

UNIVERSIDADE FEDERAL DO PARANÁ

MATHEUS MELNIK KEMER

MEMORIAL DE PROJETOS: INTELIGÊNCIA ARTIFICIAL E CIÊNCIA DE DADOS
APLICADAS AO MONITORAMENTO DE ECOSISTEMAS CRÍTICOS

CURITIBA

2026

MATHEUS MELNIK KEMER

MEMORIAL DE PROJETOS: INTELIGÊNCIA ARTIFICIAL E CIÊNCIA DE DADOS
APLICADAS AO MONITORAMENTO DE ECOSISTEMAS CRÍTICOS

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

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2026



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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **MATHEUS MELNIK KEMER**, intitulada: **MEMORIAL DE PROJETOS: INTELIGÊNCIA ARTIFICIAL E CIÊNCIA DE DADOS APLICADAS AO MONITORAMENTO DE ECOSISTEMAS CRÍTICOS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

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

Curitiba, 24 de Fevereiro de 2026.

JAIME WOJCIECHOWSKI

Presidente da Banca Examinadora

RAZER ANTHON MIER ROJAS MONTAÑO

Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este Memorial de Projetos apresenta a consolidação técnica das competências desenvolvidas na Especialização em Inteligência Artificial Aplicada, demonstrando a aptidão para arquitetar e executar soluções baseadas em dados. O trabalho integra disciplinas fundamentais como Engenharia de Dados, Aprendizado de Máquina e Estatística Computacional para resolver problemas de alta complexidade. Como aplicação prática e prova de conceito, foi desenvolvido um estudo analítico sobre o desmatamento na Amazônia, utilizando uma *pipeline* de tratamento de dados históricos do PRODES (Projeto de Monitoramento do Desmatamento da Floresta Amazônica Brasileira por Satélite). A metodologia aplicada envolveu a estruturação de bases massivas, análise de séries temporais e detecção de anomalias estatísticas. Os resultados validaram matematicamente a eficiência de políticas de preservação passadas e identificaram rupturas de tendência críticas no cenário atual, além de comprovar, via segmentação de dados, o papel estratégico das Terras Indígenas como barreiras de contenção. O portfólio confirma a capacidade técnica de transformar dados brutos em inteligência estratégica, utilizando a IA como ferramenta de auditoria e suporte à decisão em cenários de incerteza.

Palavras-chave: inteligência artificial; engenharia de dados; aprendizado de máquina; monitoramento estratégico; amazônia.

ABSTRACT

This Project Memorial presents the technical consolidation of skills developed during the Specialization in Applied Artificial Intelligence, demonstrating the aptitude to architect and execute data-driven solutions. The work integrates fundamental disciplines such as Data Engineering, Machine Learning, and Computational Statistics to solve high-complexity problems. As a practical application and proof of concept, an analytical study on deforestation in the Amazon was developed, utilizing a pipeline for processing historical data from PRODES (Brazilian Amazon Forest Satellite Monitoring Project). The applied methodology involved structuring massive datasets, time series analysis, and statistical anomaly detection. The results mathematically validated the efficiency of past preservation policies and identified critical trend breaks in the current scenario, in addition to proving, via data segmentation, the strategic role of Indigenous Lands as containment barriers. The portfolio confirms the technical capability to transform raw data into strategic intelligence, using AI as a tool for auditing and decision support in uncertain scenarios.

Keywords: artificial intelligence; data engineering; machine learning; strategic monitoring; amazon.

SUMÁRIO

1 PARECER TÉCNICO.....	7
REFERÊNCIAS.....	10
APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA.....	11
APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA.....	18
APÊNDICE C - LINGUAGEM R.....	23
APÊNDICE D - ESTATÍSTICA APLICADA I.....	28
APÊNDICE E - ESTATÍSTICA APLICADA II.....	31
APÊNDICE F - ARQUITETURA DE DADOS.....	41
APÊNDICE G - APRENDIZADO DE MÁQUINA.....	72
APÊNDICE H - DEEP LEARNING.....	83
APÊNDICE I - BIG DATA.....	104
APÊNDICE J - VISÃO COMPUTACIONAL.....	109
APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	158
APÊNDICE L - GESTÃO DE PROJETOS DE IA.....	167
APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	172
APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING.....	196
APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	203

