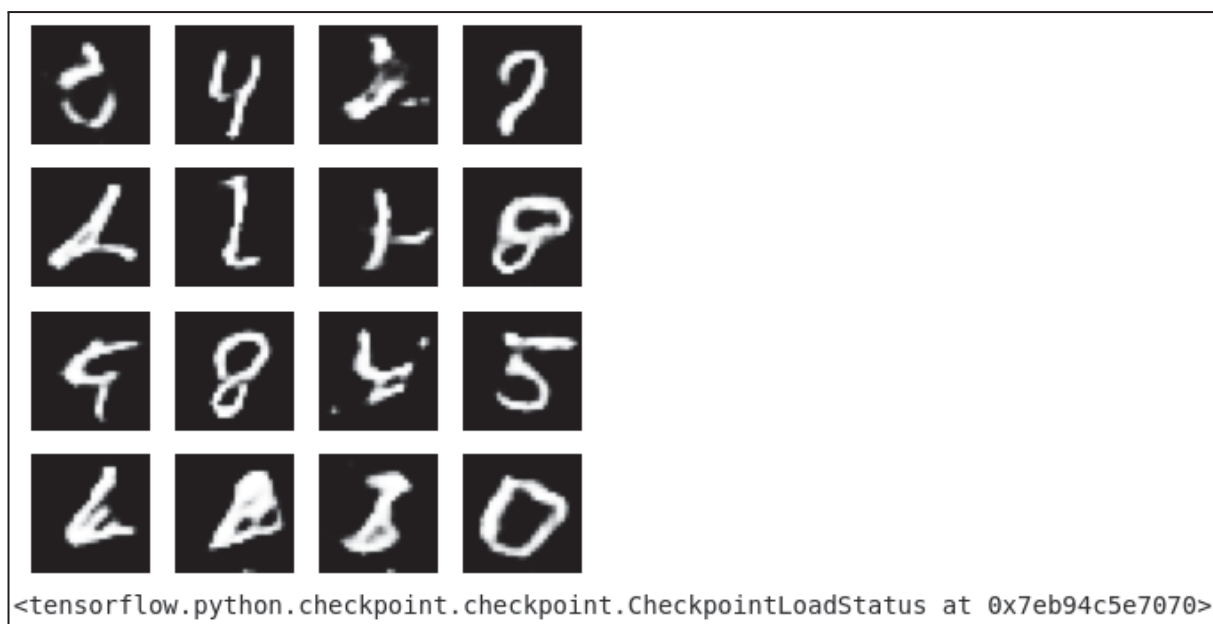
    for i in range(predictions.shape[0]):
        plt.subplot(4, 4, i+1)
        plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
        plt.axis('off')

    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))

```

FIGURA 46 - GAN - TREINAMENTO DO MODELO



```
def display_image(epoch_no):
    return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))

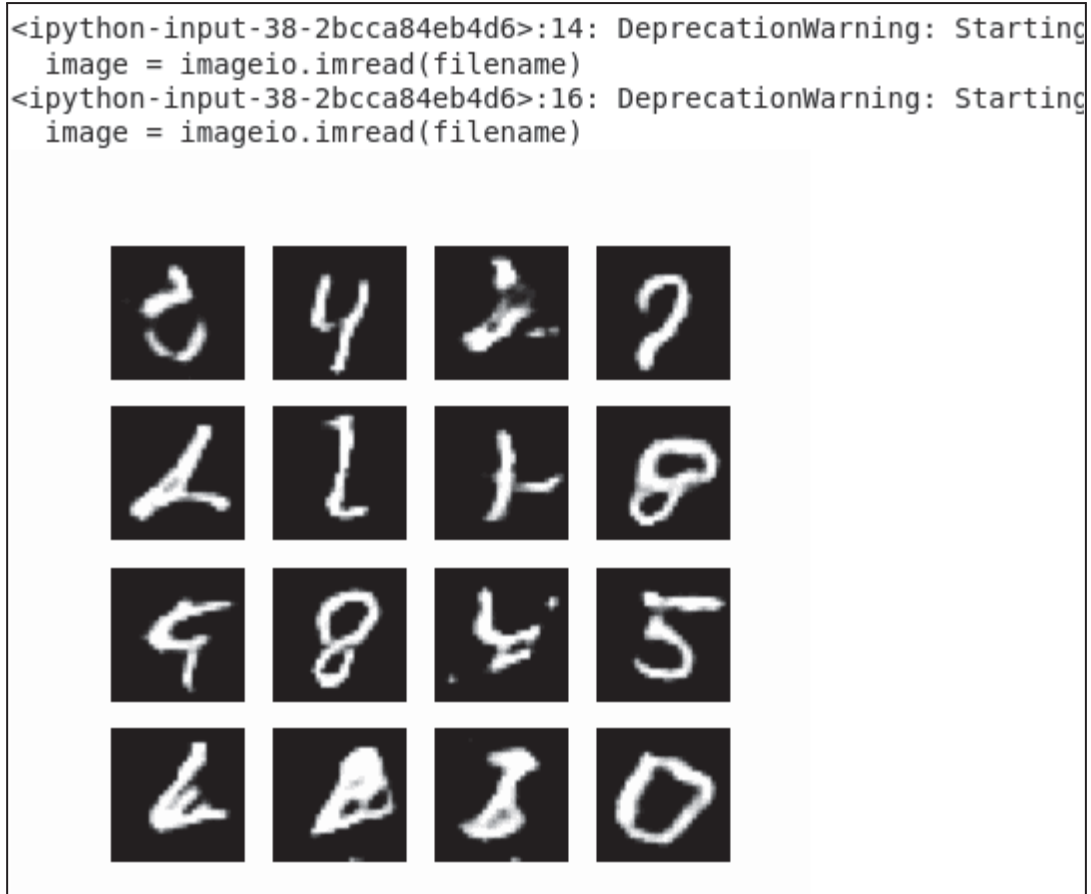
display_image(EPOCHS)

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)
```

FIGURA 47 - GAN - CRIAÇÃO DO GIF



4 - Tradutor de Textos (Transformer)

```

!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow datasets
!pip install -U tensorflow-text==2.15.0

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

import numpy as np
import matplotlib.pyplot as plt

```

```

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

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

examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en',
with_info=True,
                                as_supervised=True)
train_examples, val_examples = examples['train'], examples['validation']

# Verifica 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'))

```

FIGURA 48 - TRANSFORMER - VERIFICAÇÃO DO DATASET.

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 outr em en
# tokenizers .en tokeniza e destokeniza
tokenizers = tf.saved_model.load(model_name)

```

```

# Pipeline de entrada
# Codificar/tokenizar lotes de texto puro

def tokenize_pairs(pt, en):
    pt = tokenizers.pt.tokenize(pt)
    # Converte ragged (irregular, tam variável) para dense
    # Faz padding com zeros.
    pt = pt.to_tensor()

    en = tokenizers.en.tokenize(en)
    # ragged -> dense
    en = en.to_tensor()

    return pt, en

# Pipeline simples: processa, embaralha, agrupa os dados, prefetch
# Datasets de entrada terminam com prefetch

BUFFER_SIZE = 20000
BATCH_SIZE = 64

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

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

# Codificação posicional

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

```

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

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

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

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

# Codificação posicional
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]

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

FIGURA 49 - TRANSFORMER - CODIFICAÇÃO POSICIONAL.

(1, 2048, 512)

```
# Cria uma mascara 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 extradimensions to add the padding
    # to the attention logits
    return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_len)

# Máscara futura, usada no decoder
```

```

def create_look_ahead_mask(size):
    # zera o triângulo inferior
    mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
    return mask # (seq_len, seq_len)

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

    #Converte matmul qk para float32
    dk = tf.cast(tf.shape(k)[-1], tf.float32)
    # divide por sqrt (d_k)
    scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)

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

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

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

    return output, attention_weights

# Atenção Multi-cabeças
class MultiHeadAttention(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.num_heads = num_heads
        self.d_model = d_model

        assert d_model % self.num_heads == 0

        self.depth = d_model // self.num_heads

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

```

```

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

def split_heads(self, x, batch_size):
    # Separa a última dimensão em (num_heads, depth). Transpões o resultado
    para o shape (batch_size, num_heads, self.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 vvalores 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, att_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, att_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, att_weights_block1, att_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, 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 = self.enc_layers[i](x, training, mask)

    return x # (batch_size, target_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'.format(i+1)] = block1
        attention_weights[f'decoder_layer{i+1}_block2'.format(i+1)] = 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 the first argument
        inp, tar = inputs

        enc_padding_mask, look_ahead_mask, dec_padding_mask =
self.create_masks(inp, tar)

        # (batch_size, target_seq_len, d_model)

```

```

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

        dec_output, attention_weights = self.decoder(tar, enc_output, training,
        look_ahead_mask, dec_padding_mask)

        # (batch_size, target_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 2nd attention block in the decoder
    # This padding mask is used to mask the endocer 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)
    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)

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

```

```

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

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

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))

    trains_loss(loss)
    trains_accuracy(accuracy_function(tar_real, predictions))

```

```

for epoch in range(EPOCHS):
    start = time.time()
    trains_loss.reset_states()
    trains_accuracy.reset_states()
    # imp -> 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
{trains_loss.result():.4f} Accuracy {trains_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 {trains_loss.result():.4f} Accuracy
{trains_accuracy.result():.4f}')

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

```

FIGURA 50 - TRANSFORMER - LOG ÉPOCA 1.

```

Epoch 1 Batch 0 Loss 8.8689 Accuracy 0.0000
Epoch 1 Batch 50 Loss 8.8056 Accuracy 0.0022
Epoch 1 Batch 100 Loss 8.6968 Accuracy 0.0228
Epoch 1 Batch 150 Loss 8.5776 Accuracy 0.0311
Epoch 1 Batch 200 Loss 8.4326 Accuracy 0.0351
Epoch 1 Batch 250 Loss 8.2618 Accuracy 0.0376
Epoch 1 Batch 300 Loss 8.0719 Accuracy 0.0418
Epoch 1 Batch 350 Loss 7.8759 Accuracy 0.0485
Epoch 1 Batch 400 Loss 7.6899 Accuracy 0.0551
Epoch 1 Batch 450 Loss 7.5215 Accuracy 0.0633
Epoch 1 Batch 500 Loss 7.3745 Accuracy 0.0722
Epoch 1 Batch 550 Loss 7.2388 Accuracy 0.0805
Epoch 1 Batch 600 Loss 7.1107 Accuracy 0.0891
Epoch 1 Batch 650 Loss 6.9914 Accuracy 0.0968
Epoch 1 Batch 700 Loss 6.8823 Accuracy 0.1039
Epoch 1 Batch 750 Loss 6.7842 Accuracy 0.1102
Epoch 1 Batch 800 Loss 6.6901 Accuracy 0.1165
Epoch 1 Loss 6.6739 Accuracy 0.1175
Time taken for 1 epoch: 179.96 secs

```

FIGURA 51 - TRANSFORMER - LOG ÉPOCA 20.

```

Epoch 20 Batch 0 Loss 1.2724 Accuracy 0.7160
Epoch 20 Batch 50 Loss 1.3922 Accuracy 0.6907
Epoch 20 Batch 100 Loss 1.4050 Accuracy 0.6882
Epoch 20 Batch 150 Loss 1.4173 Accuracy 0.6853
Epoch 20 Batch 200 Loss 1.4218 Accuracy 0.6845
Epoch 20 Batch 250 Loss 1.4279 Accuracy 0.6832
Epoch 20 Batch 300 Loss 1.4296 Accuracy 0.6831
Epoch 20 Batch 350 Loss 1.4299 Accuracy 0.6830
Epoch 20 Batch 400 Loss 1.4326 Accuracy 0.6826
Epoch 20 Batch 450 Loss 1.4372 Accuracy 0.6816
Epoch 20 Batch 500 Loss 1.4388 Accuracy 0.6815
Epoch 20 Batch 550 Loss 1.4405 Accuracy 0.6813
Epoch 20 Batch 600 Loss 1.4407 Accuracy 0.6815
Epoch 20 Batch 650 Loss 1.4442 Accuracy 0.6811
Epoch 20 Batch 700 Loss 1.4485 Accuracy 0.6804
Epoch 20 Batch 750 Loss 1.4514 Accuracy 0.6799
Epoch 20 Batch 800 Loss 1.4556 Accuracy 0.6792
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-4
Epoch 20 Loss 1.4564 Accuracy 0.6790
Time taken for 1 epoch: 100.80 secs

```

```

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

    def __call__(self, sentence, max_length=20):
        # input sentence in portuguese, hence adding the stard 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)
    # select the last token from the seq_len dimension
    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

# Efetuando uma tradução com o Transformer

translator = Translator(tokenizers, transformer)
sentence = "Eu gostaria de saber ."
translated_text, translated_tokens, attention_weights =
translator(tf.constant(sentence))

print(translated_text)

```

FIGURA 52 - TRANSFORMER - EFETUANDO UMA TRADUÇÃO.

```
tf.Tensor(b'i read about triumphs in the encyclopedia .', shape=(), dtype=string)
```

APÊNDICE I - BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

Estudo de Caso: Análise de Dados de Mobilidade Urbana com Dados Abertos de Transporte Público

Descrição

Este estudo de caso visa propor uma arquitetura de Big Data para análise de padrões de mobilidade urbana em Curitiba utilizando dados públicos. Os dados disponíveis incluem e não se limitam a: frota de veículos, linhas existentes, itinerários, equipamentos públicos, terminais de transporte, ciclovias, ciclorrotas, sistema viário, arruamento, quadras e dados populacionais do Censo IBGE.

Entre os objetivos finais desejados em uma eventual implantação da arquitetura estão, a possibilidade de prever congestionamentos, identificar a necessidade de novas linhas de transporte público, propor rotas otimizadas para ciclovias, propor novas localizações de terminais de ônibus. Visualizações interativas em dashboards seriam utilizadas para facilitar a interpretação dos resultados, objetivando melhorar a eficiência do transporte público e a qualidade de vida dos cidadãos.

Ingestão e Processamento de Dados

Entre os dados relevantes para a análise estariam: a relação de linhas, itinerários, frotas e horários do transporte público, que seriam disponibilizados mensalmente pelo portal de dados abertos da Prefeitura de Curitiba; além disso, dados geográficos de arruamento, zoneamento, eixos de ruas e ciclovias da cidade, fornecidos em formato vetorizado de pontos, linhas e polígonos (SHP), também seriam disponibilizados pela Prefeitura, porém sem periodicidade definida. Adicionalmente, os dados no formato GTFS seriam acessíveis via Webservice.

Para extrair informações de um arquivo com extensão “.json” e processá-las em um servidor Hadoop, primeiro seria necessário garantir que o Hadoop está instalado e configurado, incluindo o HDFS (Hadoop Distributed File System) e o YARN para gerenciamento de recursos. Em seguida, mover o arquivo JSON para o HDFS usando o comando “hadoop fs -put /caminho/para/arquivo.json /diretorio/destino/no/HDFS”. Após o arquivo estar no HDFS, seria possível processá-lo utilizando ferramentas como Apache Hive ou Apache Pig.

No Hive, é possível criar uma tabela externa que aponta para o arquivo JSON e usa a SerDe (Serializer/Deserializer) específica para JSON. Por exemplo, a criação da tabela poderia ser feita com o seguinte comando SQL: “CREATE EXTERNAL TABLE json_table (key1 STRING, key2 STRING, key3 INT) ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe' LOCATION '/user/hadoop/input/';”, onde json_table é o nome da tabela e key1, key2, key3 são campos que correspondem às chaves no arquivo JSON. Após criar a tabela, seria possível executar consultas SQL sobre esses dados, como ‘SELECT key1, key2 FROM json_table WHERE key3 > 10;’.

Alternativamente, no Apache Pig, poderia ser escrito um script para carregar e processar o JSON. Um exemplo seria: “data = LOAD '/user/hadoop/input/dados.json’ USING JsonLoader('key1:chararray, key2:chararray, key3:int');” para carregar o arquivo, seguido de comandos para filtrar e projetar os dados. Após o processamento, os resultados podem ser armazenados de volta no HDFS, e para visualizá-los, poderia usar o comando “hadoop fs -cat /user/hadoop/output/part-00000”.

Uma outra possível abordagem, seria escrever um job MapReduce para processar o JSON diretamente, utilizando bibliotecas como ‘org.json’ ou ‘Jackson’ no código Java. O processamento final pode ser acessado no HDFS com o comando ‘hadoop fs -cat’, que exibirá os resultados após a execução das consultas ou jobs.

Para extrair informações de um arquivo ‘.dbf’ e processá-las em um servidor Hadoop, o primeiro passo é converter o arquivo para um formato compatível, como CSV ou JSON, já que o Hadoop não suporta diretamente arquivos ‘.dbf’. A conversão pode ser feita utilizando a linguagem Python com a biblioteca ‘dbfread’, que permite ler arquivos ‘.dbf’ e convertê-los para CSV. Primeiro, deve-se instalar a biblioteca com o comando ‘pip install dbfread’. Depois, utilizar um script Python para realizar a conversão, que carrega o arquivo ‘.dbf’ e escreve seu conteúdo em um arquivo CSV. Após a conversão, o arquivo CSV pode ser carregado no HDFS (Hadoop Distributed File System) com o comando ‘hadoop fs -put /caminho/para/saida.csv /diretorio/destino/no/HDFS’, onde é especificado o caminho do arquivo convertido e o diretório de destino no HDFS.

Com o arquivo no HDFS, ele pode ser processado usando ferramentas como Apache Hive ou Apache Pig. No Apache Hive, você cria uma tabela que aponta para o arquivo CSV com a seguinte instrução SQL: ‘CREATE EXTERNAL TABLE dbf_table (coluna1 STRING, coluna2 INT, coluna3 DOUBLE) ROW FORMAT DELIMITED FIELDS TERMINATED BY ‘,’ STORED AS TEXTFILE LOCATION '/user/hadoop/input/’;’. Isso define uma tabela externa baseada no CSV que está no HDFS, permitindo que se execute consultas SQL diretamente

nos dados, como ``SELECT coluna1, coluna2 FROM dbf_table WHERE coluna3 > 100;``. Alternativamente, no Apache Pig, se pode carregar o arquivo CSV com o comando ``LOAD`` e processar os dados com operações como filtro e projeção. Por exemplo, após carregar o arquivo com ``LOAD 'user/hadoop/input/saida.csv' USING PigStorage(',') AS (coluna1:chararray, coluna2:int, coluna3:double);``, se pode aplicar um filtro para selecionar registros com um valor maior que 100 em ``coluna3`` e armazenar o resultado de volta no HDFS.

Por fim, os dados processados podem ser acessados no HDFS utilizando o comando ``hadoop fs -cat /user/hadoop/output/part-00000``, que exibe os resultados diretamente no terminal. Se preferir, também seria possível desenvolver um job MapReduce personalizado para processar o CSV, utilizando bibliotecas específicas para leitura e manipulação de arquivos de texto no Hadoop.

Armazenamento e análise

Para o armazenamento dos dados de maneira distribuída e escalável, utilizaríamos o HDFS. O HBase seria empregado para gerenciar dados que demandariam acesso rápido e aleatório, sendo uma solução NoSQL construída sobre o HDFS.

O processamento e a transformação dos dados seriam feitos com o Apache Spark, aproveitando o Spark MLlib para aplicar algoritmos de machine learning e gerar modelos preditivos. Para a criação de um data warehouse sobre o HDFS, utilizaríamos o Apache Hive, que permitiria consultas semelhantes a SQL (HiveQL) para análise de grandes volumes de dados. A integração entre Hive e Spark possibilitaria otimizar o desempenho das consultas e análises.

Vs do Big Data

VOLUME

É necessário garantir o volume, pois se trata de uma grande quantidade de dados, pode-se estimar gigas de informações, dependendo do tamanho da cidade e da quantidade de dispositivos que enviarão as informações. Com o HDFS do Hadoop, é possível armazenar grande quantidade de dados, distribuídos em pequenas partes entre servidores, garantindo o V de Volume dos Big Datas.

VARIEDADE

Devido a quantidade de fontes de informações, e preciso garantir a variedade, pois são diversos dispositivos enviando seus dados, pode ser em diversos formatos, dependendo do dispositivo, então, é necessário garantir o recebimento de todos os tipos de dados. Com o Hive, HBase ou Cassandra, é possível armazenar uma grande variedade de dados, sendo possível armazenar arquivos, inserir informações no SQL ou mesmo enviar arquivos logs, json e afins. Garantindo assim o V de Variedade do Big Data.

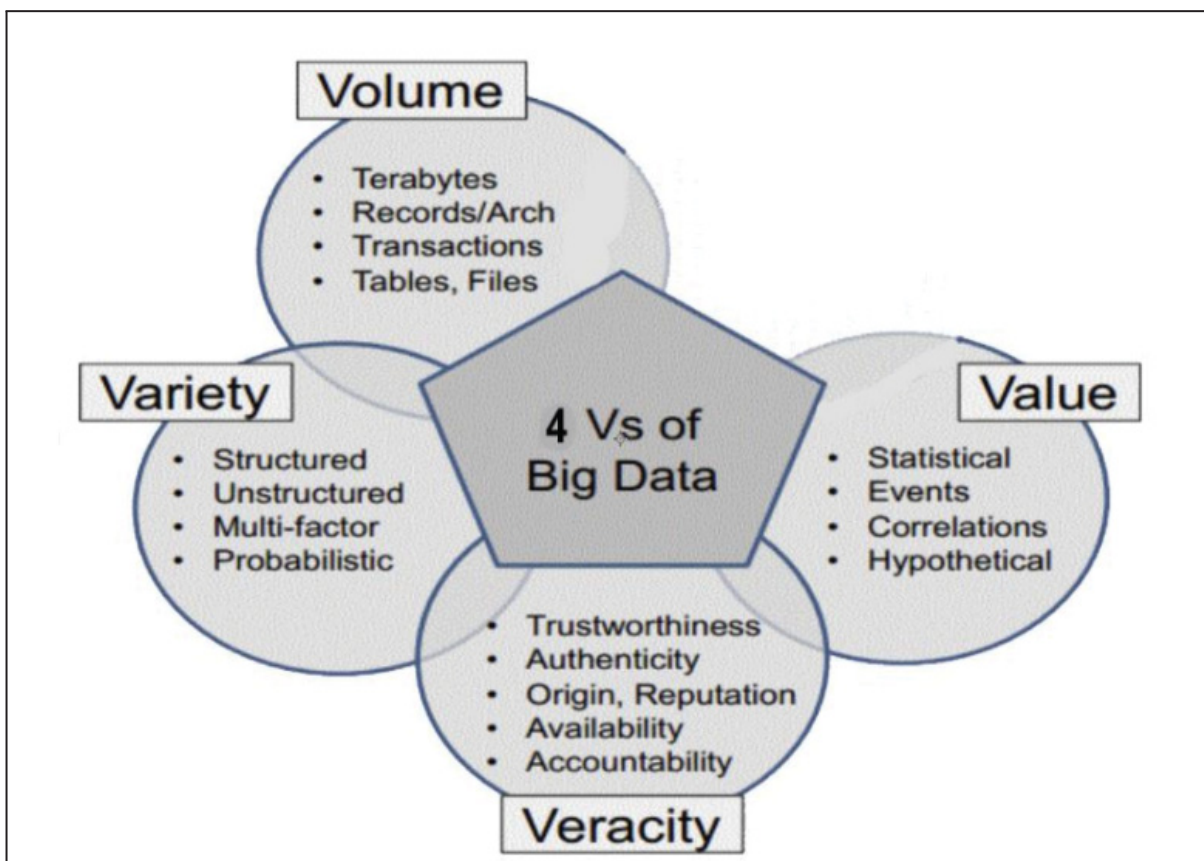
VERACIDADE

Se tratando de dados que serão usados para gerar informações para a população, é necessário garantir a veracidade dos dados. O modelo a ser usado para prever, identificar e propor requer dados confiáveis. A partir do Hadoop, pode-se garantir a disponibilidade dos dados, a segurança, o monitoramento e auditoria das informações, garantindo assim o V de Veracidade do Big Data.

VALOR

Os dados devem ser relevantes para o modelo proposto, pois dados sem valor não são úteis. Após o processamento inicial com Map-Reduce, os dados são armazenados no HBase, garantindo sua integridade para uso com Drill ou Mahout. Além disso, a coleta de dados deve ser planejada para assegurar o V de Valor do Big Data.

FIGURA 53 - VS DE BIG DATA NECESSÁRIOS PARA O PROJETO.



Visualização dos Dados

Os dados podem ser disponibilizados em tabelas relacionais com as principais consultas já configuradas e automatizadas no Power BI, cujos dashboards interativos facilitam a compreensão e análise dos profissionais das diversas áreas que poderiam fazer uso das

informações mesmo que não possuam profundo conhecimento de ferramentas de análise de dados, assim democratizando as informações obtidas gerando resultados de maior valor.

FIGURA 53 - ARQUITETURA PROPOSTA.



APÊNDICE J - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

De acordo com os treinamentos e testes executados nesse trabalho, obtivemos os seguintes resultados:

1 - Extração de Características (Resultado)

TABELA 25 - RESULTADO EXTRAÇÃO DE CARACTERÍSTICAS - TREINO.

Base de Treino	Sensibilidade	Especificidade	F1-Score
RF - LBP	0.766060	0.789186	0.757797
RF - CNN	0.843231	0.854993	0.842945
SVM - LBP	0.830044	0.860930	0.832626
SVM - CNN	0.831257	0.846108	0.832009
RNA - LBP	0.793358	0.821428	0.792482
RNA - CNN	0.792482	0.792482	0.792482

TABELA 26 - RESULTADO EXTRAÇÃO DE CARACTERÍSTICAS - TESTE.

Base de Teste	Sensibilidade	Especificidade	F1-Score
RF - LBP	0.694581	0.705142	0.696692
RF - CNN	0.772167	0.770336	0.763962
SVM - LBP	0.677200	0.733198	0.686223
SVM - CNN	0.749339	0.740869	0.736752
RNA - LBP	0.640236	0.676364	0.651428
RNA - CNN	0.730198	0.715030	0.711098

O modelo escolhido na extração de características foi o Random Forest CNN que foi melhor em todas as métricas tanto na base de treino como na de testes.

2 - Redes Neurais (Resultado)

TABELA 27 - RESULTADO REDES NEURAIAS.

Redes Neurais - Base de Teste	Sensibilidade	Especificidade	F1-Score
VGG16 com D.A.	0.338421	0.393650	0.307438
VGG16 sem D.A.	0.735396	0.736431	0.726082
ResNet50 com D.A.	0.410808	0.502497	0.354001
ResNet50 sem D.A.	0.888778	0.896246	0.888514

O modelo escolhido dentre as redes neurais foi o Resnet50 sem data augmentation, pois performou melhor em Sensibilidade, Especificidade e em F1-Score.

1 - Extração de Características (Código)

```
import numpy as np
import matplotlib.pyplot as plt
import zipfile
from google.colab.patches import cv2_imshow
import os
import cv2 as cv
from skimage.feature import local_binary_pattern
import pandas as pd

!wget /content/Train_Warwick.zip

!wget /content/Test_Warwick.zip

# Extraí o conteúdo do arquivo zip
!unzip Train_Warwick.zip
!unzip Test_Warwick.zip

# prompt: carregue a pasta Test_4cl_amostra em uma base de dados de imagens
# ## Carregar base de dados
```

```

def load_images_from_folder(folder, category):
    images = []
    for filename in os.listdir(folder):
        img = cv.imread(os.path.join(folder,filename))
        if img is not None:
            images.append((img, filename, category)) # Store image, filename, and
category
    return images

train_images = []

for i in range(4):
    # Pass the integer category to the function
    train_images += load_images_from_folder('Train_4cls_amostra/' + str(i) +
'/', i)

''' separando em treino de validação, separando as imagens de 1 paciente
por
categoria (aprox. 30 imagens por paciente) temos ~20% da base de treino
Pacientes escolhidos: 01, 14, 04 e 06'''

dados_treino = []
dados_val = []

for image, filename, category in train_images:
    if filename.startswith('01'):
        dados_val.append((image, filename, category))
    elif filename.startswith('14'):
        dados_val.append((image, filename, category))
    elif filename.startswith('04'):
        dados_val.append((image, filename, category))
    elif filename.startswith('06'):
        dados_val.append((image, filename, category))
    else:
        dados_treino.append((image, filename, category))

len(dados_treino), len(dados_val)
dados_treino

```

```

# exibindo as imagens com o nome do arquivo e categoria

import matplotlib.pyplot as plt

# Assuming your tuple structure is (image, filename, label)
for image, filename, category in train_images:
    # Display the image
    plt.imshow(cv.cvtColor(image, cv.COLOR_BGR2RGB)) # Convert to RGB for
Matplotlib
    plt.title(f"Filename: {filename}, Label: {category}")
    plt.show()

    # Print other information
    print(f"Image shape: {image.shape}")
    print(f"Filename: {filename}")
    print(f"Label: {category}")
    print("-" * 20) # Separator

#Extraí características utilizando LBP

def extract_lbp_features(image):
    gray_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
    radius = 3 # Adjust as needed
    n_points = 8 * radius # Adjust as needed
    lbp = local_binary_pattern(gray_image, n_points, radius,
method='uniform')
    # Calculate histogram to represent the LBP features
    n_bins = int(lbp.max() + 1)
    hist, _ = np.histogram(lbp, density=True, bins=n_bins, range=(0,
n_bins))
    return hist

# Extraí característica e salva em lista junto com a categoria e filename
data_treino = []
data_val = []

#Base treino
for image, filename, category in dados_treino:
    features_lbp = extract_lbp_features(image)

```

```

        data_treino.append(np.concatenate([filename], [category],
features_lbp)))

#Base validação
for image, filename, category in dados_val:
    features_lbp_val = extract_lbp_features(image)
    data_val.append(np.concatenate([filename], [category],
features_lbp_val)))

#cria um dataframe pandas e salva em csv
#base treino
colunas = ['filename', 'category'] + [f'lbp_{i}' for i in
range(len(features_lbp))]
df_treino_lbp = pd.DataFrame(data_treino, columns=colunas)

df_treino_lbp.to_csv('lbp_features.csv', index=False)

#base validação
colunas = ['filename', 'category', ] + [f'lbp_{i}' for i in
range(len(features_lbp_val))]
df_val_lbp = pd.DataFrame(data_val, columns=colunas)

df_val_lbp.to_csv('lbp_features_val.csv', index=False)

#Extrai características utilizando CNN VGG16

from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.models import Model

# Carrega o modelo VGG16 pré-treinado (sem a camada de classificação)
base_model = VGG16(weights='imagenet', include_top=True)

# Cria um novo modelo que gera as características da camada 'fc2'
model = Model(inputs=base_model.input,
outputs=base_model.get_layer('fc2').output)

```

```

def extract_features(img):
    # Redimensiona a imagem para o tamanho de entrada do VGG16 (224x224)
    img = cv.resize(img, (224, 224))
    # Converte a imagem para um array NumPy
    #x = image.img_to_array(img)
    # Expande as dimensões para criar um batch de uma única imagem
    x = np.expand_dims(img, axis=0)
    # Pré-processa a imagem de acordo com o VGG16
    x = preprocess_input(x)
    # Extrai as características usando o modelo
    features = model.predict(x)
    # Retorna as características como um array NumPy
    return features[0]

features_list = []
features_list_val = []

# BASE DE TREINO
for image, filename, category in dados_treino:
    features_cnn = extract_features(image)
    features_list.append(np.concatenate(( [filename],[category],
features_cnn)))

# BASE DE VALIDAÇÃO
for image, filename, category in dados_val:
    features_cnn_val = extract_features(image)
    features_list_val.append(np.concatenate(( [filename],[category],
features_cnn_val)))

# Create DataFrame and save to CSV
#BASE TREINO
num_features = features_cnn.shape[0] # Get the number of features from
features_cnn
colunas_cnn = ['filename'] + ['category'] + [f'cnn_{i}' for i in
range(num_features)]
df_treino_cnn = pd.DataFrame(features_list, columns=colunas_cnn)

df_treino_cnn.to_csv('cnn_features.csv', index=False)

# BASE VALIDAÇÃO

```

```

num_features = features_cnn_val.shape[0] # Get the number of features from
features_cnn
colunas_cnn = ['filename'] + ['category'] + [f'cnn_{i}' for i in
range(num_features)]
df_val_cnn = pd.DataFrame(features_list_val, columns=colunas_cnn)

df_val_cnn.to_csv('cnn_features_val.csv', index=False)

#função para avaliação de resultados

def avaliar_resultados(y_real, y_pred):
    # Avaliando a acurácia
    accuracy = accuracy_score(y_real, y_pred)
    print(f"Acurácia do modelo: {accuracy}")

    # Exibindo a matriz de confusão
    cm = confusion_matrix(y_real, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(cmap=plt.cm.Blues)
    plt.show()

    # Calculando as métricas por classe e a média macro
    report = classification_report(y_real, y_pred, output_dict=True)
    sensibilidade_macro = report['macro avg']['recall'] # Sensibilidade
    (Recall) macro
    especificidade_macro = report['macro avg']['precision'] #
    Especificidade (Precision) macro
    f1_score_macro = report['macro avg']['f1-score'] # F1-Score macro

    print(f"Sensibilidade (Macro): {sensibilidade_macro}")
    print(f"Especificidade (Macro): {especificidade_macro}")
    print(f"F1-Score (Macro): {f1_score_macro}")

    return {
        "acuracia": accuracy,
        "sensibilidade_macro": sensibilidade_macro,
        "especificidade_macro": especificidade_macro,
        "f1_score_macro": f1_score_macro
    }

```

RANDOM FOREST (LBP e CNN)

```
#LBP

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report

df_treino_lbp = pd.read_csv('lbp_features.csv')

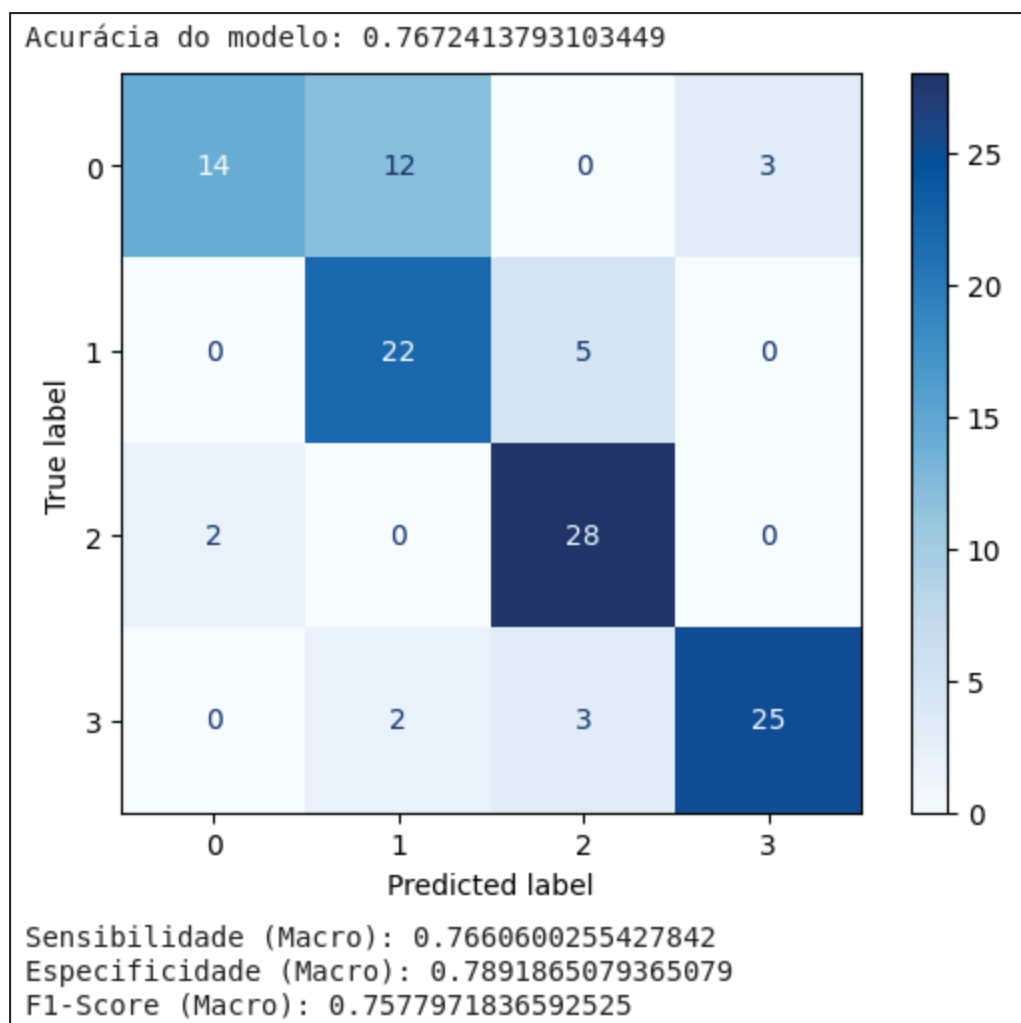
# Separate features (X) and target (y)
X_train = df_treino_lbp.drop(['filename', 'category'], axis=1) #
Caracteristicas
y_train = df_treino_lbp['category'] # alvo

# criando e treinando o modelo
rf_model_lbp = RandomForestClassifier(n_estimators=500,
                                     max_depth=17,
                                     min_samples_split=8,
                                     min_samples_leaf=5,
                                     max_features='sqrt')
#rf_model = RandomForestClassifier()

rf_model_lbp.fit(X_train, y_train)

# Avaliando o modelo para o tuning dos hiperparametros
y_pred = rf_model_lbp.predict(df_val_lbp.drop(['category', 'filename'],
axis=1))
resultado_tunning_rf_lbp = avaliar_resultados(df_val_lbp['category'],
y_pred)
```

FIGURA 54 - RANDOM FOREST - LBP.



```
# CNN
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

#df_treino_cnn = pd.read_csv('cnn_features.csv')

# Separate features (X) and target (y)
X_train = df_treino_cnn.drop(['filename', 'category'], axis=1) #
Caracteristicas
y_train = df_treino_cnn['category'] # alvo

# instanciando com hiperparametros
rf_model_cnn = RandomForestClassifier(n_estimators=500,
```

```

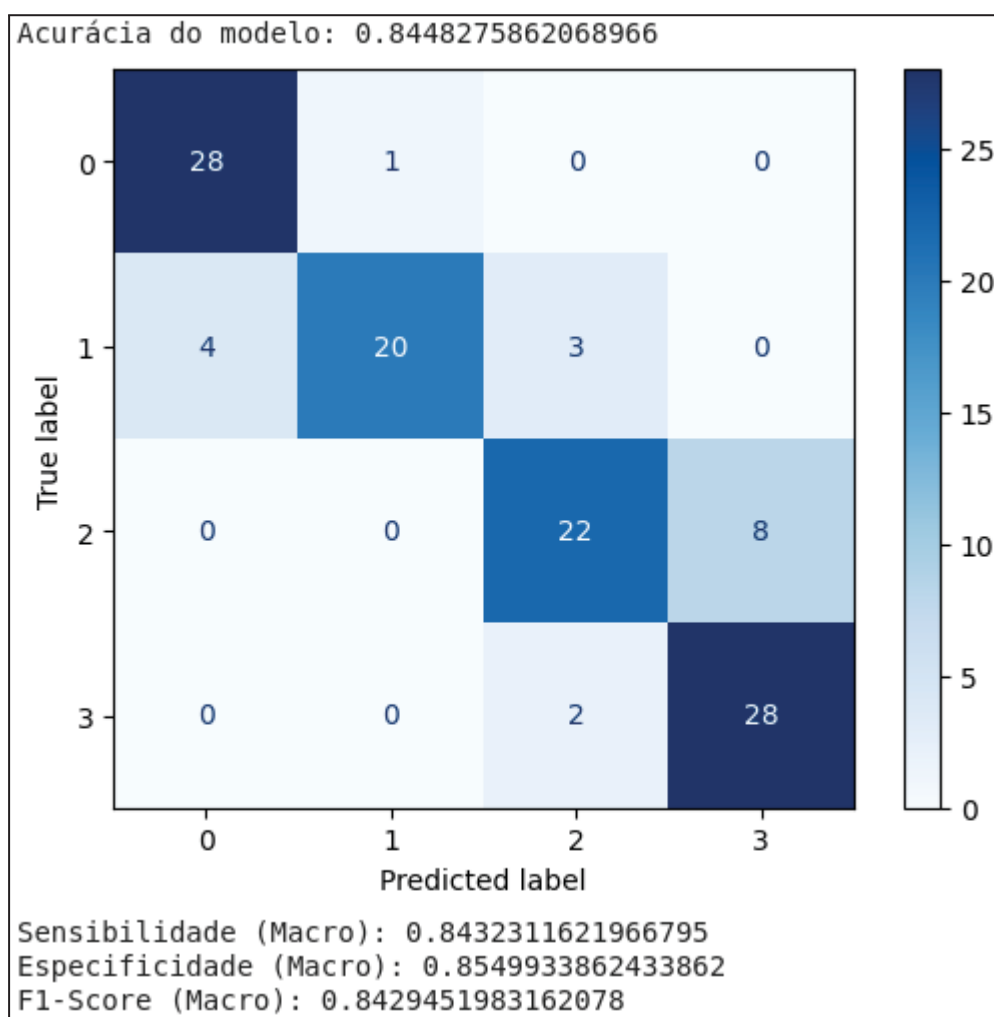
max_depth=17,
min_samples_split=8,
min_samples_leaf=5,
max_features='sqrt')

#rf_model = RandomForestClassifier()

#trainando o modelo
rf_model_cnn.fit(X_train, y_train)
# Avaliando o modelo para o tuning dos hiperparametros
y_pred = rf_model_cnn.predict(df_val_cnn.drop(['filename','category'],
axis=1))
resultado_tunning_rf_cnn = avaliar_resultados(df_val_cnn['category'],
y_pred)

```

FIGURA 55 - RANDOM FOREST - CNN.



SVM (LBP e CNN)

```
# LBP
from sklearn.svm import SVC

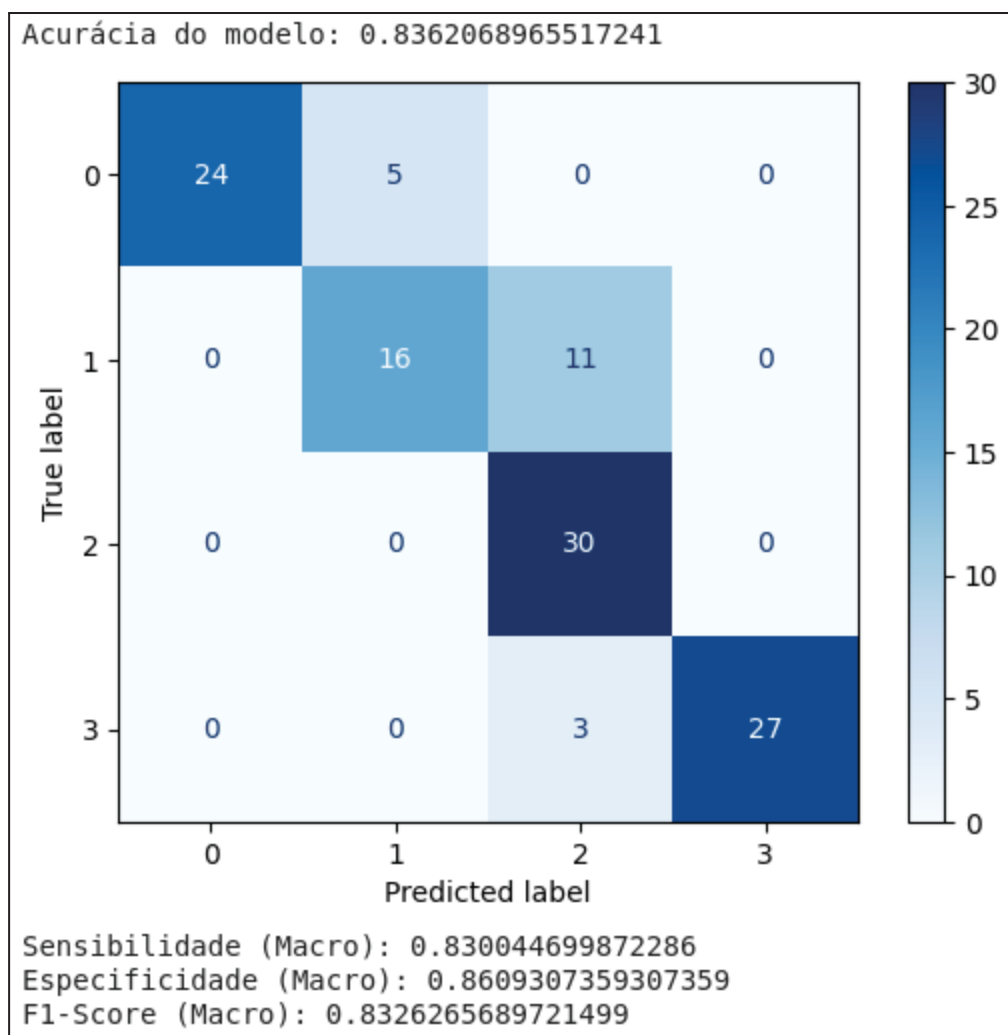
# Separate features (X) and target (y)
X_train = df_treino_lbp.drop(['filename', 'category'], axis=1) #
Caracteristicas
y_train = df_treino_lbp['category'] # alvo

#instanciando com hiperparametros
svm_model_lbp = SVC(kernel='rbf', C=700, gamma = 8,
decision_function_shape='ovo')

#treinando o modelo
svm_model_lbp.fit(X_train, y_train)

# Avaliando o modelo para o tuning dos hiperparametros
y_pred = svm_model_lbp.predict(df_val_lbp.drop(['filename', 'category'],
axis=1))
resultado_tunning_svm_lbp = avaliar_resultados(df_val_lbp['category'],
y_pred)
```

FIGURA 56 - SVM - LBP.



```
#CNN
# Separate features (X) and target (y)
X_train = df_treino_cnn.drop(['filename','category'], axis=1) #
Caracteristicas
y_train = df_treino_cnn['category'] # alvo

#instanciando com hiperparametros
svm_model_cnn = SVC(kernel='linear', C=280, gamma = 8,
decision_function_shape='ovr')

#treinando o modelo
svm_model_cnn.fit(X_train, y_train)

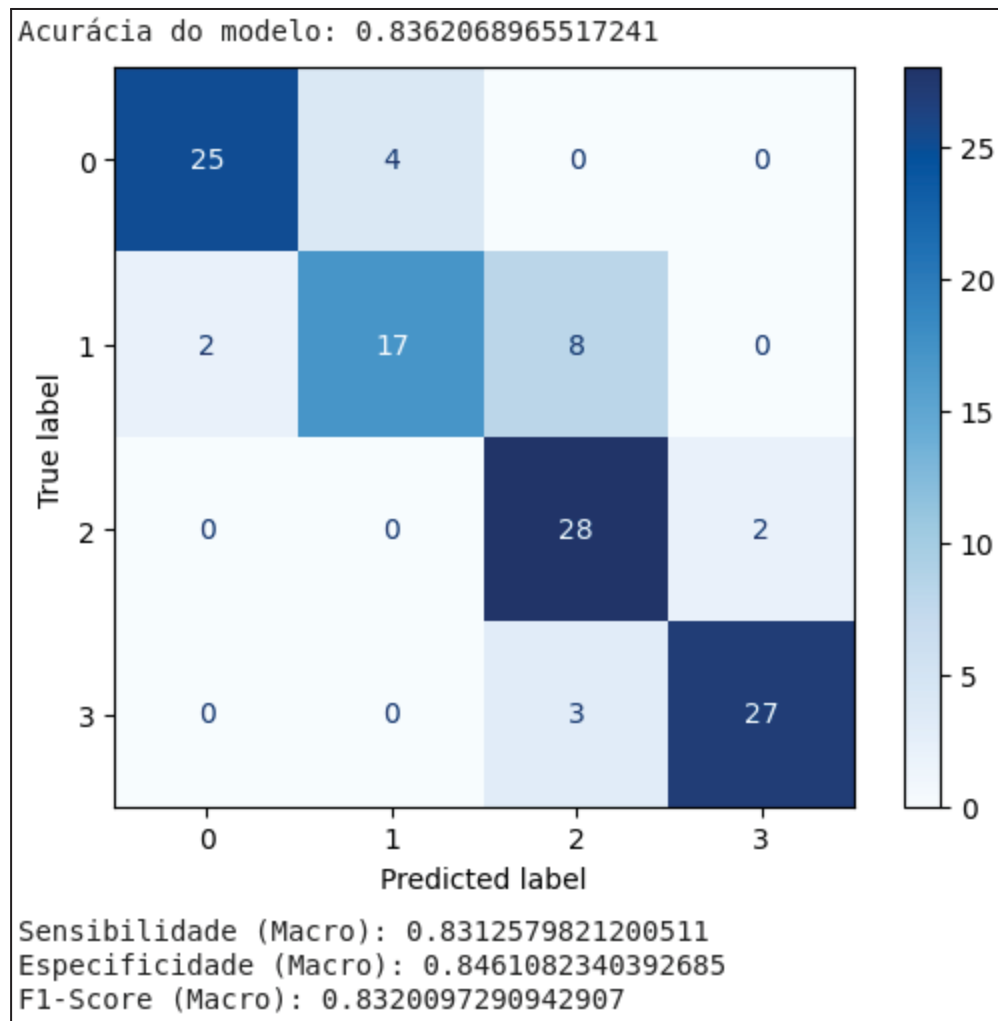
# Avaliando o modelo para o tuning dos hiperparametros
```

```

y_pred = svm_model_cnn.predict(df_val_cnn.drop(['filename', 'category'],
axis=1))
resultado_tunning_svm_cnn = avaliar_resultados(df_val_cnn['category'],
y_pred)

```

FIGURA 57 - SVM - CNN.