1 PARECER TÉCNICO

Este Memorial de Projetos tem a finalidade de documentar a trajetória técnica desenvolvida ao longo do curso, focando na arquitetura de soluções de dados de ponta a ponta (*End-to-End Data Solutions*). A abordagem adotada transcende a simples execução de códigos, alinhando-se aos fundamentos estabelecidos por Russell e Norvig (2022), que definem a Inteligência Artificial moderna não apenas como algoritmos isolados, mas como agentes racionais que operam em ambientes estocásticos para maximizar medidas de desempenho. A proposta central é apresentar o domínio integral sobre o ciclo de vida da Ciência de Dados (*Data Science Lifecycle*), desde a ingestão da engenharia bruta até a entrega de inteligência estratégica, demonstrando a capacidade de transformar grandes volumes de dados não estruturados e tabulares em ativos decisórios de alto valor agregado.

A base operacional deste portfólio apoia-se na Engenharia de Dados e Pré-processamento, etapas críticas que definem a viabilidade e a escalabilidade de qualquer projeto analítico. A manipulação das bases históricas exigiu a construção de *pipelines* de ETL (*Extract, Transform and Load* - Extrair, Transformar e Carregar) robustos, utilizando a biblioteca Pandas para a estruturação de *DataFrames* de alta performance e Numpy para operações vetoriais complexas, conforme as melhores práticas documentadas pela Pandas Development Team (2025). Seguindo os princípios de *Tidy Data* defendidos por Wickham (2014), o tratamento de inconsistências, a imputação de valores ausentes e a normalização das escalas consumiram a maior parte do esforço computacional. No contexto do Projeto PRODES (1988-2024), essa etapa foi decisiva para mitigar ruídos intrínsecos à coleta de dados geoespaciais, assegurando que a modelagem subsequente não fosse comprometida pelo fenômeno do *Garbage In, Garbage Out*, onde a qualidade da saída (resultado) de um sistema é diretamente dependente da qualidade da entrada (dados).

No nível de inteligência analítica, o trabalho evidencia um domínio profundo sobre o Aprendizado de Máquina Supervisionado, com ênfase na seleção parcimoniosa de modelos. A arquitetura de solução priorizou algoritmos de *ensemble*, como *Random Forest* e *Gradient Boosting*, em detrimento de abordagens lineares clássicas. Essa decisão técnica fundamenta-se nos estudos seminais de Breiman (2001), que comprovam a superioridade estatística desses métodos para lidar com a alta dimensionalidade e as interações não-lineares típicas de dados ambientais complexos. A implementação prática, realizada via framework Scikit-learn (Pedregosa *et al.*, 2011), envolveu rotinas rigorosas de validação cruzada (*Cross-Validation*) e otimização de hiperparâmetros (*Grid Search*), garantindo o equilíbrio ideal entre viés e variância para maximizar a capacidade de generalização do modelo em dados não vistos.

Expandindo o escopo para o estado da arte em IA, foram aplicadas e testadas técnicas de *Deep Learning* para tarefas de reconhecimento de padrões avançados. A fundamentação teórica baseou-se na obra de Goodfellow, Bengio e

Courville (2016), utilizando Redes Neurais Convolucionais (CNNs) e arquiteturas profundas como VGG16 e ResNet para capturar hierarquias de características que modelos tradicionais (*shallow learners*) falham em processar. Adicionalmente, a exploração de Processamento de Linguagem Natural (NLP) tocou na fronteira do conhecimento com arquiteturas baseadas em *Attention Mechanisms* (mecanismos de atenção), essenciais para a compreensão moderna de IA Generativa, conforme introduzido no *paper* revolucionário de Vaswani *et al.* (2017). Essa diversidade técnica demonstra aptidão para transitar entre dados estruturados e não estruturados com a mesma proficiência.

Como prova de conceito para validar essa *stack* tecnológica integrada, apresento a auditoria analítica sobre o desmatamento na Floresta Amazônica. Os dados brutos foram extraídos e validados conforme as diretrizes oficiais do INPE (Instituto Nacional de Pesquisas Espaciais - 2025), garantindo a rastreabilidade e a oficialidade das métricas utilizadas. A aplicação de Análise de Séries Temporais permitiu decompor a evolução dos dados em componentes de tendência, sazonalidade e resíduo, identificando rupturas estruturais (*structural breaks*) com precisão estatística. Foi possível isolar matematicamente o período de 2004 a 2012 — caracterizado por uma queda vertiginosa no desmatamento — e contrastá-lo com a retomada da degradação nos anos subsequentes, transformando o histórico bruto em uma narrativa analítica clara.