RNA (LBP e CNN)

```
#LBP
from sklearn.neural_network import MLPClassifier

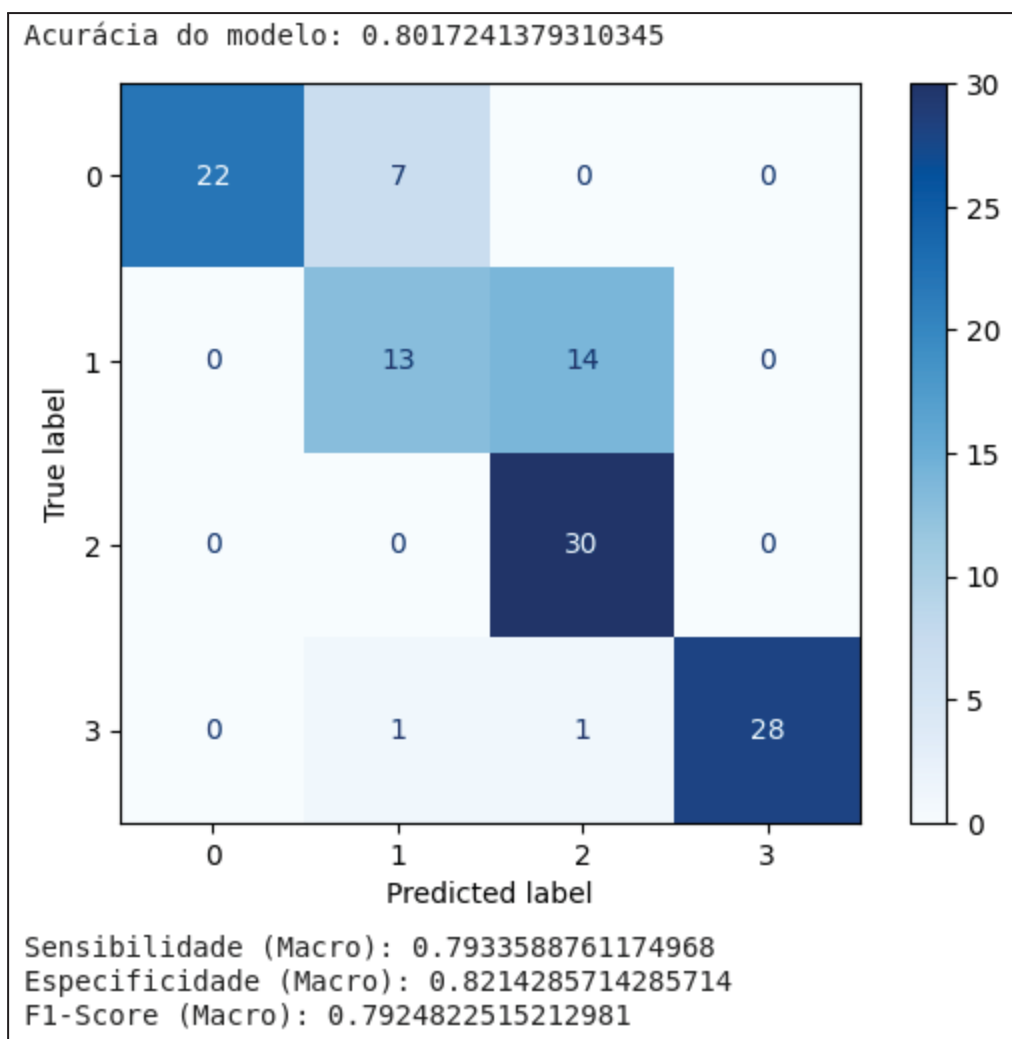
# Separate features (X) and target (y) - treinamento
X_train = df_treino_lbp.drop(['filename','category'], axis=1) #
Caracteristicas
y_train = df_treino_lbp['category']      # alvo

#instanciando com hiperparametros
rna_model_lbp = MLPClassifier(hidden_layer_sizes=(100,50),
                              learning_rate='constant',
                              learning_rate_init=0.001,
                              activation='relu',
                              solver='adam',
                              alpha=0.0001,
                              batch_size='auto',
                              max_iter=5000
                              )

#treinando modelo
rna_model_lbp.fit(X_train, y_train)

# Avaliando o modelo para o tuning dos hiperparametros
y_pred = rna_model_lbp.predict(df_val_lbp.drop(['filename','category'],
axis=1))
resultado_tunning_rna_lbp = avaliar_resultados(df_val_lbp['category'],
y_pred)
```

FIGURA 58 - RNA - LBP.



```
# CNN
# Separate features (X) and target (y) - treinamento
X_train = df_treino_cnn.drop(['filename','category'], axis=1) #
Caracteristicas
y_train = df_treino_cnn['category'] # alvo

#instanciando com hiperparametros
rna_model_cnn = MLPClassifier(hidden_layer_sizes=(100,50),
                              learning_rate='constant',
                              learning_rate_init=0.001,
                              activation='relu',
                              solver='adam',
                              alpha=0.0001,
                              batch_size='auto',
                              max_iter=5000)
```

```

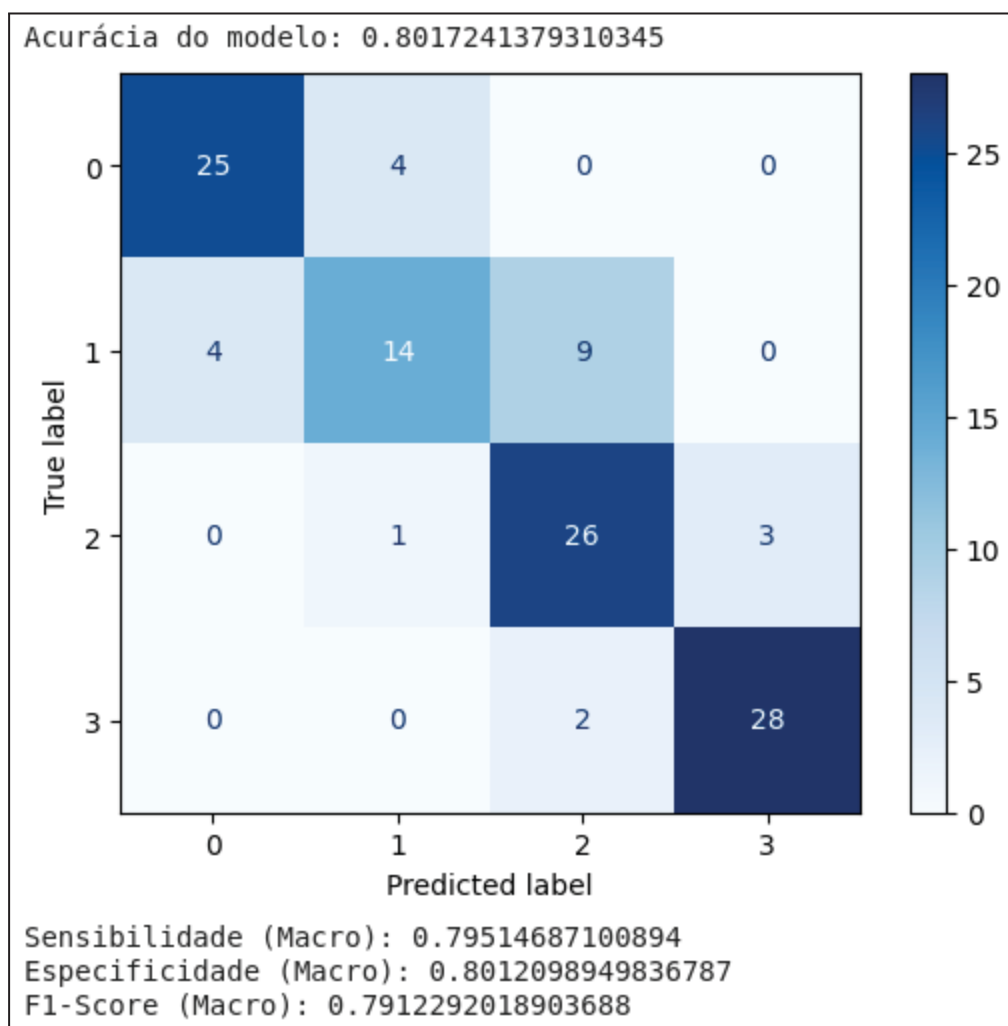
)

#treinando modelo
rna_model_cnn.fit(X_train, y_train)

# Avaliando o modelo para o tuning dos hiperparametros
y_pred = rna_model_cnn.predict(df_val_cnn.drop(['filename','category'],
axis=1))
resultado_tunning_rna_cnn = avaliar_resultados(df_val_cnn['category'],
y_pred)

```

FIGURA 59 - RNA - CNN.



```

# Carregamento da base de testes
test_images = []

for i in range(4):
    test_images += load_images_from_folder('Test_4cl_amostra/' + str(i) +
    '/', i)

#Extraí características utilizando LBP e salva em lista junto com a
categoria
data_test = []

for image, filename, category in test_images:
    features_lbp_test = extract_lbp_features(image)
    data_test.append(np.concatenate(([filename], [category],
    features_lbp_test)))

#cria um dataframe pandas e salva em csv
colunas = ['filename'] + ['category'] + [f'lbp_{i}' for i in
range(len(features_lbp_test))]
df_test_lbp = pd.DataFrame(data_test, columns=colunas)
df_test_lbp.to_csv('lbp_features_test.csv', index=False)

#Extraí características utilizando CNN e salva em lista junto com a
categoryegoria e filename
features_list_test = []

for image, filename, category in test_images:
    features_cnn_test = extract_features(image)
    features_list_test.append(np.concatenate(([filename], [category],
    features_cnn_test)))

# Create DataFrame and save to CSV
num_features = features_cnn_test.shape[0] # Get the number of features from
features_cnn
colunas_cnn = ['filename','category'] + [f'cnn_{i}' for i in
range(num_features)]
df_test_cnn = pd.DataFrame(features_list_test, columns=colunas_cnn)

df_test_cnn.to_csv('cnn_features_test.csv', index=False)

```

```

#faz as predições nas bases de teste
y_pred_rf_lbp =
rf_model_lbp.predict(df_test_lbp.drop(['filename','category'], axis=1))
y_pred_rf_cnn =
rf_model_cnn.predict(df_test_cnn.drop(['filename','category'], axis=1))

y_pred_svm_lbp =
svm_model_lbp.predict(df_test_lbp.drop(['filename','category'], axis=1))
y_pred_svm_cnn =
svm_model_cnn.predict(df_test_cnn.drop(['filename','category'], axis=1))

y_pred_rna_lbp =
rna_model_lbp.predict(df_test_lbp.drop(['filename','category'], axis=1))
y_pred_rna_cnn =
rna_model_cnn.predict(df_test_cnn.drop(['filename','category'], axis=1))

# avaliação dos resultados
print("Random Forest - LBP")
resultados_rf_lbp = avaliar_resultados(df_test_lbp['category'],
y_pred_rf_lbp)
print("\nRandom Forest - CNN")
resultados_rf_cnn = avaliar_resultados(df_test_cnn['category'],
y_pred_rf_cnn)

print("\nSVM - LBP")
resultados_svm_lbp = avaliar_resultados(df_test_lbp['category'],
y_pred_svm_lbp)
print("\nSVM - CNN")
resultados_svm_cnn = avaliar_resultados(df_test_cnn['category'],
y_pred_svm_cnn)

print("\nRNA - LBP")
resultados_rna_lbp = avaliar_resultados(df_test_lbp['category'],
y_pred_rna_lbp)
print("\nRNA - CNN")
resultados_rna_cnn = avaliar_resultados(df_test_cnn['category'],
y_pred_rna_cnn)

```

FIGURA 60 - RANDOM FOREST - LBP.

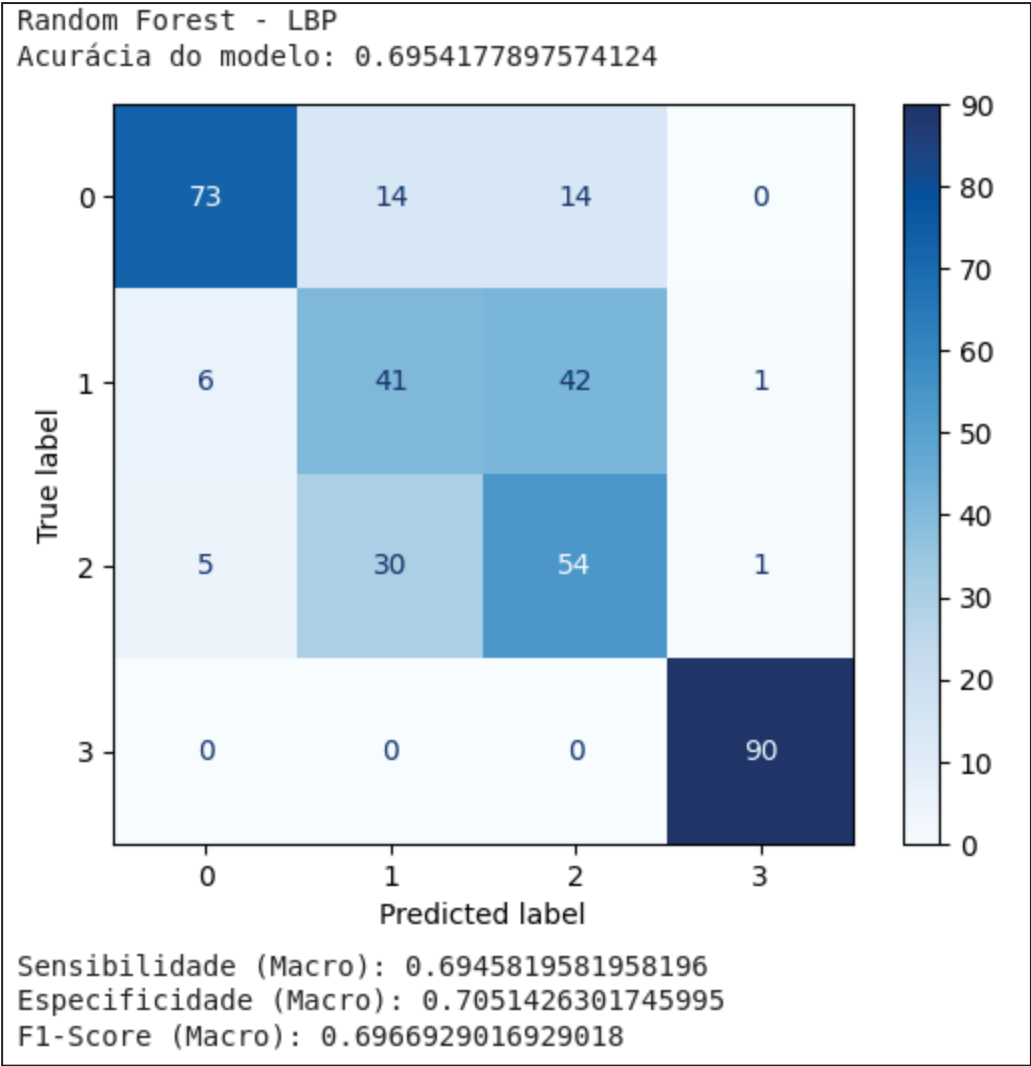


FIGURA 61 - RANDOM FOREST - CNN.

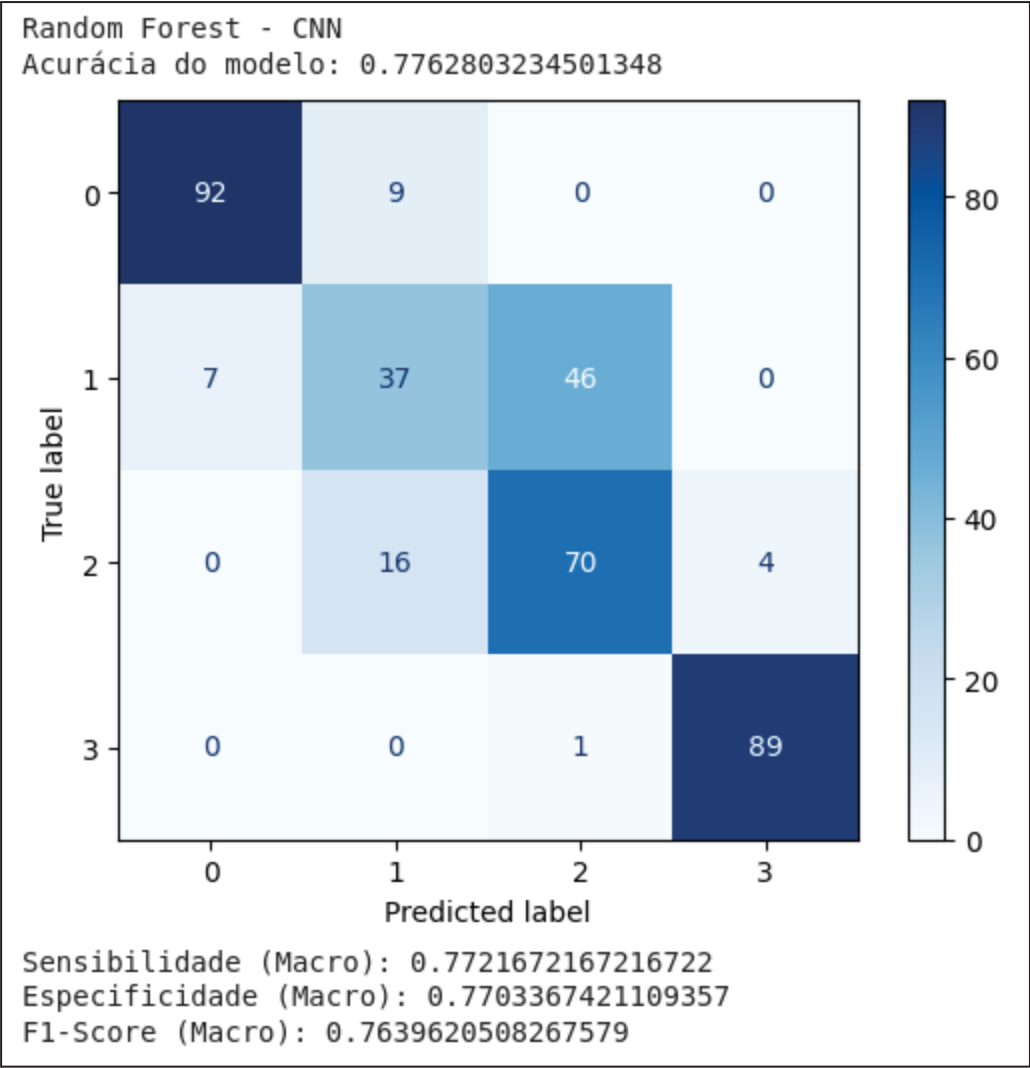


FIGURA 61 - SVM - LBP.

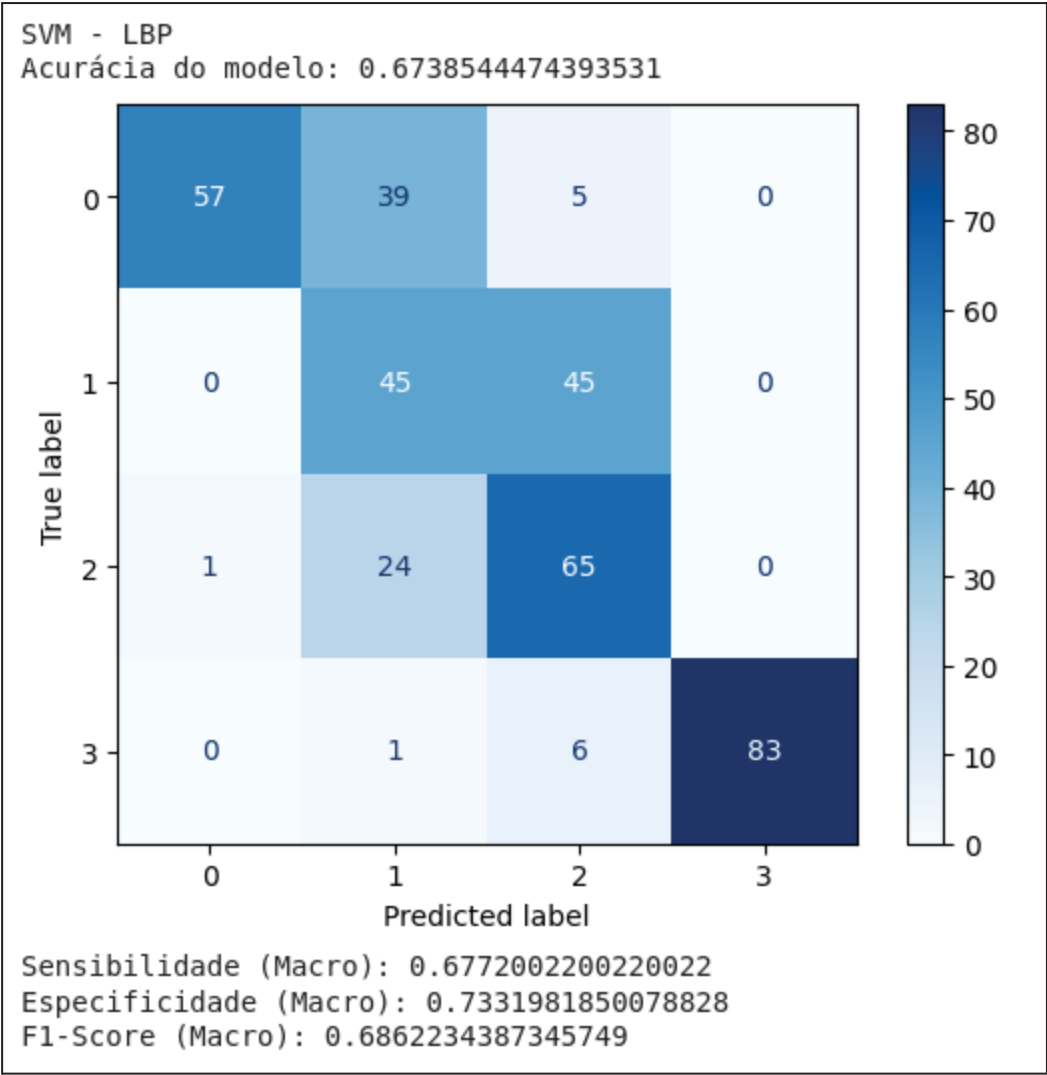


FIGURA 62 - SVM - CNN.

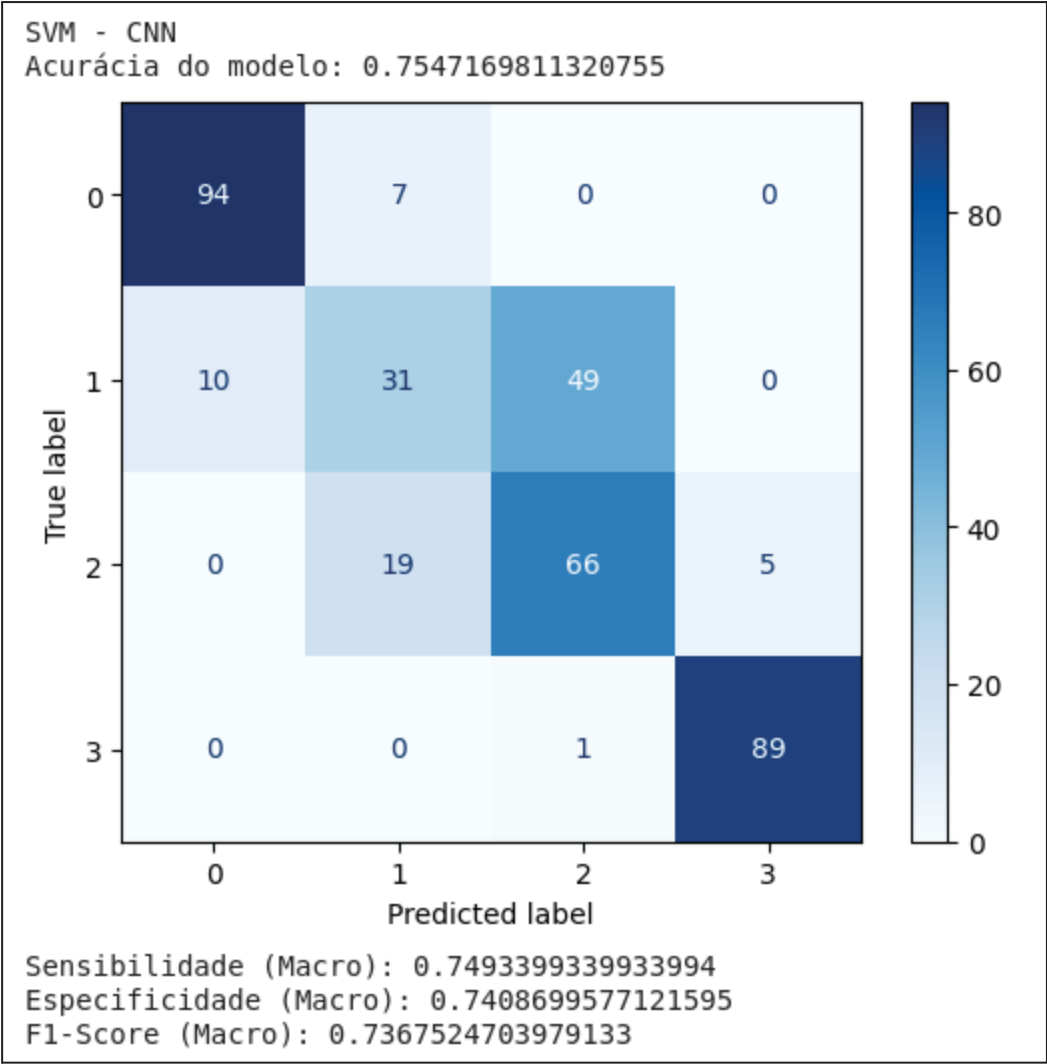


FIGURA 63 - LBA - LBP.

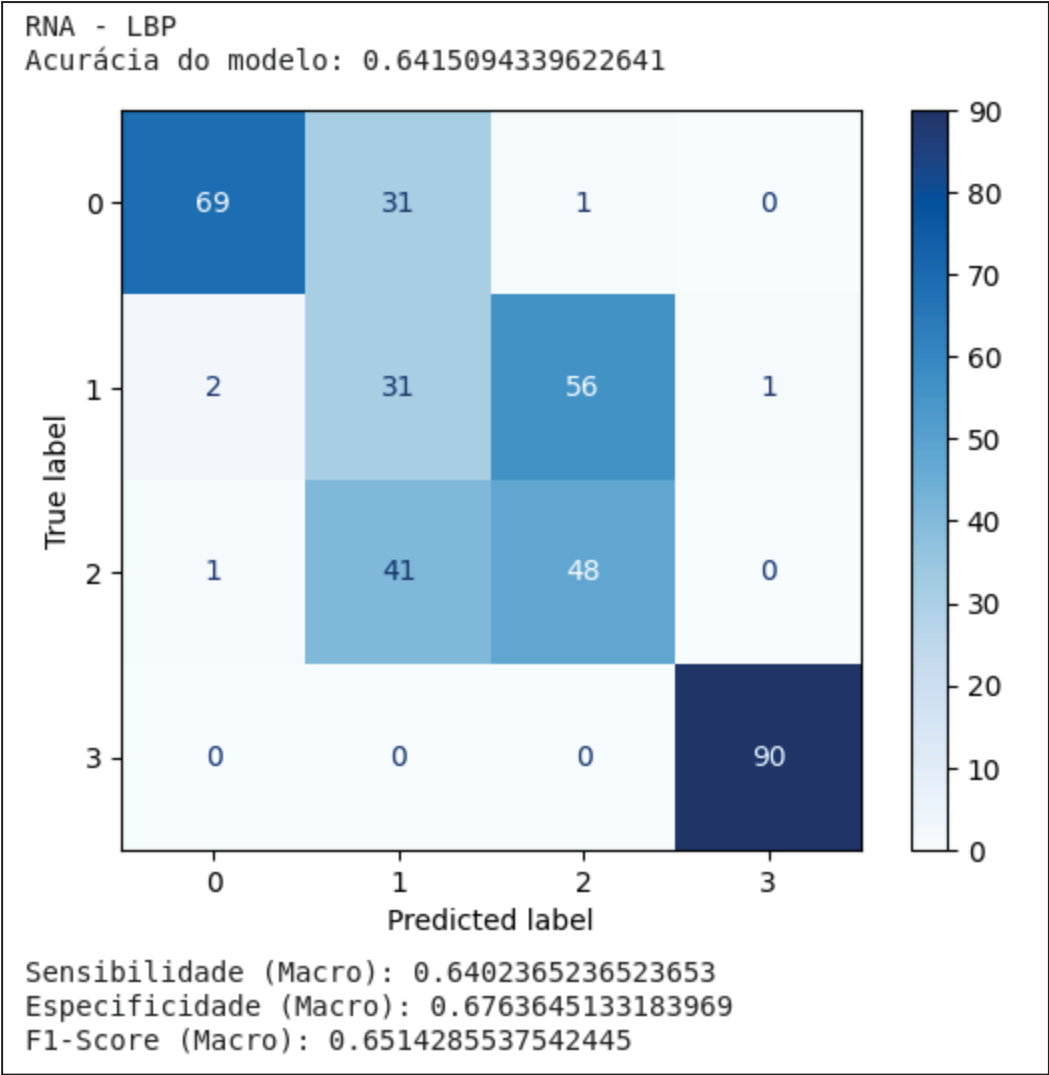
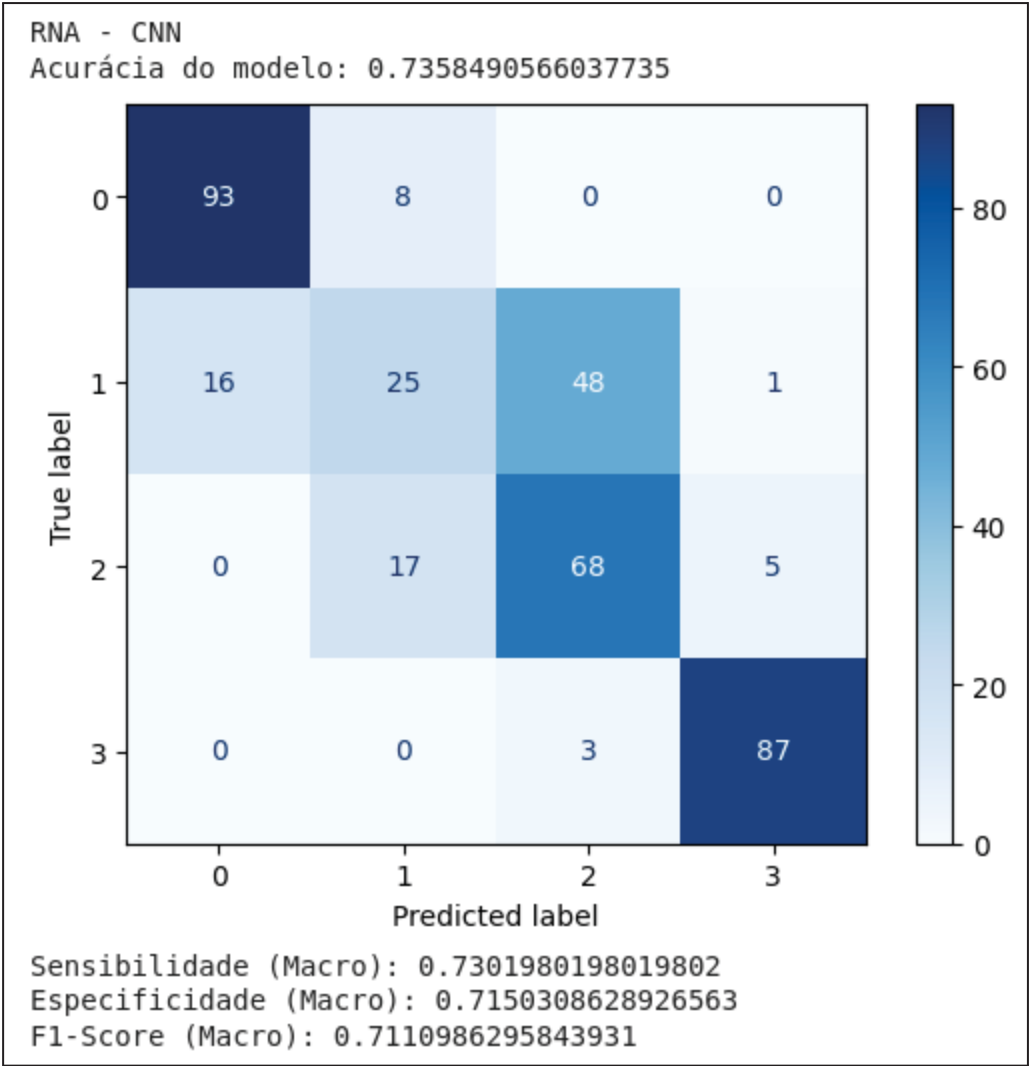


FIGURA 64 - LBA - CNN.