A interpretação desses dados ultrapassou a análise descritiva, avançando para a inferência causal sobre a eficácia de políticas públicas. Os resultados obtidos corroboram os estudos econométricos de Assunção, Gandour e Rocha (2012), do *Climate Policy Initiative*, que demonstram que a aplicação efetiva de políticas de comando e controle foi o fator determinante para a redução do desmatamento naquele intervalo histórico. Minha análise reproduziu os padrões identificados por esses autores através de novos modelos, transformando terabytes de dados históricos em uma validação algorítmica independente, provando que a ciência de dados pode atuar como uma ferramenta de auditoria governamental de alta precisão.

Além da modelagem temporal, o trabalho aplicou lógica de segmentação e testes de hipóteses para validar a eficiência de diferentes governanças territoriais. A análise comprovou estatisticamente, através de testes não-paramétricos de Mann-Whitney e Kolmogorov-Smirnov, que as Terras Indígenas apresentam taxas de preservação significativamente superiores às de áreas privadas ou de uso não designado. Esta inferência estatística alinha-se às descobertas ecológicas de Nepstad *et al.* (2006), que já apontavam as áreas protegidas como os "escudos" mais eficazes contra a expansão da fronteira agrícola. Para garantir a robustez científica dessas conclusões, a metodologia seguiu o rigor estatístico proposto por Conover (1999), assegurando que as diferenças observadas possuissem significância estatística real ($p\text{-value} < 0.05$) e não fossem fruto de aleatoriedade amostral.

Em suma, este Memorial atesta a aptidão para liderar projetos de tecnologia que exigem uma visão 360 graus, integrando engenharia de software robusta,

modelagem estatística rigorosa e visão de negócio. Conforme destaca Davenport e Harris (2017) em seus estudos sobre a competição analítica, a capacidade de decidir com base em fatos e modelos preditivos é o diferencial definitivo das organizações modernas. O perfil profissional aqui apresentado materializa essa visão, utilizando a Inteligência Artificial não como uma promessa futurista abstrata, mas como um mecanismo pragmático, auditável e eficiente para a resolução de problemas complexos.

REFERÊNCIAS

- ASSUNÇÃO, J; GANDOUR, C; ROCHA, R. Deforestation slowdown in the Legal Amazon: prices or policies. **Climate Policy Initiative**, Rio de Janeiro, 2012. Disponível em: <https://www.climatepolicyinitiative.org>. Acesso em: 20 ago. 2025.
- BREIMAN, L. Random Forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001.
- CHANG, Chih-Chung; LIN, Chih-Jen. LIBSVM: A Library for Support Vector Machines. **ACM Transactions on Intelligent Systems and Technology**, v. 2, n. 3, p. 1–27, 2011.
- DAVENPORT, Thomas H.; HARRIS, Jeanne G. **Competing on Analytics: The New Science of Winning**. Boston: Harvard Business Review Press, 2017.
- GOODFELLOW, I; BENGIO, Y; COURVILLE, A. **Deep Learning**. Cambridge: MIT Press, 2016.
- INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS (INPE). **Projeto PRODES: Monitoramento do Desmatamento da Floresta Amazônica Brasileira por Satélite**. Disponível em: <http://www.obt.inpe.br/OBT/assuntos/programas/amazonia/prodes>. Acesso em: 23 ago. 2025.
- NEPSTAD, D. Inhibition of Amazon Deforestation and Fire by Parks and Indigenous Lands. **Conservation Biology**, v. 20, n. 1, p. 65-73, 2006.
- PANDAS DEVELOPMENT TEAM. **pandas-dev/pandas: Flexible and powerful data analysis/manipulation library for Python**. Disponível em: <https://pandas.pydata.org>. Acesso em: 15 set. 2025.
- PEDREGOSA, F. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825-2830, 2011.
- PROJETO PRODES - Monitoramento do Desmatamento da Floresta Amazônica Brasileira por Satélite. Disponível em: <https://dados.gov.br/dados/conjuntos-dados/prodes> . Acesso em: 23 ago. 2025.
- RUSSELL, S; NORVIG, P. **Inteligência Artificial: Uma Abordagem Moderna**. 4. ed. Porto Alegre: Grupo GEN, 2022.
- VASWANI, A. Attention is All You Need. **Advances in Neural Information Processing Systems (NeurIPS)**, v. 30, 2017.
- WICKHAM, H. Tidy Data. **Journal of Statistical Software**, v. 59, n. 10, 2014.