2 - Redes Neurais (Código)

```
##Mudar para essa versão do Tensorflow
%pip install tensorflow==2.15

from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras import layers, models
from keras.layers import Dense, Dropout, Flatten
from keras.models import Model

import cv2 as cv
import zipfile
import os

# Módulo para imprimir os gráficos de treinamento de forma dinâmica.
!pip install livelossplot

!wget /content/Train_Warwick.zip
!wget /content/Test_Warwick.zip

# Extrai o conteúdo do arquivo zip
!unzip Train_Warwick.zip
!unzip Test_Warwick.zip

def load_images_from_folder(folder):
    '''Carrega as imagens dos diretorios em uma lista com o nome do arquivo e
    classe'''
    images = []
    for filename in os.listdir(folder):
        img = cv.imread(os.path.join(folder,filename))
        if img is not None:
            images.append((img, filename, 1))
    return images

def resize_images(images):
```

```

'''Redimensiona as imagens para 224x224 como esperado por VGG16,
ResNet50'''
resized_images = []
for image, filename, i in images:
    resized_image = cv.resize(image, (224, 224))
    resized_images.append((resized_image, filename, i))
return resized_images

# carregando as imagens em 2 listas
# Move os dados do diretório images para images/train ou images/test:
import shutil
from collections import defaultdict
import json
from pathlib import Path
import os

os.makedirs('images/train', exist_ok=True)
os.makedirs('images/test', exist_ok=True)
os.makedirs('images/val', exist_ok=True)

temp_train_images = []
test_images_list = []

for i in range(4):
    temp_train_images.extend(load_images_from_folder('Train_4cls_amostra/' +
str(i) + '/'))

for i in range(4):
    test_images_list.extend(load_images_from_folder('Test_4cl_amostra/' +
str(i) + '/'))

''' separando em treino de validação, separando as imagens de 1 paciente
por
categoria (aprox. 30 imagens por paciente) temos ~20% da base de treino
Pacientes escolhidos: 01, 14, 04 e 06'''

resized_train_images = resize_images(temp_train_images)
resized_test_images = resize_images(test_images_list)

```

```

for img, filename, i in resized_train_images:
    os.makedirs('images/val/' + str(i) + '/', exist_ok=True)
    os.makedirs('images/train/' + str(i) + '/', exist_ok=True)
    if filename.startswith('01') or filename.startswith('14') or
filename.startswith('04') or filename.startswith('06'):
        cv.imwrite(os.path.join('images/val/' + str(i) + '/', filename), img)
    else:
        cv.imwrite(os.path.join('images/train/' + str(i) + '/', filename),
img)

for img, filename, i in resized_test_images:
    os.makedirs('images/test/' + str(i) + '/', exist_ok=True)
    cv.imwrite(os.path.join('images/test/' + str(i) + '/', filename), img)

import os

class_subset = sorted(os.listdir('Train_4cls_amostra/'))
print(class_subset)

# carregando as redes com transfer learning sem as camadas fully connected

base_model_vgg16 = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
base_model_vgg16_noaug = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
base_model_resnet50 = ResNet50(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
base_model_resnet50_noaug = ResNet50(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

for layer in base_model_vgg16.layers:
    layer.trainable = False

for layer in base_model_resnet50.layers:
    layer.trainable = False

for layer in base_model_vgg16_noaug.layers:
    layer.trainable = False

for layer in base_model_resnet50_noaug.layers:

```

```

layer.trainable = False

# Adicionar camadas fully connected no VGG16

# camadas próprias - você pode colocar mais se quiser
# A saída da resnet será a entrada da camada criada
x = Flatten()(base_model_vgg16.output)

# camada de classificação com as 4 classes utilizadas
prediction = Dense(4, activation='softmax')(x)

# Criação do Objeto Modelo (a parte da resnet + as camadas Fully connected
criadas)
model_vgg16 = Model(inputs=base_model_vgg16.input, outputs=prediction)

# Adicionar camadas fully connected no VGG16 sem Data Augmentation

# camadas próprias - você pode colocar mais se quiser
# A saída da resnet será a entrada da camada criada
x_noaug = Flatten()(base_model_vgg16_noaug.output)

# camada de classificação com as 4 classes utilizadas
prediction_noaug = Dense(4, activation='softmax')(x_noaug)

# Criação do Objeto Modelo (a parte da resnet + as camadas Fully connected
criadas)
model_vgg16_noaug = Model(inputs=base_model_vgg16_noaug.input,
outputs=prediction_noaug)

# Adicionar camadas fully connected no resnet50

# camadas próprias - você pode colocar mais se quiser
# A saída da resnet será a entrada da camada criada
x_res = Flatten()(base_model_resnet50.output)

# camada de classificação com as 4 classes utilizadas
prediction_res = Dense(4, activation='softmax')(x_res)

# Criação do Objeto Modelo (a parte da resnet + as camadas Fully connected
criadas)

```

```

model_resnet50 = Model(inputs=base_model_resnet50.input,
outputs=prediction_res)

# Adicionar camadas fully connected no resnet50 sem data augmentation

# camadas próprias - você pode colocar mais se quiser
# A saída da resnet será a entrada da camada criada
x_res_noaug = Flatten()(base_model_resnet50_noaug.output)

# camada de classificação com as 4 classes utilizadas
prediction_res_noaug = Dense(4, activation='softmax')(x_res_noaug)

# Criação do Objeto Modelo (a parte da resnet + as camadas Fully connected
criadas)
model_resnet50_noaug = Model(inputs=base_model_resnet50_noaug.input,
outputs=prediction_res_noaug)

from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_generator_resnet = ImageDataGenerator(
    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)

BATCH_SIZE = 64 # quantidade de imagens criadas em cada ciclo

traingen_resnet =
train_generator_resnet.flow_from_directory('images/train',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=class_subset,
    #subset='training',

```

```

        shuffle=True,
        seed=42)

from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_generator_vgg = ImageDataGenerator(
    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)

traingen_vgg = train_generator_vgg.flow_from_directory('images/train',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=class_subset,
    shuffle=True,
    seed=42)

import tensorflow as tf
train_noaug = tf.keras.utils.image_dataset_from_directory('images/train',
    labels='inferred',
    label_mode='categorical',
    class_names=class_subset,
    image_size=(224, 224),
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=42)

validation_noaug =
tf.keras.utils.image_dataset_from_directory('images/val',
    labels='inferred',
    label_mode='categorical',
    class_names=class_subset,
    image_size=(224, 224),

```

```

        batch_size=BATCH_SIZE,
        shuffle=True,
        seed=42)

test_noaug = tf.keras.utils.image_dataset_from_directory('images/test',
        labels='inferred',
        label_mode='categorical',
        class_names=class_subset,
        image_size=(224, 224),
        batch_size=BATCH_SIZE,
        shuffle=False,
        seed=42)

model_vgg16.summary()
model_vgg16_noaug.summary()

model_resnet50.summary()
model_resnet50_noaug.summary()

%%time
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

#steps_per_epoch = traingen_vgg.samples // BATCH_SIZE
#val_steps = validation_noaug.samples // BATCH_SIZE

n_epochs = 15

optimizer = RMSprop(learning_rate=0.001)

model_vgg16.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer =
ModelCheckpoint(filepath='img_model_vgg16.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

```

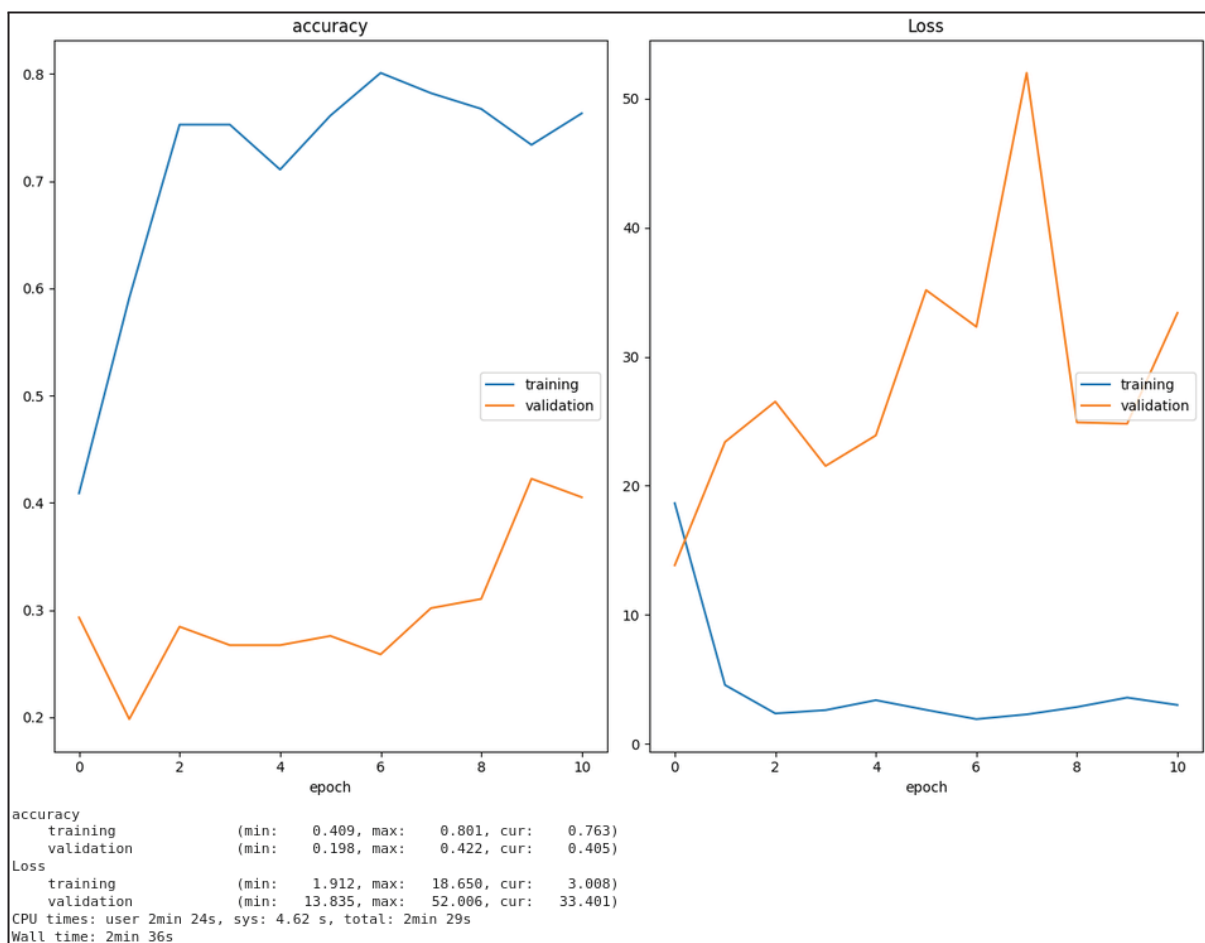
```

early_stop = EarlyStopping(monitor='val_loss',
                            patience=10,
                            restore_best_weights=True,
                            mode='min')

# Treinamento do Modelo
history_tl = model_vgg16.fit(training_vgg,
                             epochs=n_epochs,
                             steps_per_epoch=steps_per_epoch,
                             validation_data=validation_noaug,
                             validation_steps=val_steps,
                             callbacks=[checkpointer, PlotLossesKeras()],
                             callbacks=[early_stop, checkpointer, PlotLossesKeras()],
                             verbose=False)

```

FIGURA 65 - TREINAMENTO VGG16 COM DATA AUGMENTATION.



```
#função para avaliação de resultados
```

```

def avaliar_resultados(y_real, y_pred):
    # Avaliando a acurácia
    accuracy = accuracy_score(y_real, y_pred)
    print(f"Acurácia do modelo: {accuracy}")

    # Exibindo a matriz de confusão
    cm = confusion_matrix(y_real, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(cmap=plt.cm.Blues)
    plt.show()

    # Calculando as métricas por classe e a média macro
    report = classification_report(y_real, y_pred, output_dict=True)
    sensibilidade_macro = report['macro avg']['recall'] # Sensibilidade
    (Recall) macro
    especificidade_macro = report['macro avg']['precision'] #
    Especificidade (Precision) macro
    f1_score_macro = report['macro avg']['f1-score'] # F1-Score macro

    print(f"Sensibilidade (Macro): {sensibilidade_macro}")
    print(f"Especificidade (Macro): {especificidade_macro}")
    print(f"F1-Score (Macro): {f1_score_macro}")

    return {
        "acuracia": accuracy,
        "sensibilidade_macro": sensibilidade_macro,
        "especificidade_macro": especificidade_macro,
        "f1_score_macro": f1_score_macro
    }

##Aplicar Modelo treinado na base de teste
### Calcular Métrica de Sensibilidade,, Especificidade e F1-Score com base
na matriz de confusão
from sklearn.metrics import accuracy_score, ConfusionMatrixDisplay
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix

```

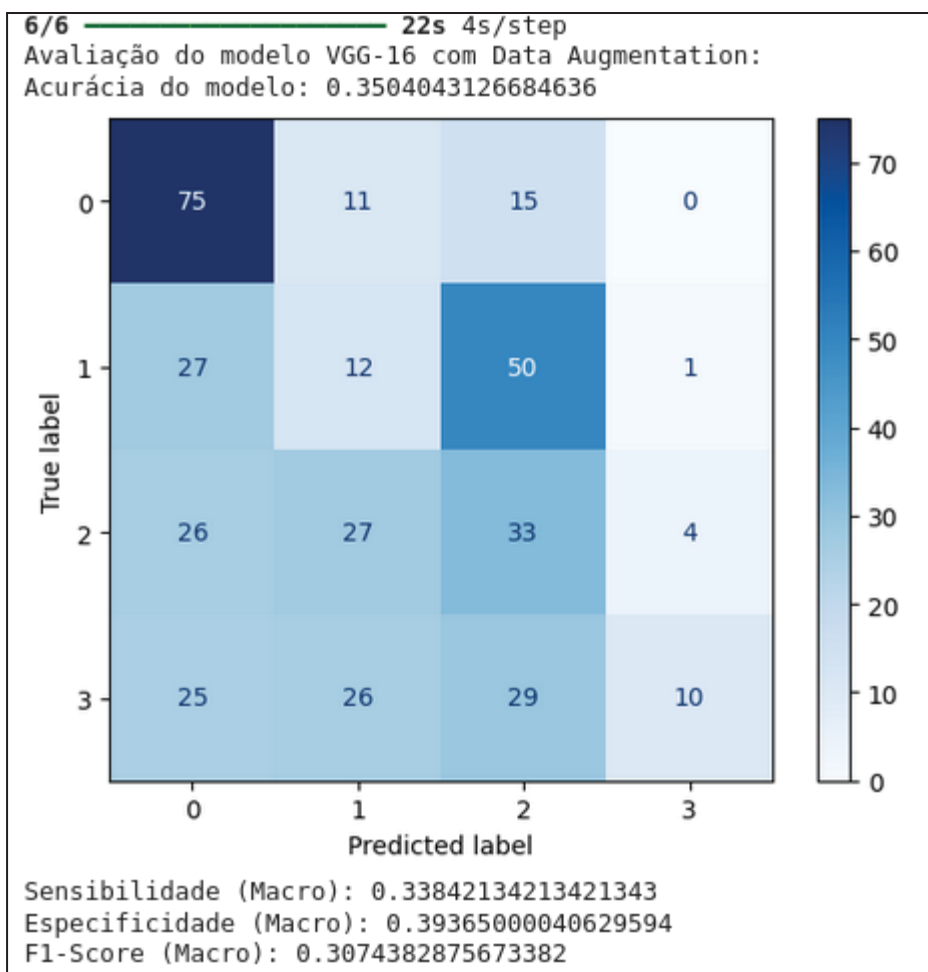
```
model_vgg16.load_weights('img_model_vgg16.weights.best.keras') ##
Inicializa o peso com o melhor treino

true_classes = np.argmax(np.concatenate([y for x, y in test_noaug],
axis=0), axis=1)
class_indices = traingen_vgg.class_indices
class_indices = dict((v,k) for k,v in class_indices.items())
predictions = model_vgg16.predict(test_noaug)
predicted_classes = np.argmax(predictions, axis=1)

print("Avaliação do modelo VGG-16 com Data Augmentation:")

avaliacao_vgg16 = avaliar_resultados(true_classes, predicted_classes)
```

FIGURA 66 - VGG-16 COM DATA AUGMENTATION - MÉTRICA DE SENSIBILIDADE, ESPECIFICIDADE E F1-SCORE.



```
%%time
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

#steps_per_epoch = train_noaug.samples // BATCH_SIZE
#val_steps = validation_noaug.samples // BATCH_SIZE

n_epochs = 15

optimizer = RMSprop(learning_rate=0.001)

model_vgg16_noaug.compile(loss='categorical_crossentropy',
optimizer=optimizer, metrics=['accuracy'])
```

```

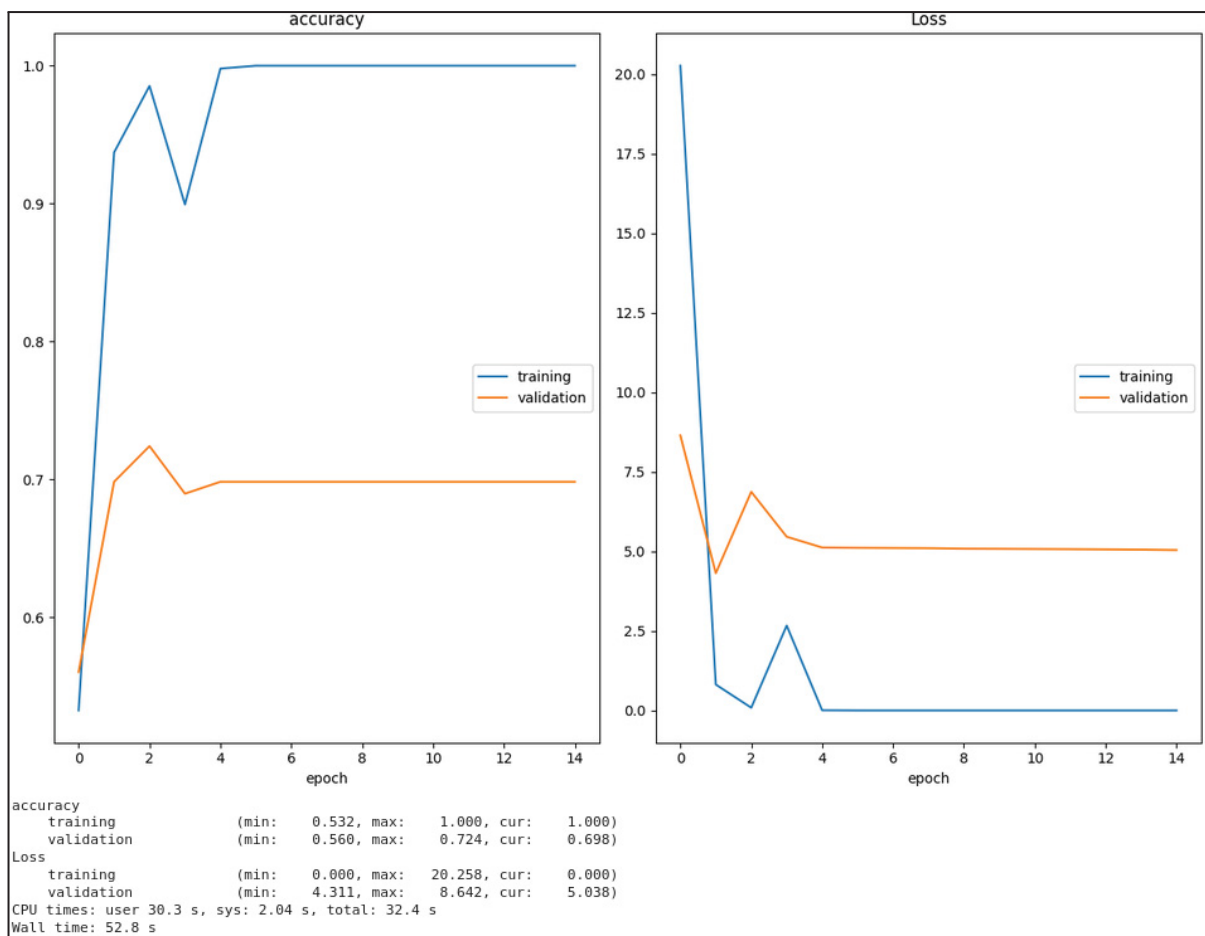
# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer =
ModelCheckpoint(filepath='img_model_vgg16_noaug.weights.best.keras',
                verbose=1,
                save_best_only=True)

# 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_t1 = model_vgg16_noaug.fit(train_noaug,
                                   epochs=n_epochs,
                                   #steps_per_epoch=steps_per_epoch,
                                   validation_data=validation_noaug,
                                   #validation_steps=val_steps,
                                   callbacks=[checkpointer, PlotLossesKeras()],
                                   #callbacks=[early_stop, checkpointer,
PlotLossesKeras()],
                                   verbose=False)

```

FIGURA 67 - TREINAMENTO VGG16 SEM DATA AUGMENTATION.



```
##Aplicar Modelo treinado na base de teste
### Calcular Métrica de Sensibilidade,, Especificidade e F1-Score com base
na matriz de confusão
from sklearn.metrics import accuracy_score
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix

model_vgg16_noaug.load_weights('img_model_vgg16_noaug.weights.best.keras')
## Inicializa o peso com o melhor treino

true_classes = np.argmax(np.concatenate([y for x, y in test_noaug],
axis=0), axis=1)
class_indices = {0: '0', 1: '1', 2: '2', 3: '3'}
```

```

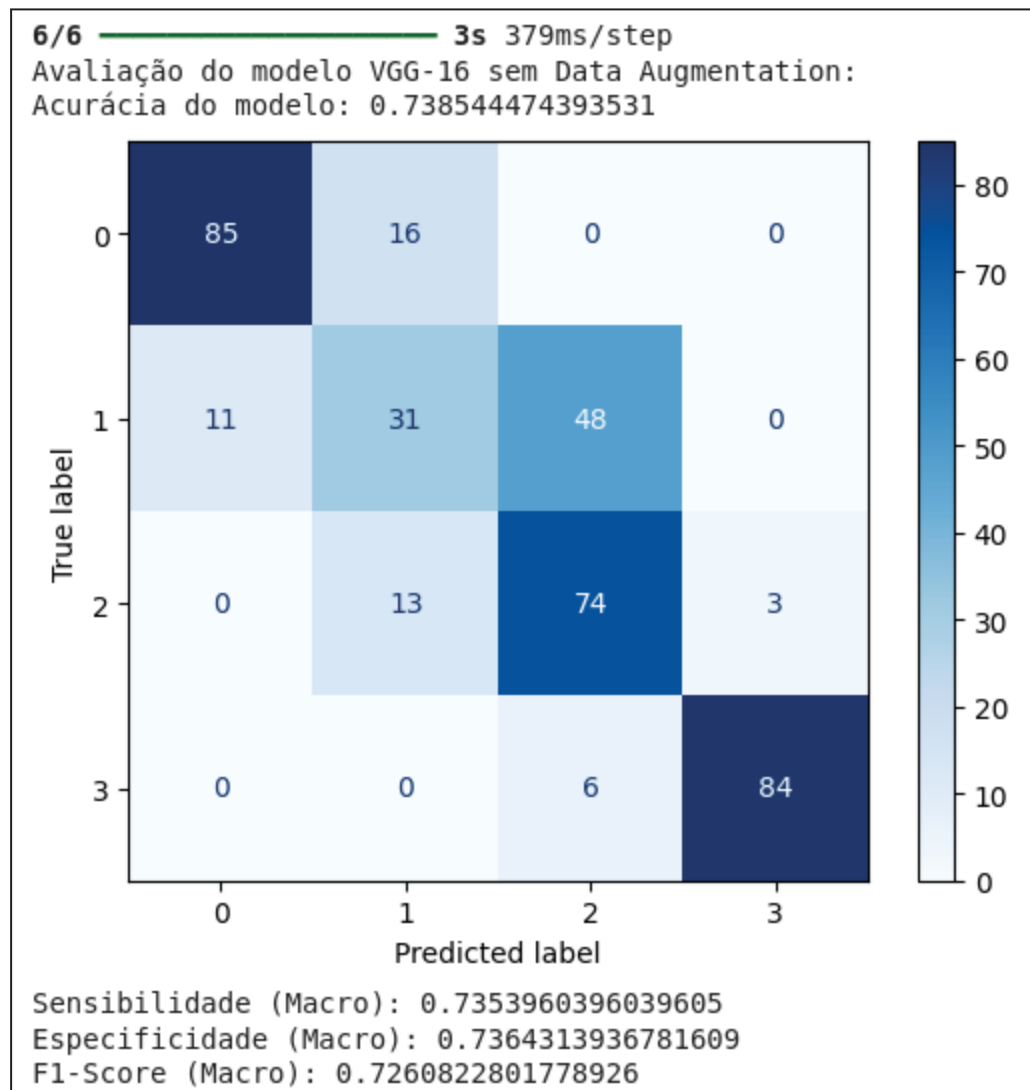
predictions = model_vgg16_noaug.predict(test_noaug)
predicted_classes = np.argmax(predictions, axis=1)

print("Avaliação do modelo VGG-16 sem Data Augmentation:")

avaliacao_vgg16_noaug = avaliar_resultados(true_classes, predicted_classes)

```

FIGURA 68 - VGG-16 SEM DATA AUGMENTATION - MÉTRICA DE SENSIBILIDADE, ESPECIFICIDADE E F1-SCORE.



```

%%time
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

#steps_per_epoch = traingen_resnet.samples // BATCH_SIZE
#val_steps = validation_noaug.samples // BATCH_SIZE

n_epochs = 15

optimizer = RMSprop(learning_rate=0.001)

model_resnet50.compile(loss='categorical_crossentropy',
optimizer=optimizer, metrics=['accuracy'])

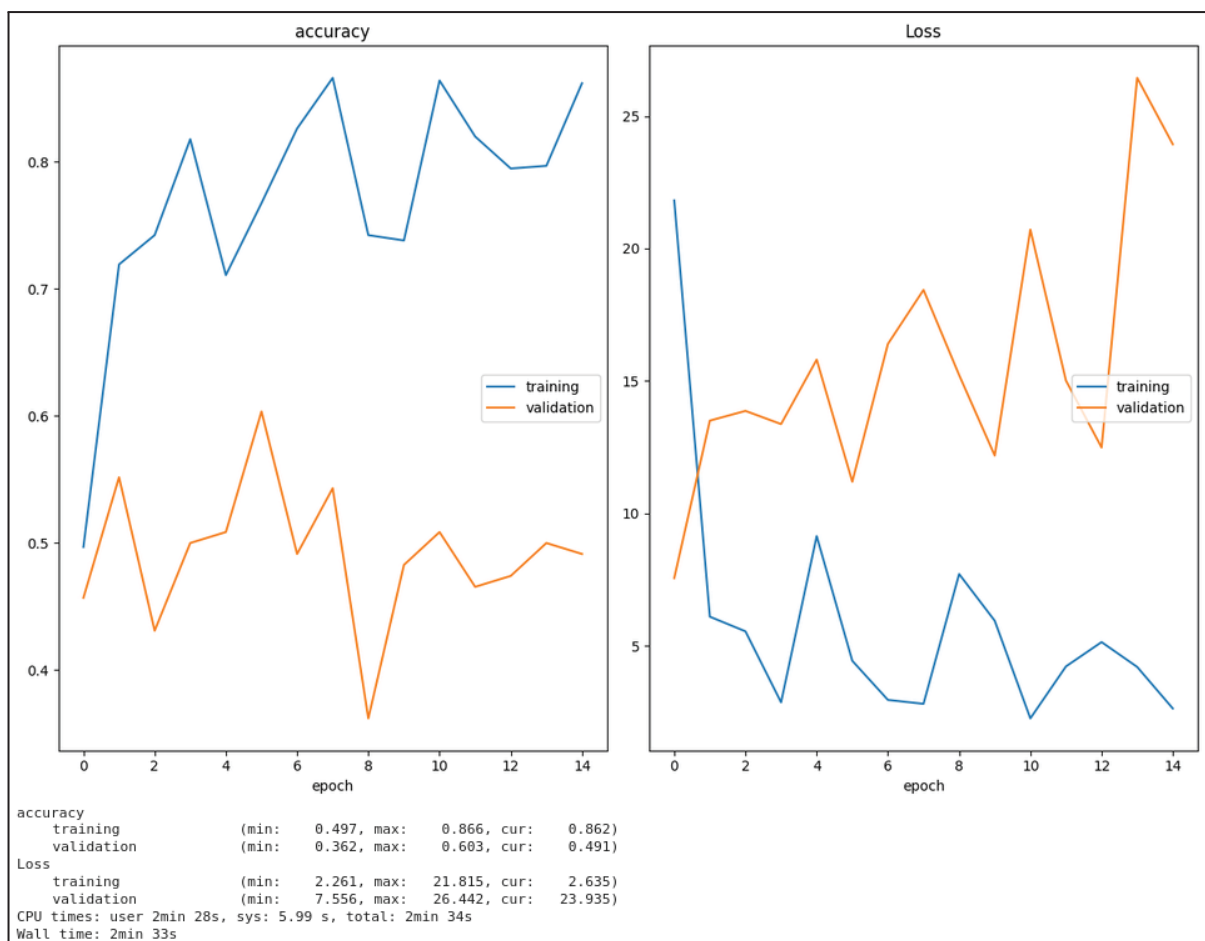
# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer =
ModelCheckpoint(filepath='img_model_resnet50.weights.best.keras',
                verbose=1,
                save_best_only=True)

# 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_t1 = model_resnet50.fit(traingen_resnet,
                               epochs=n_epochs,
                               #steps_per_epoch=steps_per_epoch,
                               validation_data=validation_noaug,
                               #validation_steps=val_steps,
                               callbacks=[checkpointer, PlotLossesKeras()],
                               #callbacks=[early_stop, checkpointer,
PlotLossesKeras()],
                               verbose=False)

```

FIGURA 69 - TREINAMENTO RESNET50 COM DATA AUGMENTATION.



```
##Aplicar Modelo treinado na base de teste
### Calcular Métrica de Sensibilidade,, Especificidade e F1-Score com base
na matriz de confusão
from sklearn.metrics import accuracy_score
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix

model_resnet50.load_weights('img_model_resnet50.weights.best.keras') ##
Inicializa o peso com o melhor treino

true_classes = np.argmax(np.concatenate([y for x, y in test_noaug],
axis=0), axis=1)
class_indices = {0: '0', 1: '1', 2: '2', 3: '3'}
```

```

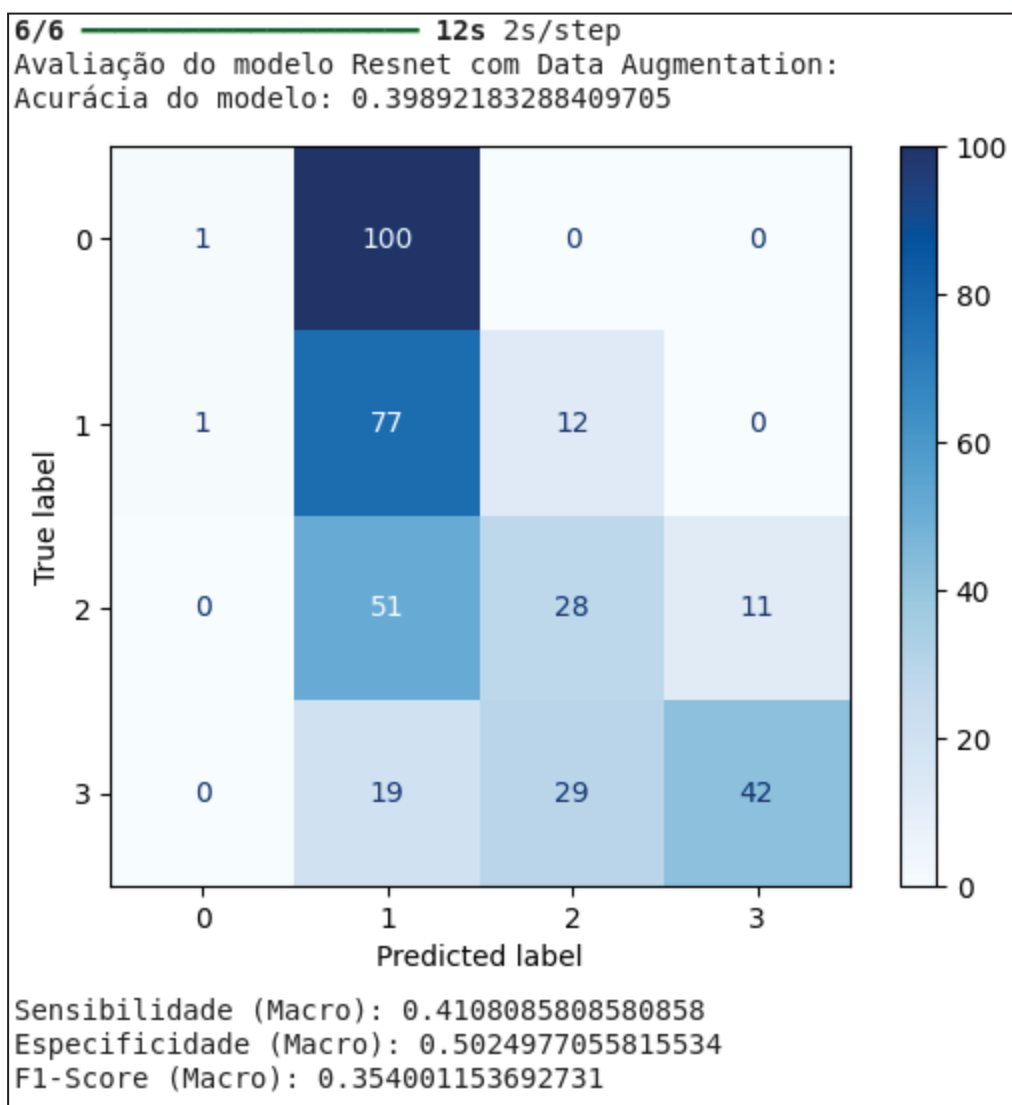
predictions = model_resnet50.predict(test_noaug)
predicted_classes = np.argmax(predictions, axis=1)

print("Avaliação do modelo Resnet com Data Augmentation:")

avaliacao_resnet = avaliar_resultados(true_classes, predicted_classes)

```

FIGURA 70 - RESNET50 COM DATA AUGMENTATION - MÉTRICA DE SENSIBILIDADE, ESPECIFICIDADE E F1-SCORE.



```

%%time
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

#steps_per_epoch = traingen.samples // BATCH_SIZE
#val_steps = validgen.samples // BATCH_SIZE

n_epochs = 15

optimizer = RMSprop(learning_rate=0.001)

model_resnet50_noaug.compile(loss='categorical_crossentropy',
optimizer=optimizer, metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer =
ModelCheckpoint(filepath='img_model_resnet50_noaug.weights.best.keras',
                verbose=1,
                save_best_only=True)

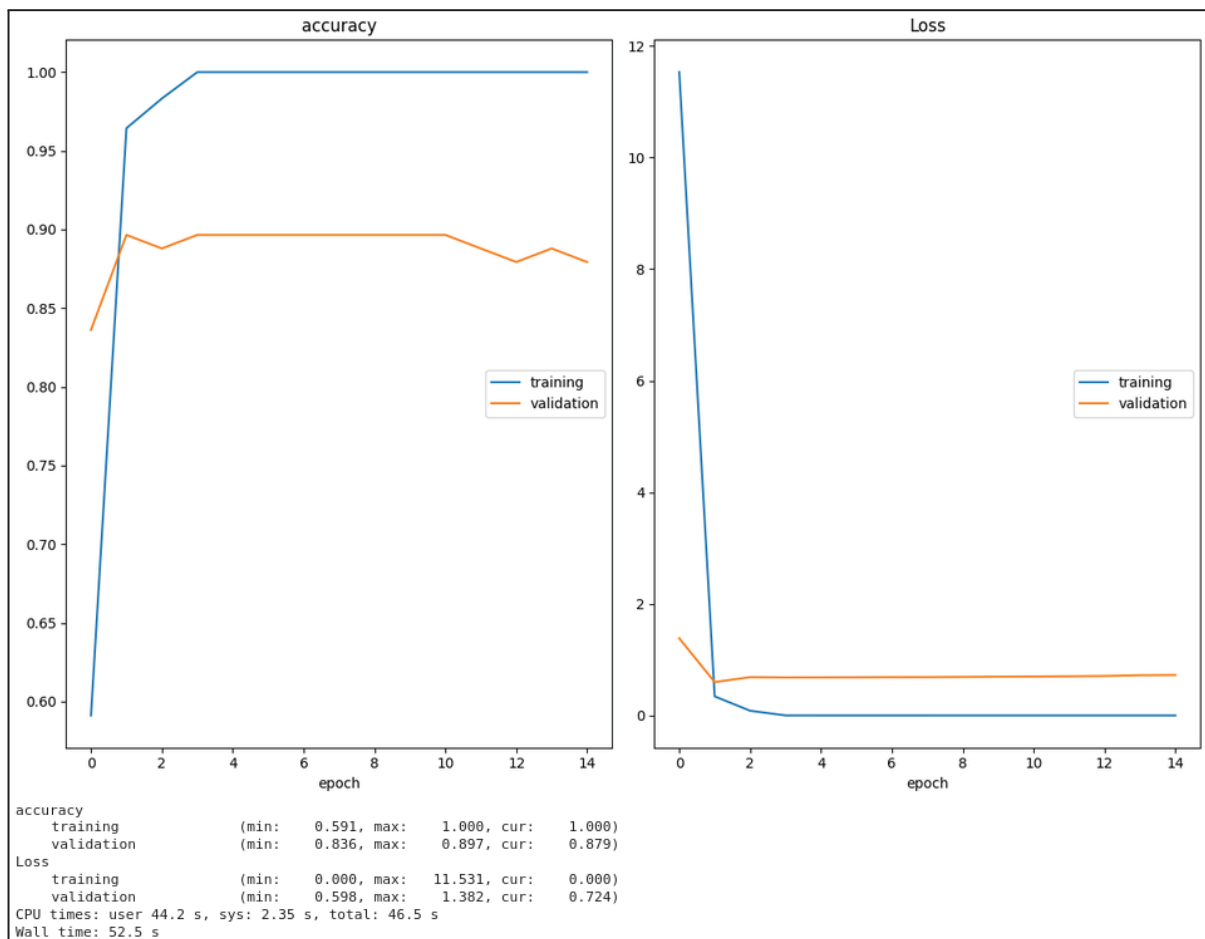
# Para o treinamento para prevenir o overfitting
# Não utilizei aqui, pois queria que rodasse todas as 30 épocas
early_stop = EarlyStopping(monitor='val_loss',
                           patience=10,
                           restore_best_weights=True,
                           mode='min')

# Treinamento do Modelo
history_t1 = model_resnet50_noaug.fit(train_noaug,
                                     epochs=n_epochs,
                                     #steps_per_epoch=steps_per_epoch,
                                     validation_data=validation_noaug,
                                     #validation_steps=val_steps,
                                     callbacks=[checkpointer, PlotLossesKeras()],
                                     #callbacks=[early_stop, checkpointer,
PlotLossesKeras()],

```

```
verbose=False)
```

FIGURA 71 - TREINAMENTO RESNET50 SEM DATA AUGMENTATION.



```
##Aplicar Modelo treinado na base de teste
### Calcular Métrica de Sensibilidade,, Especificidade e F1-Score com base
na matriz de confusão
from sklearn.metrics import accuracy_score
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix

model_resnet50_noaug.load_weights('img_model_resnet50_noaug.weights.best.ke
ras') ## Inicializa o peso com o melhor treino
```

```

true_classes = np.argmax(np.concatenate([y for x, y in test_noaug],
axis=0), axis=1)
class_indices = {0: '0', 1: '1', 2: '2', 3: '3'}

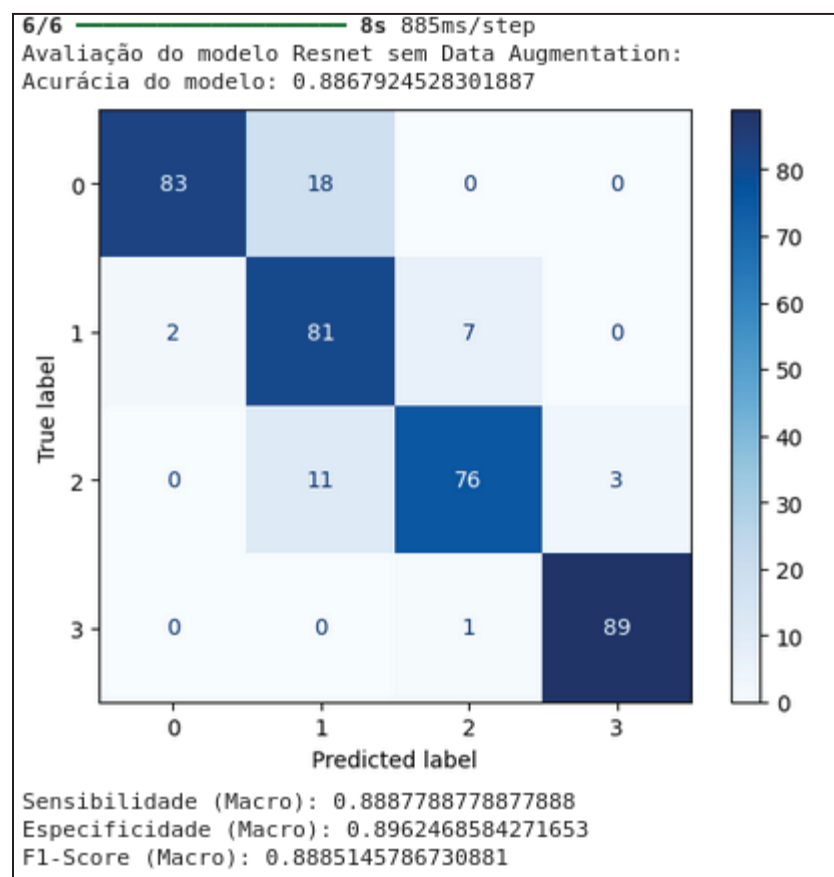
predictions = model_resnet50_noaug.predict(test_noaug)
predicted_classes = np.argmax(predictions, axis=1)

print("Avaliação do modelo Resnet sem Data Augmentation:")

avaliacao_resnet_noaug = avaliar_resultados(true_classes,
predicted_classes)

```

FIGURA 72 - RESNET50 SEM DATA AUGMENTATION - MÉTRICA DE SENSIBILIDADE, ESPECIFICIDADE E F1-SCORE.



APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

Parâmetros para elaboração do Trabalho:

- 1. Relevância Ética:** O trabalho deve abordar questões éticas significativas relacionadas ao uso da inteligência artificial, especialmente no contexto do ChatGPT. Os alunos devem identificar dilemas éticos relevantes e explorar como esses dilemas afetam diferentes partes interessadas, como usuários, desenvolvedores e a sociedade em geral.
- 2. Análise Crítica:** Os alunos devem realizar uma análise crítica das implicações éticas do uso do ChatGPT em estudos de caso específicos. Eles devem examinar como o algoritmo pode influenciar a disseminação de informações, a privacidade dos usuários e a tomada de decisões éticas. Além disso, devem considerar possíveis vieses algorítmicos, discriminação e questões de responsabilidade.
- 3. Soluções Responsáveis:** Além de identificar os desafios éticos, os alunos devem propor soluções responsáveis e éticas para lidar com esses dilemas. Isso pode incluir sugestões para políticas, regulamentações ou práticas de design que promovam o uso responsável da inteligência artificial. Eles devem considerar como essas soluções podem equilibrar os interesses de diferentes partes interessadas e promover valores éticos fundamentais, como transparência, justiça e privacidade.
- 4. Colaboração e Discussão:** O trabalho deve envolver discussões em grupo e colaboração entre os alunos. Eles devem compartilhar ideias, debater diferentes pontos de vista e chegar a conclusões informadas através do diálogo e da reflexão mútua. O estudo de caso do ChatGPT pode servir como um ponto de partida para essas discussões, incentivando os alunos a aplicar conceitos éticos e legais aprendidos ao analisar um caso concreto.
- 5. Limite de Palavras:** O trabalho terá um limite de 6 a 10 páginas teria aproximadamente entre 1500 e 3000 palavras.
- 6. Estruturação Adequada:** O trabalho siga uma estrutura adequada, incluindo introdução, desenvolvimento e conclusão. Cada seção deve ocupar uma parte proporcional do total de páginas, com a introdução e a conclusão ocupando menos espaço do que o desenvolvimento.

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

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

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

B – RESOLUÇÃO

RESUMO

Este trabalho aborda os diversos impactos e implicações éticas do uso do ChatGPT em diferentes contextos contemporâneos. Inicialmente, discute-se como a inteligência artificial está reconfigurando processos educacionais, substituindo métodos tradicionais de pesquisa e redação por ferramentas automatizadas, o que desafia os educadores a adaptarem suas metodologias de avaliação para garantir a autenticidade do aprendizado. Além da educação, são exploradas as potenciais melhorias no diagnóstico médico com a integração da IA, destacando-se a necessidade de aperfeiçoamento dos algoritmos e medidas robustas de segurança de dados para assegurar sua implementação ética. No setor jurídico, o artigo examina como o ChatGPT pode aumentar a produtividade na elaboração de documentos legais, embora enfrente desafios como a garantia de precisão e a proteção da privacidade. Por fim, no entretenimento, são levantadas questões sobre o uso ético de dados na criação de conteúdos gerados por IA, enfatizando a necessidade de regulamentação e compensação adequada para o trabalho produzido por essas ferramentas. Em síntese, a IA representa um avanço significativo com potencial transformador, desde que seu uso seja orientado por princípios éticos que preservem a integridade e responsabilidade humana.

Palavras-chave: ChatGPT. Inteligência artificial. Educação. Saúde. Sistema legal. Entretenimento. Ética. Adaptação educacional. Diagnóstico médico. Privacidade de dados.

ABSTRACT

This paper addresses the various impacts and ethical implications of using ChatGPT in different contemporary contexts. Initially, it discusses how artificial intelligence is reconfiguring educational processes, replacing traditional research and writing methods with automated tools, which challenges educators to adapt their assessment methodologies to ensure the authenticity of learning. In addition to education, the potential improvements in medical diagnosis with AI integration are

explored, highlighting the need for algorithm refinement and robust data security measures to ensure ethical implementation. In the legal sector, the article examines how ChatGPT can increase productivity in legal document drafting, although facing challenges such as ensuring accuracy and protecting privacy. Finally, in entertainment, questions are raised about the ethical use of data in creating AI-generated content, emphasizing the need for regulation and appropriate compensation for work produced by these tools. In summary, AI represents a significant advancement with transformative potential, provided its use is guided by ethical principles that preserve human integrity and responsibility.

Keywords: ChatGPT. Artificial intelligence. Education. Health. Legal system. Entertainment. Ethics. Educational adaptation. Medical diagnosis. Data privacy.

1 - INTRODUÇÃO

No cenário atual, é possível observar muitos trabalhos acadêmicos, notícias e artigos com a temática dos diversos usos do ChatGPT, muitos deles levantando questões relacionadas a mudanças fundamentais em processos de trabalho e aprendizado, suas vantagens, desvantagens e implicações éticas nas relações humano-máquina.

Na educação, por exemplo, o ChatGPT tem transformado a maneira como os alunos realizam pesquisas e produzem trabalhos, substituindo a necessidade de consultar diversas fontes e elaborar textos de forma independente. Isso apresenta um desafio para os educadores, que precisam encontrar maneiras diferentes de avaliar se os alunos realmente compreendem o conteúdo, como através de apresentações orais e escrita manual. A adaptação das metodologias de ensino e avaliação é crucial para garantir que o aprendizado não seja prejudicado pelo uso da IA.

Além de seus impactos na educação, este trabalho abordará as implicações éticas do uso do ChatGPT em contextos como Saúde, Sistema Legal e Entretenimento. Exploraremos como essa ferramenta pode ser integrada de maneira ética e responsável, contribuindo positivamente para essas áreas sem comprometer a autonomia e a integridade dos usuários.

1.1 - CONTEXTO 1: EDUCAÇÃO

Os impactos do ChatGPT na educação são significativos. Anteriormente, para fazer um trabalho escolar, os alunos precisavam realizar suas pesquisas no Google e, em seguida, ler e organizar a pesquisa em um texto. O professor se preocupava principalmente com o "copiar e colar" sem as devidas citações e referências. Mesmo sem dar os devidos créditos às fontes, o professor tinha certeza de que o aluno havia lido e desenvolvido o trabalho, garantindo que ele tivesse aprendido algo nesse esforço.

Com o surgimento do ChatGPT, esse trabalho de consultar diversas fontes para escrever deixa de ser realizado pelo aluno, dificultando assim, a avaliação do aprendizado pelo professor. O professor precisa garantir que o aluno aprendeu o conteúdo ensinado na matéria, e faz isso por meio de avaliações diversas, como provas escritas, trabalhos, seminários, etc. Todos os meios de

avaliação possuem falhas. Uma prova escrita, por exemplo, não garante que o aluno que tirou uma nota baixa não saiba o conteúdo; ele pode não ter entendido a questão ou estar em um dia ruim.

Com a chegada do ChatGPT, alguns tipos de avaliação precisarão ser adaptados para garantir que o aluno absorveu o conteúdo explicado. O ChatGPT é uma ferramenta e não deve ser banida da academia. Deve-se apenas instruir os alunos sobre como utilizá-la da melhor forma. Alguns professores têm adotado mais avaliações manuscritas como forma de driblar o uso do ChatGPT. Karine Karsten, professora doutoranda em Educação na UFPR, diz: “Com a chegada das inteligências artificiais como o ChatGPT, os trabalhos de pesquisa por si só não serão suficientes para avaliar o aluno. Acredito que será necessário complementar o trabalho escrito com uma apresentação oral do que foi feito ou outras formas de complementar as avaliações, como a escrita em sala de um resumo do que havia no trabalho. Assim, mesmo que o trabalho tenha sido desenvolvido utilizando IA, será possível avaliar se o aluno leu e entendeu o que escreveu. Portanto, se o aluno fez ou não o trabalho utilizando IA não seria relevante, desde que ele tenha se apropriado do conteúdo, das competências e habilidades que eram o objetivo ao se propor um trabalho.”

Ronaldo Lemos, professor da Schwarzman College, passou a exigir que seus alunos sempre utilizassem o ChatGPT para os trabalhos. No entanto, em vez de avaliar as respostas, ele passou a avaliar as perguntas (ou "prompts") feitas para a ferramenta.