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.

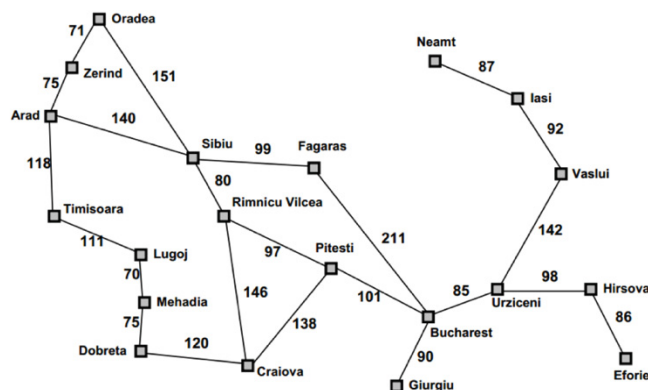
2 Busca Heurística

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

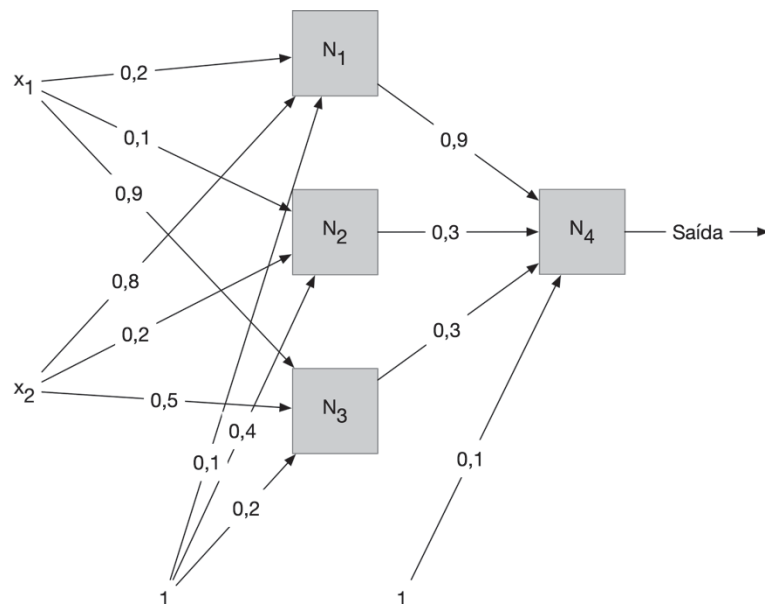
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

Segundo o ChatGPT:

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

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

A IA tem aplicações em uma ampla variedade de setores, incluindo medicina, finanças, manufatura, transporte, entretenimento e muito mais."

Classificação segundo as quatro abordagens (Russell e Norvig, 2022):

- Agir como Humanos: parcialmente relacionada.
- Pensar como Humanos: não replica diretamente processos mentais, mas se inspira em redes neurais.
- Pensar Racionalmente: vai além da lógica pura, contemplando decisões sob incerteza.

- Agir Racionalmente: é a que mais se aplica, pois busca sempre o melhor resultado possível.

O ChatGPT é um chatbot da OpenAI baseado na arquitetura **GPT (Generative Pre-Trained Transformer)**. Seu funcionamento combina três fases:

- Pré-treinamento auto-supervisionado (aprendizado autoregressivo).
- Treinamento supervisionado com pares pergunta-resposta.
- Aprendizado por Reforço com Feedback Humano (RLHF).

A arquitetura **Transformer** (Vaswani et al., 2017) usa mecanismos de atenção para lidar com contexto, permitindo aplicações em texto, imagens, áudios, etc. Essa combinação torna o ChatGPT uma ferramenta robusta e adaptável a diversos contextos.

Busca Heurística - Algoritmo A*

Problema: encontrar o melhor caminho de **Lugoj** até **Bucharest** usando o algoritmo A*, considerando a heurística da distância em linha reta.

O algoritmo utiliza:

```
#CODE
```

$$f(n) = g(n) + h(n)$$

```
#ENDCODE
```

O resultado esperado é a construção da árvore de busca mostrando os valores de f , g e h em cada nó.