Na área de humanas, para obter uma resposta mais detalhada, é necessário fazer perguntas detalhadas sobre o tema. Perguntas óbvias geram respostas óbvias. Fazer as perguntas certas está se tornando uma forma de programação, mais acessível a pessoas sem conhecimento técnico, prompts complexos são necessários se o que se espera são resultados complexos nas respostas do ChatGPT.

1.2 - CONTEXTO 2: SAÚDE

A IA no diagnóstico médico promete melhorar a precisão, eficiência e velocidade dos diagnósticos, beneficiando pacientes e profissionais de saúde. No entanto, enfrenta desafios como a necessidade de aperfeiçoamento dos algoritmos para melhor precisão, garantia da privacidade dos dados e superação de barreiras culturais para a utilização da ferramenta. A aceitação e uso ético da IA são cruciais, e, por isso, até a OMS sugere princípios para assegurar o uso responsável, promovendo autonomia, bem-estar, transparência e sustentabilidade da tecnologia. Esses princípios visam, principalmente, garantir que a IA não substitua completamente o julgamento humano, mas sim seja usada como uma ferramenta de auxílio nos diagnósticos. Eles asseguram que a IA beneficie a saúde pública de maneira segura e eficiente.

O uso da IA ainda é tímido em muitas instituições de saúde, porém, em alguns locais, já se utilizam algoritmos que facilitam o reconhecimento de padrões em imagens médicas, principalmente para exames como tomografias e ressonâncias, além de soluções para áreas administrativas.

Um dos desafios para utilizar eficazmente a IA é fornecer todos os dados necessários para que ela possa fazer qualquer tipo de diagnóstico ou sugestão. Muitos sistemas hospitalares são distribuídos e seus dados não se comunicam entre si. Portanto, criar uma forma centralizada de

coleta de dados dos sistemas e disponibilização para uma IA é essencial. Medidas de segurança e privacidade devem ser aplicadas no local centralizado para evitar quaisquer tipos de incidentes que possam vaziar dados de pacientes.

Assegurar a segurança dos sistemas que armazenam informações de pacientes envolve a implementação de diversas medidas essenciais. A criptografia transforma os dados em formatos ilegíveis sem a chave adequada, garantindo confidencialidade e integridade. A redundância cria cópias dos dados em vários locais para mitigar riscos de perda e permitir recuperação eficiente em incidentes. A conformidade com LGPD ou GDPR assegura que empresas sigam diretrizes rigorosas, fortalecendo a confiança dos clientes ao proteger a privacidade e os direitos individuais. A detecção e correção de brechas e vulnerabilidades identifica e resolve falhas de segurança, protegendo dados sensíveis e garantindo conformidade com regulamentações de proteção de dados.

Outro fator muito relevante a ser considerado é a possível desumanização da relação médico-paciente proporcionada pelo uso da IA. Equilibrar o uso desta ferramenta com a necessidade de manter a empatia e o cuidado humanizado na relação médico-paciente é de suma importância. A implementação bem-sucedida da IA na medicina deve sempre considerar o aspecto humano, garantindo que o paciente continue sendo o foco central do cuidado, evitando um formato excessivamente mecânico de diagnósticos.

Outro aspecto importante é que o uso da IA no contexto médico, sem uma auditoria ou verificação clara dos algoritmos, pode propagar os chamados vieses de algoritmos que podem gerar discriminações por etnia, gênero, idade, entre outras características. Dessa forma, a supervisão humana dos algoritmos deve fazer parte do processo de implementação e verificação periódica.

Também é relevante discutir a responsabilidade do médico quando obtém um diagnóstico utilizando uma ferramenta de IA. Juridicamente, o profissional tem a palavra final em qualquer decisão de diagnóstico e tratamento. Dessa forma, a argumentação de que uma IA tomou uma má decisão não exime o médico de responsabilização por danos causados a pacientes em determinados tratamentos ou pela falta deles.

1.3 - CONTEXTO 3: JUSTIÇA E SISTEMA LEGAL

Na área jurídica, o ChatGPT vem causando uma revolução na produtividade. A elaboração de documentos jurídicos de baixa e média complexidade e o gerenciamento de documentos permitem aos advogados coletar e analisar grandes quantidades de dados jurídicos com rapidez e precisão. Procedimentos que levavam horas ou dias, envolvendo várias pessoas, agora podem ser feitos em questão de minutos ou segundos por uma única pessoa que faz a solicitação correta. As respostas podem ser usadas para apoiar argumentos no tribunal, gerar aconselhamento jurídico econômico ou preparar para o julgamento.

No entanto, toda essa comodidade e praticidade cobra seu preço: o ChatGPT, como outras inteligências artificiais, é passível de “alucinações”, isto é, criar conteúdo incorreto ou indicar jurisprudências inexistentes. A mesma característica que permite respostas tão personalizadas tem a capacidade de fabricar fatos e produzir fontes sem ter acesso a dados suficientes.

Outro problema são os erros que ele comete ao transcrever os artigos das leis. Como essa solução é baseada em um grande conjunto de dados contidos no GPT-3, ela tende a reproduzir as informações e os vieses ali encontrados. Sua base de dados, na versão gratuita, vai até setembro de 2021, então todas as atualizações e mudanças na lei após essa data não foram incluídas, o que pode causar problemas.

Muitos juristas ainda se preocupam com a privacidade de seus dados e de seus clientes, pois as perguntas feitas para o ChatGPT são armazenadas e usadas em seu treinamento constante, o que pode ocasionar uma quebra de privacidade se algum viés resgatar os dados pessoais que o usuário passou sem perceber.

Para além da especulação que possamos fazer sobre os problemas que o ChatGPT pode causar no meio jurídico, já começam a aparecer os primeiros casos de mau uso dessa ferramenta.

Nos Estados Unidos, o advogado Steven Schwartz, com mais de 30 anos de experiência, fez sua pesquisa jurídica de casos anteriores usando o ChatGPT e se baseou em suas respostas para redigir sua petição sobre um caso contra uma companhia aérea. O problema é que o ChatGPT criou casos de jurisprudência que simplesmente não existiam. O juiz do caso, percebendo o problema, puniu o advogado com uma multa de 5 mil dólares e sanções.

No Brasil, um advogado foi multado em R\$ 2.400 depois de protocolar uma petição redigida pelo programa ChatGPT. Além disso, um juiz federal do TRF da 1ª região, que assinou sentença feita por meio de inteligência artificial, será investigado pelo CNJ. A razão para essa investigação reside no fato de a ferramenta ter criado jurisprudências do STJ que fundamentaram a decisão do magistrado.

1.4 - CONTEXTO 4: ENTRETENIMENTO

Primeiro, a implicação de que não é ético usar dados que, mesmo estando disponíveis online, não estejam disponíveis de maneira legal. Isso inclui dados oriundos de pirataria ou que estejam acessíveis apenas mediante compensação monetária. Utilizar tais dados sem a devida autorização ou pagamento pode infringir direitos autorais e éticos.

Segundo, surge a questão de quem deve ser compensado pelo que é produzido por meio do ChatGPT. Se uma tarefa cumprida pela IA atinge um alto nível de qualidade e pode substituir o trabalho humano, quem deveria receber a compensação financeira por esse trabalho? Devemos considerar o papel do desenvolvedor da IA, do usuário que fez o prompt e de qualquer outro colaborador humano no processo, inclusive, os autores do conteúdo usado no treinamento.

Terceiro, é necessário discutir sobre quem recai a responsabilidade pelo uso final do que é gerado pelo ChatGPT para fins ilegais e imorais. Se uma obra gerada pela IA é utilizada de maneira prejudicial, quem deve ser responsabilizado? O criador da IA, o usuário ou ambos?

Um caso real que ilustra esses dilemas é o do ganhador do prêmio de produção artística cuja obra “Théâtre D’opéra Spatial” foi o produto de um híbrido entre o que foi feito por uma IA e o que foi editado posteriormente pelo artista. Este, teve seu pedido de direitos autorais negado, destacando a complexidade de se definir autoria e direitos em criações híbridas entre humanos e IA.

2 - METODOLOGIA

O método utilizado neste trabalho foi a revisão de literatura de artigos disponíveis na internet fora do âmbito acadêmico, mas elaborados por profissionais dos diferentes contextos abordados. Esta abordagem foi escolhida para melhor demonstrar a percepção das implicações éticas pelos profissionais que, de alguma forma, se sentem prejudicados ou são a favor do uso da ferramenta nos processos de trabalho.

Nesses artigos, buscamos identificar o cenário atual de cada contexto abordado, avaliar se a percepção é favorável ao uso do ChatGPT e fornecer um panorama de informações suficiente para discussão dos dilemas éticos e propor possíveis soluções.

3 - APRESENTAÇÃO DOS RESULTADOS

O quadro abaixo apresenta os resultados observados na síntese dos artigos utilizados como fonte deste trabalho:

FIGURA 73 - QUADRO DE PERCEPÇÕES, SOLUÇÕES E PRINCIPAIS DILEMAS ÉTICOS DE CADA CONTEXTO (PARTE 1)

Contexto	Percepção	Dilemas Éticos	Possíveis Soluções
Educação	Favorável com restrições	• Autenticidade do aprendizado • Qualidade da Avaliação • Uso adequado da ferramenta • Privacidade e segurança	• Utilizar formas diferentes de avaliação • Disponibilização de Treinamento/Cursos • Regulamentação
Saúde	Favorável	• Necessidade de aperfeiçoamento dos Algoritmos • Garantia da Privacidade dos Dados • Barreiras Culturais	• Transparência nos algoritmos usados. • Uso da IA como auxílio e não

FIGURA 74 - QUADRO DE PERCEPÇÕES, SOLUÇÕES E PRINCIPAIS DILEMAS ÉTICOS DE CADA CONTEXTO (PARTE 2)

Contexto	Percepção	Dilemas Éticos	Possíveis Soluções
		- Desumanização da Relação Médico-Paciente - Vieses de Algoritmos - Responsabilidade Médica	- substituição de diagnóstico final - Conscientização da possível agilidade adicional que a IA pode trazer no diagnóstico médico
Justiça e Sistema Legal	Favorável com ressalvas	- Precisão e confiabilidade - Privacidade dos Dados - Responsabilidade Legal	- Sempre revisar com cautela, textos gerados. - Criar e usar chatbots especializados e atualizados para consulta. - Regulamentação de boas práticas no uso.
Entretenimento	Crítica com restrições	- Uso de dados ilegais ou não autorizados - Compensação financeira - Responsabilidade pelo uso para fins ilegais ou imorais - Direitos autorais e criação híbrida	- Inibição do acesso ilegal a dados para treinamento de IAs - Definição legal de limites de posse para compensação monetária - Maior fiscalização e melhor rastreabilidade de artefatos gerados por IA - Definição dos limites de direitos autorais de artefatos gerados por IA que sofrem intervenção humana

4 - CONSIDERAÇÕES FINAIS

Na era atual, comparável a marcos tecnológicos como a Revolução Industrial e a Era dos Computadores, a inteligência artificial (IA) emerge como uma nova fronteira. Assim como ocorreu na Revolução Industrial, surgem debates sobre o impacto da IA no emprego, destacando-se a necessidade de adaptação a novos paradigmas profissionais.

Como qualquer tecnologia, a IA possui vantagens e desvantagens. Educadores, alunos e a sociedade em geral devem aprender a utilizar essas novas ferramentas de maneira eficaz.

Na área da saúde, a integração da IA promete melhorias significativas na precisão e eficiência dos diagnósticos. No entanto, é crucial implementar medidas robustas de segurança de dados e garantir conformidade com regulamentações como a LGPD. É essencial também que a IA complemente, e não substitua, o julgamento humano, preservando a empatia na relação médico-paciente.

No meio jurídico, a IA, como o ChatGPT, deve ser utilizada para tarefas de baixa complexidade, auxiliando os juristas. A responsabilidade final e a validação dos resultados devem sempre permanecer com os especialistas, garantindo a qualidade e a precisão das decisões.

No campo do entretenimento, a IA pode ser utilizada ética e criativamente, desde que o acesso aos dados para treinamento seja legal e consentido. Ela complementa, mas não substitui, a criatividade humana na produção de artes gráficas, textos e outros conteúdos.

Em resumo, a IA representa uma evolução tecnológica com potencial transformador em diversas áreas, desde que seja adotada de maneira ética e responsável, preservando sempre o papel central da inteligência humana e sua criatividade.

REFERÊNCIAS

CASSOL, Daniel. Quais os impactos do ChatGPT e da inteligência artificial na educação? Disponível em:

<<https://www.ifsc.edu.br/web/ifsc-verifica/w/quais-os-impactos-do-chatgpt-e-da-inteligencia-artificial-na-educacao->

#:~:text=%20%20ChatGPT%20p%C3%B5e%20em%20xeque,obter%20com%20uma%20boa%20aula>. Acesso em: 25 jun. 2024.

LEMOS, Ronaldo. ChatGPT e a arte de fazer perguntas. Disponível em

<<https://www1.folha.uol.com.br/colunas/ronaldolemos/2023/02/chatgpt-e-a-arte-de-fazer-perguntas.shtml>>. Acesso em: 25 jun. 2024.

MEDICINA DIRETA. Inteligência artificial no diagnóstico médico. Disponível em:

<<https://medicinadireta.com.br/blog/inteligencia-artificial-no-diagnostico-medico/>>. Acesso em: 02 jun. 2024.

BALERINI, Cristina. Uso da inteligência artificial na saúde: resultados e desafios.

Disponível em: <<https://www.saudebusiness.com/ti-e-inovacao/uso-da-inteligencia-artificial-na-saude-resultados-e-desafios>>. Acesso em: 17 jun. 2024.

NEURALMED. Proteção de dados na saúde. Disponível em:

<<https://www.neuralmed.ai/blog/protecao-de-dados-na-saude>>. Acesso em: 01 jun. 2024.

FORTI, Jefferson K.. A inteligência artificial como uma nova disciplina em medicina.

Disponível em: <<https://academiamedica.com.br/blog/a-inteligencia-artificial-como-uma-nova-disciplina-em-medicina>>. Acesso em: 05 jun. 2024.

RIPLA, Andre. AI and dehumanization in interactions: Exploring an ethical case.

Disponível em: <<https://www.linkedin.com/pulse/ai-dehumanization-interactions-exploring-ethical-case-ripla-pgcert-bnlye>>. Acesso em: 25 jun. 2024.

ARAUJO, Jailson S.; HORNUNG, Jociane A. Inteligência artificial e suas implicações éticas. Disponível em: <<https://revistathemis.tjce.jus.br/THEMIS/article/view/899/pdf>>.

Acesso em: 25 jun. 2024.

MELLO, Victor. Chat GPT na Advocacia. Disponível em:

<<https://www.migalhas.com.br/depeso/385996/chat-gpt-na-advocacia>>. Acesso em: 23 jun. 2024.

MODELO INICIAL. Chat GPT na advocacia: entenda os riscos e funcionalidades.

Disponível em: <<https://modeloinicial.com.br/artigos/chat-gpt>>. Acesso em: 23 jun. 2024.

Redação. Lee, Brock & Camargo Advogados restringe uso de ChatGPT e IA externas.

Disponível em: <<https://www.migalhas.com.br/quentes/386994/lee-brock-camargo-advogados-restringe-uso-de-chatgpt-e-ia-externas>>. Acesso em: 23 jun. 2024.

VIEIRA, Rodrigo. Advogado é multado pelo TSE por usar ChatGPT em petição. Disponível

em: <<https://www.jornaljurid.com.br/noticias/advogado-e-multado-pelo-tse-por-usar-chat-gpt-em-peticao>>. Acesso em: 23 jun. 2024.

FABRO, Clara. Advogado usa ChatGPT para fazer pesquisa jurídica e é multado em R\$

2,4 mil. Disponível em: <<https://www.techtudo.com.br/noticias/2023/07/advogado-usa-chatgpt-para-fazer-pesquisa-juridica-e-e-multado-em-r-24-mil-entenda-edviralizou.ghtml>>.

Acesso em: 23 jun. 2024.

STRECK, Lenio. ChatGPT inventando precedente e a terceirização da justiça. Disponível

em: <<https://www.conjur.com.br/2023-nov-16/chatgpt-inventando-precedente-e-a-terceirizacao-da-justica/>>. Acesso em: 23 jun. 2024.

PIGNATI, Giovana. ChatGPT: escritório de advocacia usa IA para desenvolver contrato.

Disponível em: <https://www.terra.com.br/byte/chatgpt-escritorio-de-advocacia-usa-ia-para-desenvolver-contrato,da814011f6a94389521dc6999824a98421z4mv6x.html?utm_source=clipboard>.

Acesso em: 23 jun.2024.

REDAÇÃO G1: Empresa que criou primeiro advogado robô do mundo é processada em

ação coletiva. Disponível em: <<https://g1.globo.com/tecnologia/noticia/2023/03/13/empresa-que-criou-primeiro-advogado-robo-do-mundo-e-processada-em-acao-coletiva.ghtml>>.

Acesso em: 23 jun.2024.

CHESTERMAN, Simon. Good models borrow, great models steal: intellectual property rights and generative AI. Disponível em: < <https://doi.org/10.1093/polsoc/puae006>>. Acesso em: 18 jun. 2024.

APÊNDICE L - GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

- Quais os principais resultados obtidos pelo estudo?

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

Não altere as questões do template.

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

B – RESOLUÇÃO

FIGURA 75 - QUADRO DE RESOLUÇÃO DO TRABALHO (PARTE 1)

Aluna/o: Igor Buess Atala Y Mansour Nome do artigo escolhido: Software engineering practices for machinelearning —Adoption, effects, and team assessment			
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 é investigar como equipes de especialistas desenvolvem, implantam e mantêm softwares com componentes de aprendizado de máquina (ML). Também busca identificar práticas de engenharia que contribuam para garantir a robustez e a confiabilidade dos sistemas de ML, com foco em propriedades como segurança, responsabilidade e justiça. Para alcançar esses objetivos, o estudo utilizou uma metodologia mista, incluindo uma revisão sistemática da literatura, uma pesquisa em larga escala e entrevistas com profissionais para validar e aprofundar os resultados.	O estudo foi motivado pelo crescente uso de técnicas de aprendizado de máquina (ML) em aplicações prontas para produção, o que aumenta a necessidade de práticas de engenharia maduras para garantir o desenvolvimento, a implantação e a manutenção robustos desses sistemas. Com a evolução rápida da área de ML e o surgimento constante de novas metodologias e ferramentas, os profissionais enfrentam desafios em disseminar e selecionar práticas adequadas para seus projetos. Além disso, faltam evidências empíricas amplas sobre a relevância dessas práticas em diferentes domínios e organizações. Assim, o estudo busca sistematizar e validar práticas de engenharia que possam ajudar equipes a aprimorar a qualidade dos sistemas de ML e atender a critérios de confiabilidade, como justiça e segurança, especialmente em aplicações de missão crítica onde erros técnicos podem ter impactos significativos para usuários, sociedade e meio ambiente.	Os autores utilizaram uma metodologia de métodos mistos, com uma estratégia exploratória sequencial, para coletar e analisar informações sobre as práticas de engenharia de software para aprendizado de máquina (ML). A metodologia foi dividida em três etapas principais: 1. Revisão Sistemática da Literatura (SLR): A primeira etapa foi uma revisão sistemática da literatura, que incluiu fontes acadêmicas e não acadêmicas. Essa revisão teve o objetivo de identificar e catalogar práticas de engenharia utilizadas no desenvolvimento de software com componentes de ML. Foram realizadas buscas em mecanismos como Google e Google Scholar, aplicando filtros e critérios de qualidade para selecionar os estudos mais relevantes. 2. Pesquisa em Larga Escala (Survey): A segunda etapa foi uma pesquisa quantitativa para validar as práticas identificadas na revisão da literatura e medir o grau de adoção dessas práticas entre profissionais. Essa pesquisa foi aplicada a mais de 400 praticantes de ML, que responderam a perguntas sobre a adoção das práticas e os efeitos percebidos dessas práticas em seus projetos. 3. Entrevistas de Validação com Profissionais: A terceira etapa incluiu	Os principais resultados do estudo incluem: 1. Catálogo de Práticas de Engenharia para ML: O estudo apresentou um catálogo abrangente de práticas de engenharia para o ciclo de vida completo do desenvolvimento de software com componentes de aprendizado de máquina. Essas práticas cobrem aspectos como governança, organização de equipes, e garantias de confiabilidade e robustez, com foco especial em práticas de ML confiável, como justiça, segurança e responsabilidade. 2. Adoção das Práticas: A pesquisa revelou que equipes maiores e mais experientes tendem a adotar mais práticas de engenharia para ML. No entanto, práticas relacionadas à confiabilidade e à responsabilidade — como evitar vieses, garantir a segurança e fornecer explicações aos usuários — ainda são amplamente negligenciadas, especialmente em empresas não tecnológicas e organizações governamentais. 3. Impacto das Práticas nos Efeitos Desejados: O estudo demonstrou que é possível prever certos efeitos, como agilidade da equipe, qualidade do software, rastreabilidade e responsabilidade, a partir de grupos específicos de práticas de engenharia. Por exemplo, práticas como o versionamento de dados e modelos e a monitorização

FIGURA 76 - QUADRO DE RESOLUÇÃO DO TRABALHO (PARTE 2)

		entrevistas qualitativas com profissionais para complementar os resultados quantitativos da pesquisa. Essas entrevistas forneceram uma perspectiva mais profunda sobre a relevância das práticas e os desafios enfrentados pelos profissionais ao adotar essas práticas. Os dados foram analisados utilizando estatísticas descritivas para as respostas da pesquisa, e métodos de análise temática para categorizar e interpretar as respostas das entrevistas. Essa abordagem de triangulação permitiu que os autores validassem as práticas catalogadas e analisassem a relação entre a adoção das práticas e os efeitos observados em projetos de ML.	continua da qualidade mostraram ter impacto positivo na capacidade de rastrear e reproduzir resultados. 4. Contraste entre Popularidade e Eficácia das Práticas: O estudo identificou que algumas práticas amplamente adotadas, como scripts reutilizáveis para limpeza de dados, têm impacto limitado em objetivos como a reprodutibilidade. Em contrapartida, práticas menos adotadas, principalmente as de confiabilidade, apresentam alto impacto em atributos desejáveis, como a responsabilidade e a transparência. 5. Propostas para Ferramentas de Avaliação de Equipes: Com base nos dados coletados, os autores sugerem o desenvolvimento de ferramentas que ajudem a avaliar e melhorar a capacidade de engenharia de equipes que trabalham com ML, o que pode impulsionar a adoção de práticas úteis e aumentar a qualidade e confiabilidade dos sistemas desenvolvidos. Esses resultados fornecem uma visão abrangente das práticas de engenharia de ML, destacando áreas de melhoria e oportunidades para incentivar práticas que tragam maior segurança e robustez para projetos com ML.
--	--	---	---

APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

2 Regressão (RNA)

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

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

Informações:

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

```

url =
"https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/win
equality-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_sno
w.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)

Breve explicação dos resultados:

- Gráficos de perda e de acurácia;

Em um teste com 10 épocas, na sexta época percebemos que os dados de validação estabilizam com perda perto de 0.33 e acurácia perto de 0.88.

Níveis relativamente bons pela quantidade de épocas de treinamento. - Imagem gerada na seção “Mostrar algumas classificações erradas”, apresentada na aula prática.

Nas classificações erradas temos uma bota de cano curto que o modelo classificou como uma sandália.

2 - Regressão (RNA)

Breve explicação dos resultados:

- Gráficos de avaliação do modelo (loss):

Utilizando o método “Early stop”, verificamos que o modelo estabiliza com valores muito bons na quadragésima oitava época com os dados de validação com perda perto de 0.0078.

- Métricas de avaliação do modelo (pelo menos uma entre MAE, MSE, R2). A princípio usando os dados brutos, o modelo apresentou um desempenho ruim, depois de normalizarmos os dados de Score obtivemos resultados satisfatórios com:

- mse = 0.004
- rmse = 0.0686
- r2 = 0.8442

3 - Sistemas de Recomendação

Breve explicação dos resultados:

- Gráficos de avaliação do modelo (loss);

Em um teste com 100 épocas, com o parâmetro de perda utilizando os dados de validação ficam entre 8.4 tendendo para 8.6 o que nos leva a inferir que mais épocas de treinamento não vão melhorar o modelo

- Exemplo de recomendação de livro para determinado Usuário.

- Recomendação para o usuário 276729:
- Livro - 373293062 / 4.967702 *
- Nome: High Country Hero (Harlequin Historical Series)

4 - Deepdream

Breve explicação dos resultados:

- Imagem onírica obtida por Main Loop;

O padrão visual do gatinho gerado pela rede neural se destaca com detalhes interessantes e a neve desaparece completamente

- Imagem onírica obtida ao levar o modelo até uma oitava;

Levando o modelo até oitava, verificamos que os padrões oníricos se intensificam e realmente parece mais um “gatinho dos sonhos”

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

Percebemos que tratando a imagem e elevando a uma oitava, o gatinho começa a ficar indistinguível parecendo até ser uma raposa.

CÓDIGO

```
#####
##### CLASSIFICAÇÃO #####
#####

import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from tensorflow import keras
from keras.datasets import fashion_mnist

(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.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)

display(x_train)

x_train, x_test = x_train/255.0, x_test/255.0

i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)

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

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

r = model.fit(x_train,
              y_train,
```

```

        validation_data=(x_test, y_test),
        epochs=10)

r = model.fit(x_train,
              y_train,
              validation_data=(x_test, y_test),
              epochs=10)

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

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

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

# Matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
                      show_normed=True)

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

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

#####
##### REGRESSÃO #####
#####

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

# Verificar versão do TensorFlow
tf.__version__

from sklearn.datasets import load_wine

data = load_wine()

print(data.keys())
print(data['target'])

X = data['data']
Y = data['target']

print(X.shape)
print(Y.shape)

from sklearn.datasets import load_wine

data = load_wine()

print(data.keys())
print(data['target'])

X = data['data']
Y = data['target']

print(X.shape)
print(Y.shape)

# 3 camadas
i = tf.keras.layers.Input(shape=(13,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)

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

```

```

# Criação de funções para as métricas R2 e RMSE serem inseridas no modelo
def rmse(y_true, y_pred):
    return sqrt(mean_squared_error(y_true, y_pred))

def r2(y_true, y_pred):
    return r2_score(y_true, y_pred)

# 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=['accuracy'])

# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True)

r = model.fit(x_train, y_train,
             epochs=1500,
             validation_data=(x_test, y_test),
             callbacks=[early_stop])

plt.plot( r.history["loss"], label="loss" )
plt.plot( r.history["val_loss"], label="val_loss" )
plt.legend()

plt.plot( r.history["rmse"], label="rmse" )
plt.plot( r.history["val_rmse"], label="val_rmse" )
plt.legend()

plt.plot( r.history["r2"], label="r2" )
plt.plot( r.history["val_r2"], label="val_r2" )
plt.legend()

# Predição

```

```

y_pred = model.predict(x_test).flatten()

# Cálculo das métricas de acurácia: mse, r2 e rmse
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Resultados das métricas de acurácia
print("mse      = ", mse)
print("rmse     = ", rmse)
print("r2       = ", r2)

#####
##### RECOMENDAÇÃO #####
#####

import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Embedding, Flatten,
Concatenate
from tensorflow.keras.models import Model
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

!wget https://marlonprudente.eng.br/files/Base_livros.csv

df = pd.read_csv("Base_livros.csv")
df.head()

!wget https://marlonprudente.eng.br/files/Base_livros.csv

df = pd.read_csv("Base_livros.csv")
df.head()

# prompt: transformar coluna Notas do df em int

```

```

# Convertendo a coluna 'Notas' para tipo numérico (int)
df['Notas'] = pd.to_numeric(df['Notas'],
errors='coerce').fillna(0).astype(int)
df['ISBN'] = pd.to_numeric(df['ISBN'],
errors='coerce').fillna(0).astype(int)
df['ID_usuario'] = pd.to_numeric(df['ID_usuario'],
errors='coerce').fillna(0).astype(int)

df[df['Notas'] > 10] = 10

df.head()

# Dimensões
N = len(set(df.ID_usuario))
M = len(set(df.ISBN))

# dimensão do embedding (tentar outros)
K = 10

# 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])

x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)

model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)

```

```

user_ids, book_ids, ratings = shuffle(df.ID_usuario, df.ISBN, df.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

train_user = user_ids[:Ntrain]
train_books = book_ids[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_books = 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

user_ids, book_ids, ratings = shuffle(df.ID_usuario, df.ISBN, df.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

train_user = user_ids[:Ntrain]
train_books = book_ids[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_books = 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

plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

# Gerar o array com o usuário único
# repete a quantidade de filmes

```

```

input_usuario = np.repeat(a=276729, repeats=M) #10058 276729
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)
book_data = df[df['ISBN'] == film[idx]]
print("Recomendação: Livro - ", film[idx], " / ", rat[idx] , "*")
print("Nome: ", book_data['Titulo'].values[0])

#####
##### DEEPDREAM #####
#####

import tensorflow as tf
import numpy as np

import matplotlib as mpl

import IPython.display as display
import PIL.Image

# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
    name = url.split('/')[-1]
    image_path = tf.keras.utils.get_file(name, origin=url)
    img = PIL.Image.open(image_path)
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)

# Normalização da imagem
def deprocess(img):
    img = 255*(img + 1.0)/2.0
    return tf.cast(img, tf.uint8)

```

```

# Mostra a imagem
def show(img):
    display.display(PIL.Image.fromarray(np.array(img)))

# Redução do tamanho da imagem para facilitar o trabalho da RNN
original_img = download(url, max_dim=500)
show(original_img)
display.display(display.HTML('Image cc-by: <a
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg">
Von.grzanka</a>'))

base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')

# Maximizando as ativações das camadas
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]

# Criação do modelo
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)

def calc_loss(img, model):
    # Passe a imagem pelo modelo para recuperar as ativações.
    # Converte a imagem em um batch de tamanho 1.
    img_batch = tf.expand_dims(img, axis=0)
    layer_activations = model(img_batch)
    if len(layer_activations) == 1:
        layer_activations = [layer_activations]

    losses = []
    for act in layer_activations:
        loss = tf.math.reduce_mean(act)
        losses.append(loss)

    return tf.reduce_sum(losses)

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)

deepdream = DeepDream(dream_model)

def run_deep_dream_simple(img, steps=100, step_size=0.01):

    img = tf.keras.applications.inception_v3.preprocess_input(img)
    img = tf.convert_to_tensor(img)
    step_size = tf.convert_to_tensor(step_size)
    steps_remaining = steps
    step = 0
    while steps_remaining:
        if steps_remaining > 100:
            run_steps = tf.constant(100)
        else:
            run_steps = tf.constant(steps_remaining)
        steps_remaining -= run_steps
        step += run_steps

        loss, img = deepdream(img, run_steps, tf.constant(step_size))

        display.clear_output(wait=True)
        show(deprocess(img))
        print ("Step {}, loss {}".format(step, loss))

result = deprocess(img)
display.clear_output(wait=True)
show(result)

```

```

    return result

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

```

APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

- O **conjunto de dados brutos** (ou **uma visualização de dados** que você acredite que possa ser **melhorada**);
- Explicação do **contexto e o público-alvo** da visualização de dados e do storytelling que será desenvolvido;
- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada; (50 pontos)**

B – RESOLUÇÃO

Conjunto de Dados Brutos:

Os dados brutos utilizados são de histórico epidemiológico do HIV em crianças e jovens de 0 a 19 anos, fornecidos pela UNICEF.

Link original com os dados brutos:

https://data.unicef.org/wp-content/uploads/2024/07/HIV_Epidemiology_Children_Adolescents_2024.xlsx

Contexto e público-alvo:

Durante o início dos anos 2000 as campanhas de prevenção contra o HIV populavam as televisões e rádios, instruindo sobre o uso de preservativos, principalmente no público jovem e que estava iniciando sua vida sexual.

Contudo nos últimos anos, apesar das campanhas terem perdido força e relevância, os dados epidemiológicos ainda mostram que há uma diminuição contínua nos números de infecções.

Sendo assim, o público-alvo desta publicação são os profissionais de medicina e governos.

Visualização dos dados

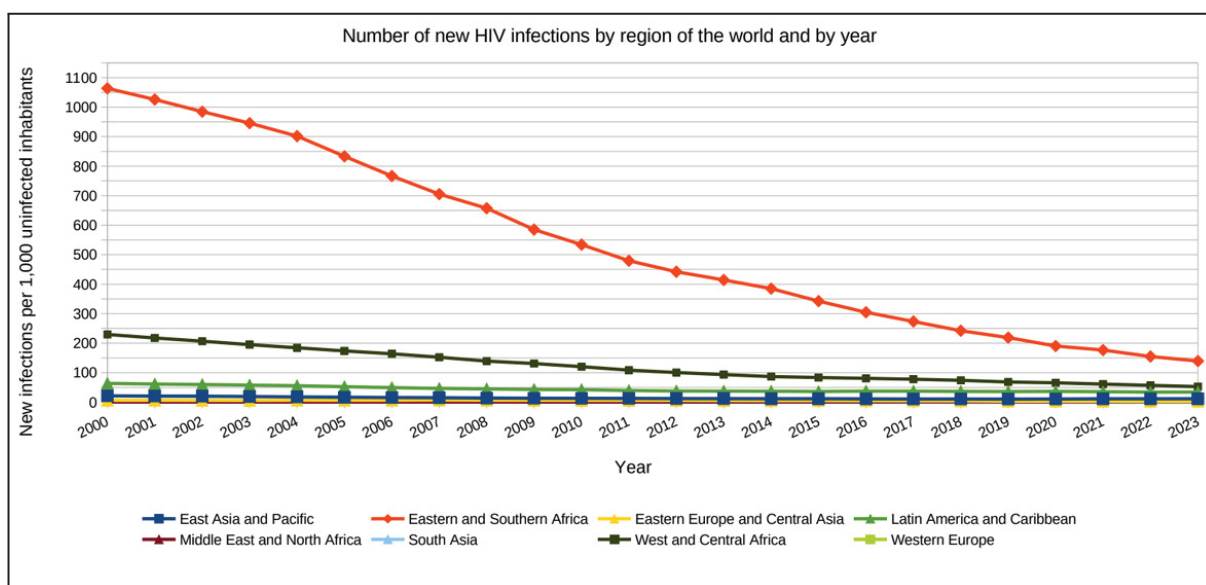
Com base nos dados brutos, criou-se uma visualização de dados que demonstrasse o nível e novas infecções ao longo dos anos, separadas por regiões do mundo.

A planilha principal utilizada está disponível no Zip enviado do trabalho com o nome: “HIV_Epidemiology_Children_Adolescents_2024_vFinal”

Para que isso fosse possível, a partir dos dados brutos, foi criada uma tabela dinâmica que consolidasse os dados numéricos por regiões do mundo (aba “Tabela dinâmica_Data_2”).

Com a tabela dinâmica feita, criou-se o gráfico abaixo, demonstrando as informações objetivadas:

FIGURA 77 - GRÁFICO DE QUANTIDADE DE INFECÇÕES DE HIV POR REGIÃO DO MUNDO



Entre 2000 e 2023, observou-se uma redução significativa no número de novas infecções por IV entre crianças e adolescentes de 0 a 19 anos em diversas regiões do mundo. Globalmente, as novas infecções nessa faixa etária diminuíram 62% desde 2010. Esse progresso é atribuído a iniciativas como a ampliação do acesso ao tratamento antirretroviral para mulheres grávidas vivendo

com HIV, que aumentou de 46% para 81% em uma década. No entanto, apesar dos avanços, desafios persistem, especialmente em regiões com maior carga da doença.

A África Subsaariana continua a ser a região mais afetada, registrando o maior número de novas infecções por HIV entre crianças e adolescentes. Fatores como desigualdade no acesso à informação, dificuldades na prevenção e barreiras ao tratamento ainda impactam significativamente essa população. Além disso, a transmissão vertical do HIV, de mãe para filho, continua sendo um problema relevante, exigindo políticas mais eficazes para garantir que todas as gestantes tenham acesso ao tratamento adequado e que seus filhos recebam a profilaxia necessária.

Em outras regiões, como a Ásia Oriental e o Pacífico, América Latina e Caribe, e Ásia do Sul, também houve progressos na redução de novas infecções por HIV entre crianças e adolescentes. No entanto, essas regiões ainda enfrentam desafios significativos, incluindo desigualdades no acesso ao tratamento e serviços de prevenção. Globalmente, quase um milhão de pessoas de 0 a 19 anos vivendo com HIV não estão recebendo tratamento, com mais da metade desse grupo concentrado na África Oriental e Austral. Esses dados ressaltam a importância de esforços contínuos e direcionados para alcançar metas globais de eliminação do HIV entre crianças e adolescentes.

APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java.

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida).

Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels.

O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

- (1) considere o cromossomo formado pelas cidades, onde a cidade de início (escolhida aleatoriamente) deverá estar na posição 0 e 100 e a ordem das cidades visitadas nas posições de 1 a 99 deverão ser definidas pelo algoritmo genético.
- (2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
- (3) Utilize no mínimo uma população com 100 indivíduos;
- (4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;
- (5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);
- (6) Preserve sempre a melhor solução de uma geração para outra.

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

1 - Algoritmo Genético

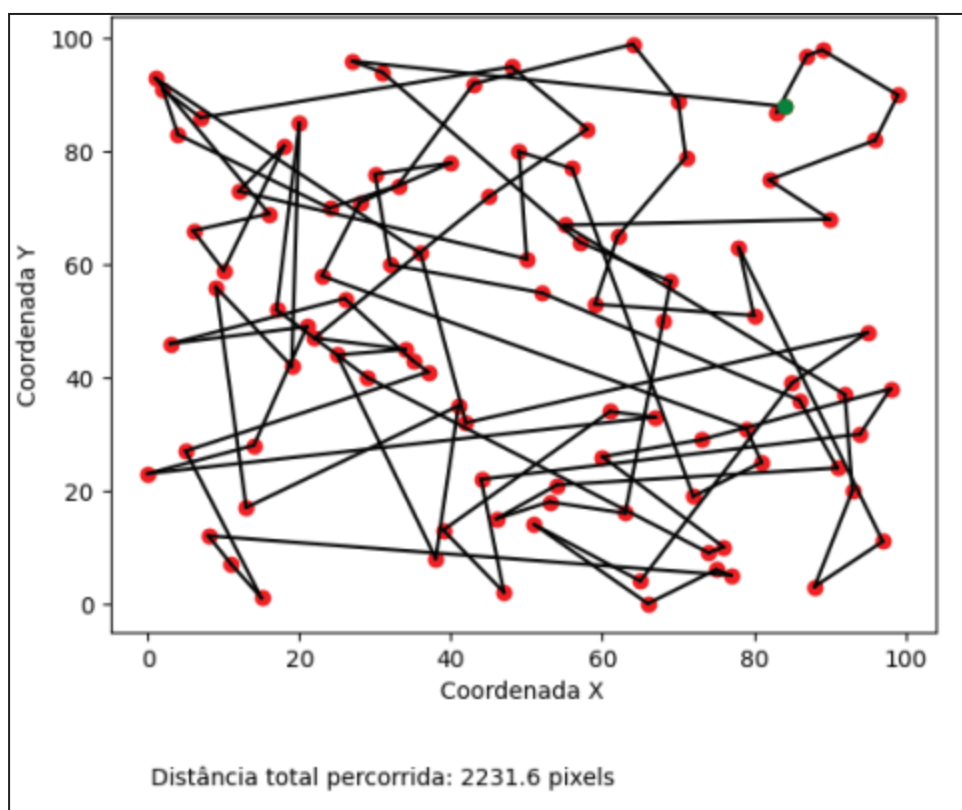
O Problema do Caixeiro Viajante (PCV) é um problema clássico de otimização combinatória, que busca o menor caminho para visitar um conjunto de cidades. Por ser um problema NP-Completo, sua complexidade aumenta superpolinomialmente com o número de cidades. Neste trabalho, um algoritmo genético foi usado para propor uma solução otimizada em relação a uma solução aleatória, dado que o PCV ainda não possui uma solução ótima para um número muito grande de cidades.

Para o cruzamento, foi utilizada a técnica de Crossover de Ordem (OX) com um ponto de corte, devido à restrição de que os valores das cidades não podem se repetir. Para a mutação, o indivíduo foi dividido em dois segmentos, e os alelos do segundo segmento foram embaralhados.

Na população inicial, foram criados 100 indivíduos aleatoriamente, e o indivíduo com o melhor fitness foi selecionado para a segunda geração, implementando o elitismo. Foram observadas 30.000 gerações, com uma taxa de cruzamento de 90% e uma taxa de mutação de 1%.

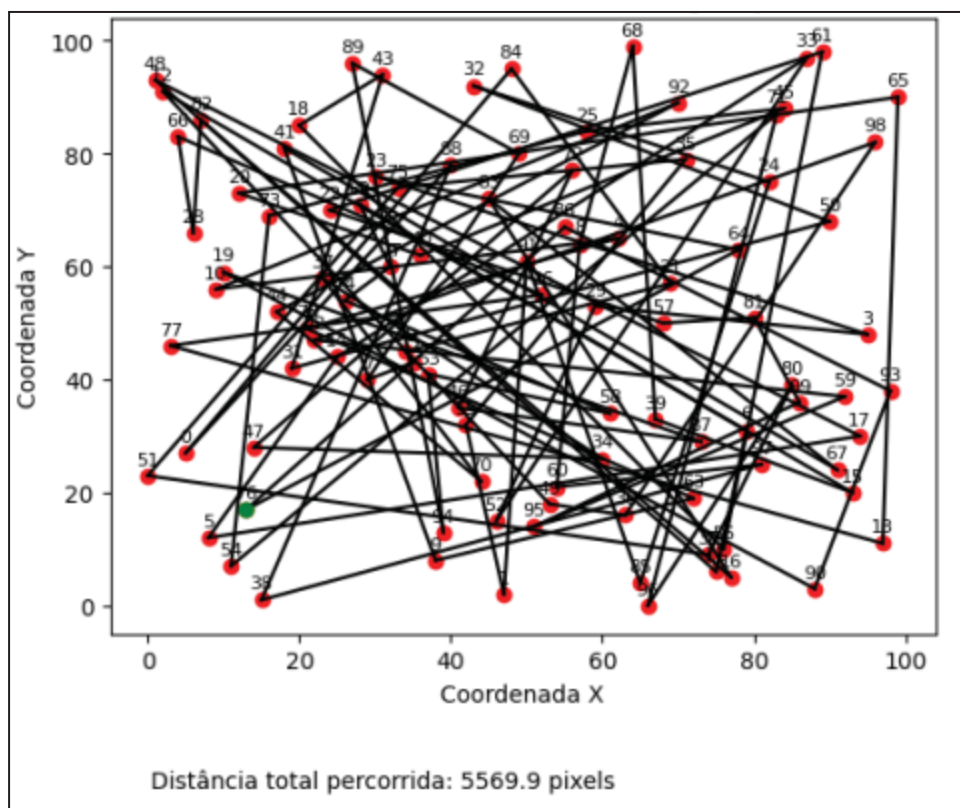
Melhor indivíduo da primeira geração:

FIGURA 78 - GRÁFICO DE CAMINHO DA PRIMEIRA GERAÇÃO



Melhor indivíduo da primeira geração:

FIGURA 79 - GRÁFICO DE CAMINHO ENCONTRADO



Os resultados demonstram a eficácia do AG em reduzir a distância total das rotas. Além da observação da menor distância euclidiana, a plotagem das rotas otimizadas revela uma significativa diminuição no número de cruzamentos entre os caminhos percorridos. Essa redução de cruzamentos sugere uma menor complexidade da rota, o que corrobora a diminuição da distância total percorrida.

2 - Comparando a representação de dois modelos vetoriais

Para visualizar a representação vetorial de textos, selecionamos dez fragmentos curtos, abrangendo temas como clima, culinária e animais, com similaridades intencionais para evidenciar o agrupamento vetorial. O pré-processamento incluiu conversão para minúsculas, remoção de pontuação, números e stopwords, seguido de tokenização.

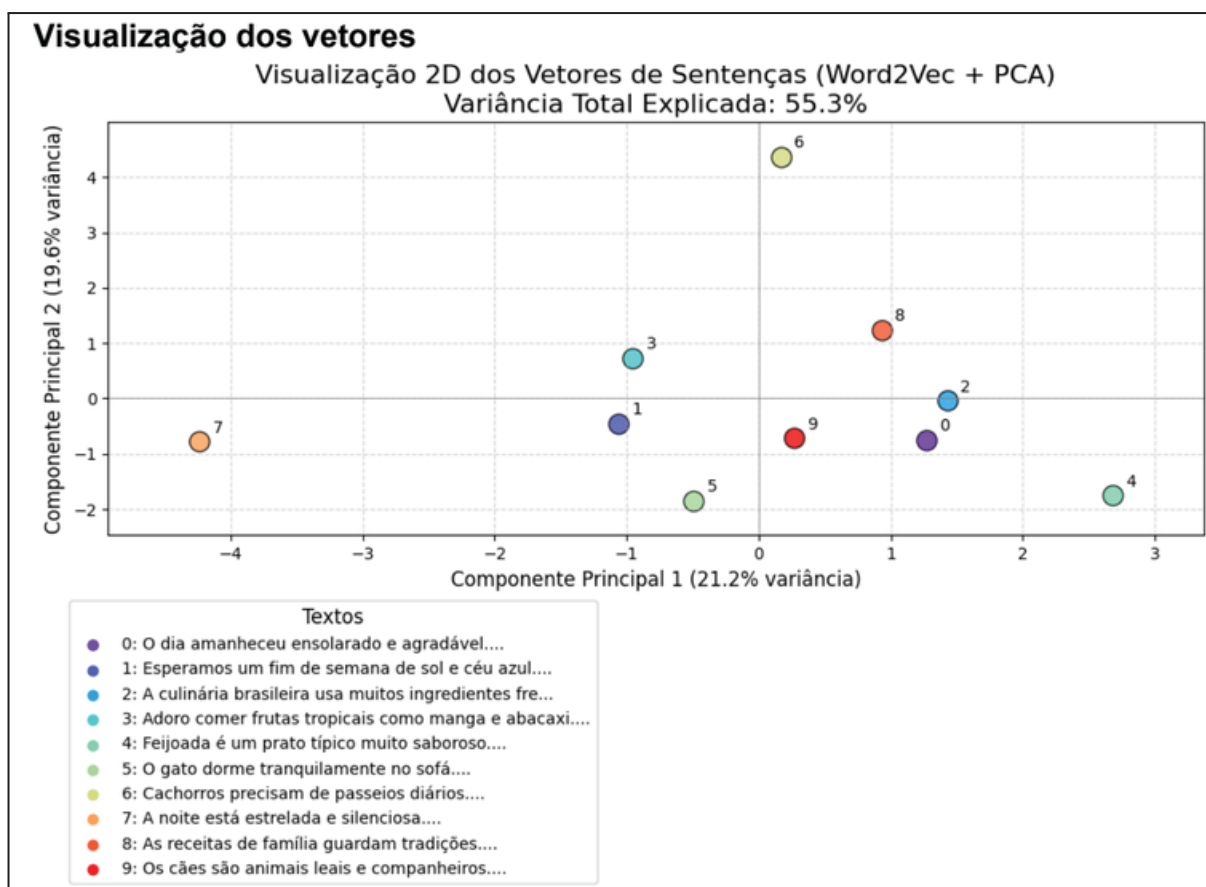
A representação vetorial foi obtida via Word2Vec, treinado com 100 épocas para capturar nuances semânticas em nosso pequeno corpus. A representação de cada sentença foi calculada pela média dos vetores das palavras, resultando em vetores de alta dimensionalidade.

A Análise de Componentes Principais (PCA) foi aplicada para reduzir a dimensionalidade, facilitando a visualização em gráficos 2D. Os resultados, multiplicados por 100 para clareza visual, demonstram o agrupamento de sentenças semanticamente similares, validando a eficácia da representação vetorial e da PCA na identificação de relações textuais.

FIGURA 80 - FRASES UTILIZADAS VS RELAÇÕES

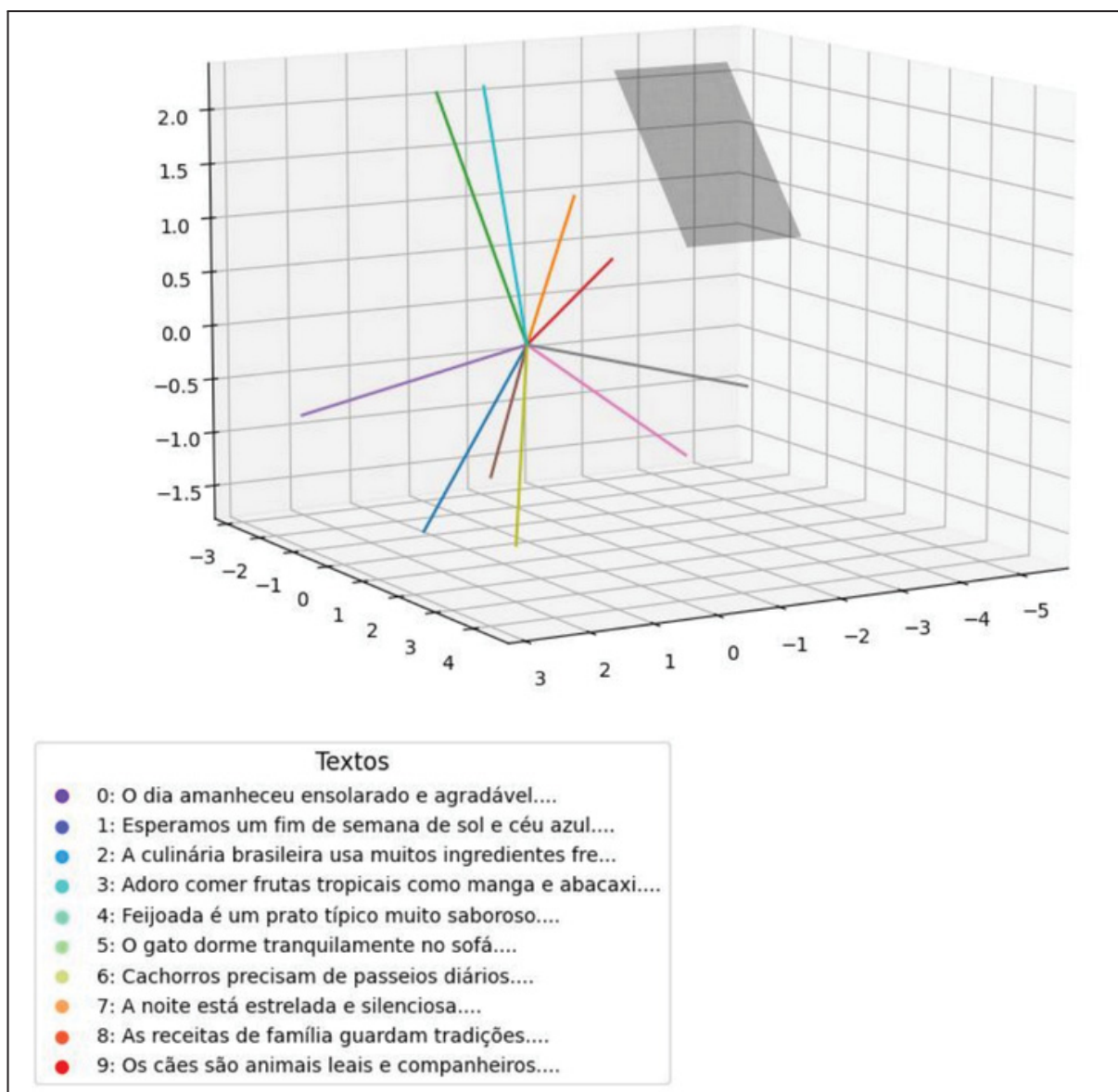
Frases utilizadas		relações
1	O dia amanheceu ensolarado e agradável.	Clima 1
2	Esperamos um fim de semana de sol e céu azul.	Clima 2 (similar a 1)
3	A culinária brasileira usa muitos ingredientes frescos.	Culinária 1
4	Adoro comer frutas tropicais como manga e abacaxi.	Culinária/frutas 2
5	Feijoada é um prato típico muito saboroso.	Culinária 3 (similar ao 3)
6	O gato dorme tranquilamente no sofá.	Animais 1
7	Cachorros precisam de passeios diários.	Animais 2
8	A noite está estrelada e silenciosa.	Clima/Noite 3 (ligeiramente similar ao 1/2)
9	As receitas de família guardam tradições.	Culinária 4 (similar ao 3/5)
10	Os cães são animais leais e companheiros.	Animais 3 (similar ao 7)

FIGURA 81 - VISUALIZAÇÃO DE VETORES



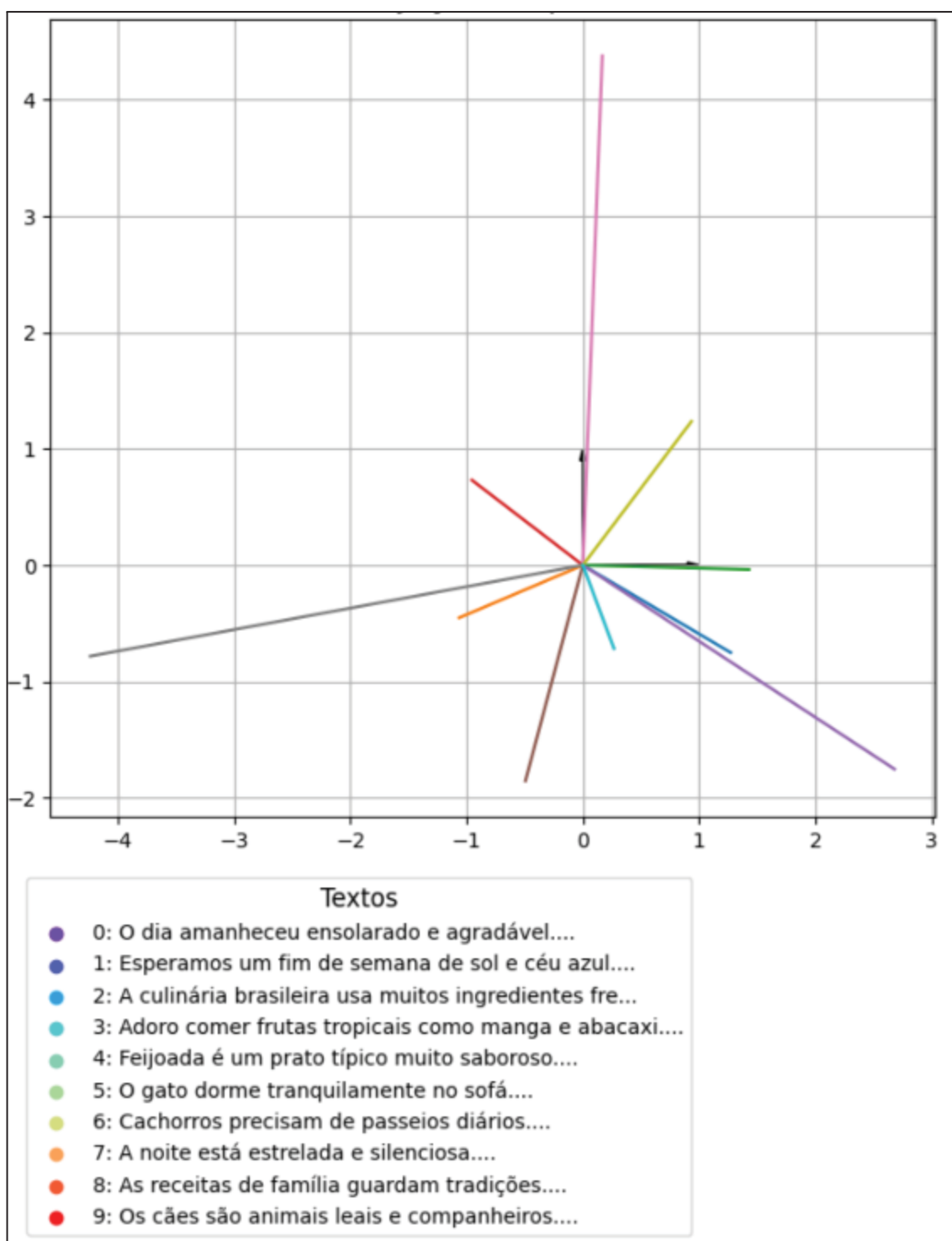
O gráfico acima mostra cada texto original como um ponto no espaço 2D. A posição de cada ponto é determinada pelo vetor da sentença (média dos vetores das palavras) após a redução com PCA. Textos com conteúdo semanticamente similar (segundo o modelo Word2Vec) devem aparecer mais próximos uns dos outros. Com um corpus pequeno, as relações capturadas pelo Word2Vec podem ser fracas ou baseadas em coocorrências simples de palavras.

FIGURA 82 - VISUALIZAÇÃO DOS VETORES DAS SENTENÇAS EM TRÊS DIMENSÕES



O gráfico acima representa a visualização dos vetores das sentenças em três dimensões, essa técnica é uma alternativa para gerar uma visualização 2D desses vetores com a dimensionalidade menor e, portanto, mais compreensível para a interpretação humana, como segue abaixo:

FIGURA 83 - PROJEÇÃO NO PLANO



Anexo I – Código Fonte do Exercício 1

```
import numpy as np
from random import randint, random
import random
import matplotlib.pyplot as plt
import math
import copy

NRO_CIDADES = 100

x_points = list(range(NRO_CIDADES))
y_points = list(range(NRO_CIDADES))

random.shuffle(x_points)
random.shuffle(y_points)

plt.title("Gráfico de cidades geradas.")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")
plt.plot(x_points, y_points, 'ro')
plt.show()

def populacaoInicial(tamanho):
    percursos = []
    percurso = []
    for i in range(0, tamanho):
        cidades = list(range(0, NRO_CIDADES))
        listaCidades = cidades.copy()
        random.shuffle(listaCidades)
        for cidade in cidades:
            cidadePercurso = random.choice(listaCidades)
            listaCidades.remove(cidadePercurso)
            percurso.append(cidadePercurso)
        percursos.append(percurso)
        percurso = []
        cidades = []
    return percursos;

##DEBUG função População Inicial
```

```

percursos = populacaoInicial(2)

for percurso in percursos:
    print(percurso)

def has_duplicates(lst):
    n = max(set(lst), key=lst.count)
    return lst.count(n) > 1

def has_duplicates(lst):
    n = max(set(lst), key=lst.count)
    return lst.count(n) > 1

##Calcular Distância Euclidiana
##distancia = sqrt(((x2 - x1) ^ 2) + ((y2 - y1) ^ 2));
def FuncaoDeAvaliacao(percursos):
    soma_distancia = np.zeros(len(percursos));
    for index, percurso in enumerate(percursos):
        for i, cidade in enumerate(percurso):
            if i < (len(percurso) - 1):
                soma_distancia[index] += math.sqrt((x_points[percurso[i + 1]] -
x_points[percurso[i]])**2 + (y_points[percurso[i + 1]] -
y_points[percurso[i]])**2);
            else:
                soma_distancia[index] += math.sqrt((x_points[percurso[i]] -
x_points[percurso[0]])**2 + (y_points[percurso[i]] -
y_points[percurso[0]])**2);
            if has_duplicates(percurso):
                soma_distancia[index] = 999999999
    return soma_distancia;

def FuncaoDeAvaliacaoV2(percurso):
    soma_distancia = 0
    for i, cidade in enumerate(percurso):
        if i < (len(percurso) - 1):
            soma_distancia += math.sqrt((x_points[percurso[i + 1]] -
x_points[percurso[i]])**2 + (y_points[percurso[i + 1]] -
y_points[percurso[i]])**2);

```

```

    else:
        soma_distancia += math.sqrt((x_points[percurso[i]] -
x_points[percurso[0]])**2 + (y_points[percurso[i]] -
y_points[percurso[0]])**2);
    if has_duplicates(percurso):
        soma_distancia = 999999999
    return soma_distancia;

##DEBUG Função de Avaliação

percursos = populacaoInicial(2)
print(percursos)
avaliacao = FuncaoDeAvaliacao(percursos)
print(avaliacao)

print(f'funcao v2 {FuncaoDeAvaliacaoV2(percursos[0])}')

def preservaMelhor(geracao, novaGeracao):
    avaliacao = FuncaoDeAvaliacao(geracao)
    maior = 0
    for i in range(len(avaliacao)):
        if avaliacao[maior] > avaliacao[i]:
            maior = i
    #print(avaliacao[maior])
    novaGeracao.append(geracao[maior])

#Debug preservar melhor geração
novaGeracao = []
preservaMelhor(percursos, novaGeracao)
print(novaGeracao)

##TODO - Implementar cruzamento OX, para evitar repetição

def cruzamento(geracao, num, nova):
    while True:
        for i in range(0, num):
            indA = random.randrange(0, len(geracao))
            indB = indA;
            while indA==indB:
                indB = random.randrange(0, len(geracao))

```

```

    #print(f'cruzamento {indA} e {indB}')
    ponto1 = random.randrange(1,round((len(geracao[0]) - 1)))
    #print(f'ponto de corte {ponto1}')

    paiA = geracao[indA]
    paiB = geracao[indB]

    res1 = list(set(paiA[0:ponto1]) & set(paiB[ponto1:]))
    res2 = list(set(paiB[0:ponto1]) & set(paiA[ponto1:]))

    cromossomosA = paiA[0:ponto1]
    cromossomosB = paiB[ponto1:]
    for i in range(0, len(res1)):
        cromossomosB.remove(res1[i])
        cromossomosB.append(res2[i])

    filhol1 = cromossomosA
    filhol1.extend(cromossomosB)
    #print(f'filhol1 {filhol1} duplicado? {has_duplicates(filhol1)}')
    nova.append(filhol1)
    break

##DEBUG cruzamento
novaGeracao = []
geracao = populacaoInicial(5)
print(geracao)
#preservaMelhor(geracao, novaGeracao)
cruzamento(geracao, 4, novaGeracao)
print(geracao)
print(novaGeracao)

#TODO: Implementar verificação de avaliação antes de inserir o individuo
que sofreu mutação na solução, incluir porcentagem para isso ocorrer

def mutacao(geracao, taxaMutacao):
    if randint(1,100) <= taxaMutacao:
        individuoMutacao = random.randrange(0,len(geracao))
        print(f'Realizando mutacao no Indivíduo: {individuoMutacao}')
        gene_1 = random.randrange(0,len(geracao[0]))
        gene_2 = random.randrange(0,len(geracao[0]))

```

```

while gene_1 == gene_2:
    gene_2 = random.randrange(0,len(geracao[0]))
    copiaGeracao = copy.deepcopy(geracao)
    temp = copiaGeracao[individuoMutacao][gene_1]
    copiaGeracao[individuoMutacao][gene_1] =
copiaGeracao[individuoMutacao][gene_2]
    copiaGeracao[individuoMutacao][gene_2] = temp
    #print(f'Individuo a ser mutado: {geracao[individuoMutacao]}
//Resultado após mutação: {copiaGeracao[individuoMutacao]}')
    avaliacaoCopia = FuncaoDeAvaliacaoV2(copiaGeracao[individuoMutacao])
    avaliacaoGeracao = FuncaoDeAvaliacaoV2(geracao[individuoMutacao])
    #print(f'Avaliação antes da mutação: {avaliacaoGeracao}')
    #print(f'Avaliação após a mutação: {avaliacaoCopia}')
    ##30% de chances de verificar se a mutação criou individuo melhor antes
de executar a mutação
    if random.randrange(0,100) < 101: ##Forçando a sempre entrar aqui para
testes
        print(f'Entrou nos 30% de chance de avaliação da Mutação - Avaliando
...')
        if avaliacaoCopia < avaliacaoGeracao:
            print(f'Avaliação superior ao anterior, enviando para a mansão X')
            geracao[individuoMutacao] = copiaGeracao[individuoMutacao]
        else:
            print(f'Avaliação abaixo do anterior, tentando uma nova mutacao
...')
            mutacao(geracao,100)
        else:
            print(f'Mutação não considerou avaliação')
            geracao[individuoMutacao] = copiaGeracao[individuoMutacao]

#Teste Mutacao

geracao = populacaoInicial(100)
print(geracao)

for i in range(0,100):
    mutacao(geracao, 100)
print(geracao)

#PRESETS AG

```

```

#-----
numGeracoes = 10000
populacao = 100
taxaCruzamento = 90
taxaMutacao = 1
#-----
p0 = populacaoInicial(populacao)
melhorPrimeiraGeracao = []
preservaMelhor(p0, melhorPrimeiraGeracao)
print(f'Distância percorrida:
{FuncaoDeAvaliacaoV2(melhorPrimeiraGeracao[0])}')

plt.title("Gráfico do primeiro caminho encontrado.")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")
plt.plot(x_points, y_points, 'ro')

for i in range(0, len(melhorPrimeiraGeracao[0]) - 1):
    plt.plot([x_points[melhorPrimeiraGeracao[0][i]],
x_points[melhorPrimeiraGeracao[0][i + 1]],
[y_points[melhorPrimeiraGeracao[0][i]], y_points[melhorPrimeiraGeracao[0][i
+ 1]]], 'k-')
    if i == (len(melhorPrimeiraGeracao[0]) - 2):
        plt.plot([x_points[melhorPrimeiraGeracao[0][i + 1]],
x_points[melhorPrimeiraGeracao[0][0]],
[y_points[melhorPrimeiraGeracao[0][i + 1]],
y_points[melhorPrimeiraGeracao[0][0]]], 'k-')
plt.show()

avaliacaoP0 = FuncaoDeAvaliacao(p0)

geracao = copy.deepcopy(p0)
while numGeracoes > 0:

    if numGeracoes % 100 == 0:
        print(f'Geração {numGeracoes}')

    novaGeracao = []
    mutacao(geracao, taxaMutacao)

```

```

preservaMelhor(geracao, novaGeracao)
cruzamento(geracao, taxaCruzamento, novaGeracao)
numGeracoes = numGeracoes - 1
avaliacaoNovaGeracao = FuncaoDeAvaliacao(novaGeracao)
novaGeracao2 = sorted(zip(avaliacaoNovaGeracao, novaGeracao),
reverse=True)
geracao = [x for _, x in novaGeracao2]

geracao = novaGeracao.copy()
novaGeracao = []
preservaMelhor(geracao, novaGeracao)

plt.title("Gráfico de caminho encontrado.")
plt.xlabel("Coordenada X")
plt.ylabel("Coordenada Y")
plt.plot(x_points, y_points, 'ro')

print(f'Distância percorrida: {FuncaoDeAvaliacaoV2(novaGeracao[0])}')
print(f'Duplicidade? {has_duplicates(novaGeracao[0])}')

for i in range(0, len(novaGeracao[0]) - 1):
    plt.plot([x_points[novaGeracao[0][i]], x_points[novaGeracao[0][i + 1]],
[y_points[novaGeracao[0][i]], y_points[novaGeracao[0][i + 1]], 'k-')
    if i == (len(novaGeracao[0]) - 2):
        plt.plot([x_points[novaGeracao[0][i + 1]], x_points[novaGeracao[0][0]],
[y_points[novaGeracao[0][i + 1]], y_points[novaGeracao[0][0]], 'k-')

plt.plot(x_points[novaGeracao[0][0]], y_points[novaGeracao[0][0]], 'go')
plt.show()

```

Anexo II – Código Fonte do Exercício 2

```

#Gensim do Colab não funciona tem que instalar e reiniciar
!pip install gensim nltk scikit-learn matplotlib numpy
# --- 0. Instalação e Imports (Necessário no Colab se não estiver
pré-instalado) ---

```

```

# Descomente e execute as linhas !pip se receber erro de módulo não
encontrado
# !pip install gensim nltk scikit-learn matplotlib numpy

import re # Para expressões regulares (limpeza de texto)
import numpy as np
from gensim.models import Word2Vec
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import matplotlib.cm as cm # Para gerar cores distintas para os pontos
import nltk
import math
from warnings import simplefilter
from scipy.cluster.hierarchy import dendrogram, linkage, ClusterWarning
#pacotes necessários para o tokenizador
nltk.download('punkt')
# Baixar o tokenizador 'punkt' do NLTK (necessário na primeira execução no
ambiente)
try:
    nltk.data.find('tokenizers/punkt')
except nltk.downloader.DownloadError:
    print("Baixando o tokenizador 'punkt' do NLTK...")
    nltk.download('punkt', quiet=True)
    print("Download concluído.")

print("Bibliotecas importadas e NLTK pronto.")
print("-" * 40)
# --- 1. Definição dos Textos ---
# Mínimo 6 textos, com pelo menos 2 pares similares.
# Limitando a ~10-12 para clareza na visualização.
texts = [
    "O dia amanheceu ensolarado e agradável.",          # Clima 1
    "Esperamos um fim de semana de sol e céu azul.",    # Clima 2
    (similar ao 1)
    "A culinária brasileira usa muitos ingredientes frescos.", # Culinária
1
    "Adoro comer frutas tropicais como manga e abacaxi.", #
Culinária/Frutas 2
    "Feijoada é um prato típico muito saboroso.",      # Culinária 3
    (similar ao 3)

```

```

    "O gato dorme tranquilamente no sofá.",          # Animais 1
    "Cachorros precisam de passeios diários.",        # Animais 2
    "A noite está estrelada e silenciosa.",            # Clima/Noite 3
    (ligeiramente similar ao 1/2)
    "As receitas de família guardam tradições.",      # Culinária 4
    (similar ao 3/5)
    "Os cães são animais leais e companheiros.",      # Animais 3
    (similar ao 7)
]

# Garantir que não exceda um limite razoável para o plot
max_texts_to_plot = 12
if len(texts) > max_texts_to_plot:
    print(f"Limitando a {max_texts_to_plot} textos para visualização.")
    texts = texts[:max_texts_to_plot]

print(f"Textos de entrada ({len(texts)}):")
for i, t in enumerate(texts):
    print(f"{i}: {t}")
print("-" * 40)
nltk.download('punkt_tab')
# --- 2. Pré-processamento ---
def preprocess_text(text):
    """Limpa e tokeniza o texto."""
    # Converte para minúsculas
    text = text.lower()
    # Remove pontuações básicas e números (opcional, mas ajuda a focar nas
    palavras)
    text = re.sub(r'^\w\s', '', text) # Remove pontuação
    text = re.sub(r'\d+', '', text)   # Remove números
    # Tokeniza (divide em palavras)
    tokens = nltk.word_tokenize(text, language='portuguese') # Especificar
    idioma ajuda
    # Remove palavras vazias (stopwords) - Opcional, pode ou não ajudar com
    poucos dados
    from nltk.corpus import stopwords
    nltk.download('stopwords') # Se for usar
    stop_words = set(stopwords.words('portuguese'))
    tokens = [word for word in tokens if word not in stop_words and
    len(word) > 1] # Remove stopwords e palavras de 1 letra

```

```

    tokens = [word for word in tokens if len(word) > 1] # Apenas remove
palavras de 1 letra
    return tokens

tokenized_texts = [preprocess_text(t) for t in texts]

print("Textos Tokenizados (Exemplo do primeiro):")
print(tokenized_texts[0])
print("-" * 40)

# --- 3. Treinamento do Modelo Word2Vec ---
# Parâmetros importantes para datasets pequenos:
# min_count=1: Inclui todas as palavras. Essencial aqui.
# epochs: Aumentar o número de épocas de treino.
# vector_size: Dimensão dos vetores (100 é comum, mas 50 pode ser
suficiente aqui).
# window: Contexto de palavras vizinhas.
# sg=1: Usa Skip-gram (geralmente melhor para datasets menores).

vector_dim = 50 # Dimensão dos vetores de palavras
w2v_model = Word2Vec(sentences=tokenized_texts,
                      vector_size=vector_dim,
                      window=3,          # Janela menor pode ser melhor para
frases curtas
                      min_count=1,       # Crucial para datasets pequenos
                      workers=4,         # Número de threads (ajuste conforme
CPU disponível no Colab)
                      sg=1,             # Skip-gram
                      epochs=100)       # Mais épocas para compensar poucos
dados

print("Modelo Word2Vec treinado.")
print(f"Tamanho do vocabulário: {len(w2v_model.wv.index_to_key)}")
# Exemplo: Palavras mais similares a 'sol' (se 'sol' estiver no
vocabulário)
try:
    similar_words = w2v_model.wv.most_similar('sol', topn=3)
    print(f"Palavras mais similares a 'sol': {similar_words}")
except KeyError:
    print("Palavra 'sol' não encontrada no vocabulário (corpus muito
pequeno).")

```

```

print("-" * 40)
# --- 4. Cálculo dos Vetores de Sentenças ---
# Estratégia: Média dos vetores das palavras na sentença

def get_sentence_vector(tokens, model, vector_size):
    """Calcula o vetor médio para uma lista de tokens."""
    vector = np.zeros(vector_size)
    count = 0
    for word in tokens:
        if word in model.wv: # Checa se a palavra está no vocabulário do
            modelo
            vector += model.wv[word]
            count += 1
    if count > 0:
        vector /= count # Calcula a média
    # else: # Opcional: Lidar com sentenças onde nenhuma palavra está no
    vocabulário
    #     print(f"Aviso: Sentença com tokens '{tokens}' não possui palavras
    no vocabulário.")
    return vector
# Calcula o vetor para cada texto tokenizado
sentence_vectors = np.array([get_sentence_vector(tokens, w2v_model,
vector_dim)
                             for tokens in tokenized_texts])

print(f"Vetores das sentenças calculados. Shape: {sentence_vectors.shape}")
# Verificar se há vetores nulos (indicativo de problemas)
if np.any(np.all(sentence_vectors == 0, axis=1)):
    print("AVISO: Uma ou mais sentenças resultaram em vetor nulo (nenhuma
palavra no vocabulário?).")
print("-" * 40)
# --- 5. Redução de Dimensionalidade com PCA ---
# Reduzir os vetores de 'vector_dim' dimensões para 2 dimensões para
plotagem

pca = PCA(n_components=3, random_state=42) # random_state para
reprodutibilidade
pca_result = pca.fit_transform(sentence_vectors)

print(f"Vetores reduzidos para 2D com PCA. Shape: {pca_result.shape}")

```

```

print("Componentes Principais (x, y, z) para cada sentença:")

pca_result = pca_result * 100 #Multiplicado por 100 para melhor
visualização nos gráficos.

for i, (x, y, z) in enumerate(pca_result):
    print(f"Texto {i}: ({x:.4f}, {y:.4f}, {z:.4f})")
print("-" * 40)
# --- 6. Visualização dos Vetores 2D ---
#
plt.figure(figsize=(12, 9)) # Ajuste o tamanho conforme necessário

# Gerar cores distintas para cada ponto usando um colormap
colors = cm.rainbow(np.linspace(0, 1, len(texts)))

# Plotar cada ponto (vetor 2D da sentença)
for i, (x, y, z) in enumerate(pca_result):
    plt.scatter(x, y, color=colors[i], s=150, alpha=0.8, edgecolors='k')
    # Adicionar anotação (número do texto) ligeiramente deslocado do ponto
    plt.text(x + 0.1, y + 0.1, str(i), fontsize=10, ha='left', va='bottom')

# Variância explicada
explained_variance_ratio = pca.explained_variance_ratio_
total_explained_variance = explained_variance_ratio.sum() * 100

plt.title(f'Visualização 2D dos Vetores de Sentenças (Word2Vec +
PCA)\nVariância Total Explicada: {total_explained_variance:.1f}%',
fontsize=16)
plt.xlabel(f'Componente Principal 1 ({explained_variance_ratio[0]*100:.1f}%
variância)', fontsize=12)
plt.ylabel(f'Componente Principal 2 ({explained_variance_ratio[1]*100:.1f}%
variância)', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.5)

legend_handles = [plt.scatter([], [], color=colors[i], label=f'{i}:
{texts[i][:50]}...')]
    for i in range(len(texts))]
plt.legend(handles=legend_handles, loc='center left', bbox_to_anchor=(0.5,
-0.5),

```

```

        fontsize='medium', title="Textos", title_fontsize='large',
borderaxespad=1.)

plt.margins(0.1)
plt.axhline(0, color='grey', lw=0.5)
plt.axvline(0, color='grey', lw=0.5)

plt.tight_layout(rect=[0, 0, 0.85, 1])
plt.show()

print("\n--- Interpretação do Gráfico ---")
print("O gráfico acima mostra cada texto original como um ponto no espaço
2D.")
print("A posição de cada ponto é determinada pelo vetor da sentença (média
dos vetores das palavras) após a redução com PCA.")
print("Textos com conteúdo semanticamente similar (segundo o modelo
Word2Vec) devem aparecer mais próximos uns dos outros.")
print("Com um corpus pequeno, as relações capturadas pelo Word2Vec podem
ser fracas ou baseadas em coocorrências simples de palavras.")
centro = pca_result.mean(axis=0)
v_centro = pca_result - centro

print('centro', '\n', centro)
print( np.allclose(pca_result, v_centro+centro))

# obter o plano para projeção ortogonal
U,s,Vh = np.linalg.svd(v_centro)
print('Variância: ', np.square(s) / np.square(s).sum() )
W2 = Vh.T[:, :2]
v1_plano = Vh.T[:, 0]
v2_plano = Vh.T[:, 1]
projetados2d = v_centro.dot(W2)
print(projetados2d)

# calculando o plano de projeção
menor = np.min(pca_result,axis=0)-1
maior = np.min(pca_result,axis=0)+1
x1s = np.linspace(menor[0], maior[0], 10)
y1s = np.linspace(menor[1], maior[1], 10)
z1s = np.linspace(menor[2], maior[2], 10)
C = Vh

```

```

R = C.T.dot(C)
x1, x2 = np.meshgrid(x1s, y1s)
z = (R[0, 2] * x1 + R[1, 2] * x2) / (1 - R[2, 2])
#
# tentando visualizar as projeções no espaço 3d
fig = plt.figure(figsize=(12, 9)) # Ajuste o tamanho conforme necessário
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x1, x2, z, alpha=0.2, color="k")
#ax.view_init(elev=180, azim=180)
# pontos originais
for v in pca_result:
    xs = np.linspace(0, v[0], 10)
    ys = np.linspace(0, v[1], 10)
    zs = np.linspace(0, v[2], 10)
    plt.plot(xs,ys,zs=zs)
plt.show()
fig = plt.figure(figsize=(12, 9))
ax = fig.add_subplot(111, aspect='equal')
# pontos projetados
for v in pca_result:
    xs = np.linspace(0, v[0], 10)
    ys = np.linspace(0, v[1], 10)
    ax.plot(xs,ys)

ax.arrow(0, 0, 0, 1, head_width=0.05, \
        length_includes_head=True, head_length=0.1, fc='k', ec='k')
ax.arrow(0, 0, 1, 0, head_width=0.05, \
        length_includes_head=True, head_length=0.1, fc='k', ec='k')
ax.grid(True)
#plt.xscale('log')
#plt.yscale('log')
#plt.xlim(-0.05,0.05)
#plt.ylim(-0.05,0.05)
plt.legend(handles=legend_handles, loc='center left', bbox_to_anchor=(0.5,
-0.5),
        fontsize='medium', title="Textos", title_fontsize='large',
borderaxespad=1.)
plt.show()
def vectorDistance(v1,v2):
    soma = 0

```

```

    for i in range(0,len(v1)):
        d = v1[i]-v2[i]
        soma = soma + d*d
    return math.sqrt(soma)

def calcDistance(mat):
    nrows = len(mat)
    ncols = len(mat[0])
    res = np.zeros( (nrows,nrows) )
    for i in range(0,nrows):
        m1 = mat[i]
        for j in range(0,nrows):
            if i!=j:
                m2 = mat[j]
                res[i][j] = vectorDistance(m1,m2)
            else:
                res[i][j] = 0
    return res

d = calcDistance(pca_result)
d1 = calcDistance(projetados2d)
print('original')
print(d)
print('projetada')
print(d1)

simplefilter("ignore", ClusterWarning)
Z1 = linkage(d, 'single')
fig = plt.figure()
dn = dendrogram(Z1)
plt.show()
Z2 = linkage(d1, 'single')
fig = plt.figure()
dn = dendrogram(Z2)
plt.show()

```