(Aqui você deve inserir a imagem da árvore que construiu.)

Lógica Proposicional

Verificar se o argumento é válido:

1. Se as uvas caem, então a raposa as come.
2. Se a raposa as come, então estão maduras.
3. As uvas estão verdes ou caem.

Conclusão: A raposa come as uvas se e somente se as uvas caem.

R1: $p \rightarrow q$
 R2: $q \rightarrow r$
 R3: $\neg r \vee p$
 C: $q \leftrightarrow p$
 Equivalências:
 R4: $\neg p \vee q$
 R5: $\neg q \vee r$
 R6: $r \rightarrow p$
 Inferências:
 R7: $p \rightarrow r$
 R8: $q \rightarrow p$
 R9: $r \rightarrow q$
 Conjunção:
 R10: $(q \rightarrow p) \wedge (p \rightarrow q)$
 Bicondicional:
 R11: $(q \leftrightarrow p)$

Portanto, o argumento é válido. 

Redes Neurais Artificiais

Configuração:

- N1, N2, N3 $\rightarrow$ ativação linear.
- N4 $\rightarrow$ ativação tangente hiperbólica (tanh).

- Entradas: $x_1 = -3$, $x_2 = 1$.

$$N1 = (-3)0,2 + 10,8 + 10,1 = 0,3$$

$$N2 = (-3)0,1 + 10,2 + 10,4 = 0,3$$

$$N3 = (-3)0,9 + 10,5 + 1*0,2 = -2$$

$$x = N10,9 + N20,3 + N30,3 + 10,1$$

$$x = -0,14$$

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

$$N4 \approx -0,14$$

O valor final da rede é aproximadamente **-0,14**.

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

O dataset original possui 267.542 registros e 11 colunas. Foram removidos 65.245 registros que apresentavam valores faltantes em todas as colunas, resultando em 202.297 registros. Destes havia 2 registros duplicados que quando removidos resultaram em 202.295 registros.

'year_of_reference', 'year_model', 'avg_price_brl' são explicitamente numéricas. 'month_of_reference' também poderia ser transformada em numérica para facilitar o ordenamento em algumas análises. 'engine_size' também parece ser uma boa candidata a ser numérica, pois possui somente valores numéricos e que possuem ordem entre si. As demais colunas são categóricas.

Os dados referem-se aos anos de referência entre 2021 e 2023, com anos dos modelos entre 2000 e 2023, sendo que metade deles estão entre 2000 e 2012 e outra metade até 2023. "Fiat Palio Week. Adv/Adv TRYON 1.8 mpi Flex" e "Ford Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p" são os modelos que contêm mais registros ao longo do período de referência presente no dataset.

Com relação aos preços (em Reais), o valor médio é de 52.756,91 e um desvio padrão de 51.628,68, que se distribui ao longo do valor mínimo de 6.647,00 e máximo de 979.358,00.

```
import pandas as pd
df = pd.read_csv("precos_carros_brasil.csv")
df.shape
df.isnull().sum()
df.dropna(how="all", inplace=True)
df.drop_duplicates(inplace=True)
df.info()
df.describe()
```

Geralmente carros automáticos são vendidos com um preço médio superior ao dos carros manuais, podendo chegar ao dobro do valor, que é o caso de algumas marcas como a Fiat e VW.

A marca Renault é a única em que o preço médio de carro automático é menor que manual.

A marca Ford é a que tem o menor preço médio para automáticos: R\$ 84.769,00.

A marca VW-Volkswagen é a que tem o maior preço médio para automáticos: R\$ 99.735,00.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot(x="brand", y="avg_price", hue="gear", data=df)
plt.xticks(rotation=90)
plt.show()
```

Pode-se perceber que em todas as marcas os carros se tornam mais caros quando são movidos à diesel, depois gasolina e por último álcool. Renault e Nissan não possuem carros à álcool no dataset analisado.

Os carros à diesel têm um preço médio que corresponde a aproximadamente o dobro que os carros à gasolina. Já os carros à álcool têm um preço médio muito menor.

Os preços médios dos carros à diesel são extremamente elevados em todas as marcas, sendo a mais cara a VW chegando a um preço médio de R\$ 139.216,00.

```
sns.barplot(x="brand", y="avg_price", hue="fuel", data=df)
plt.xticks(rotation=90)
plt.show()
```

Analisando os melhores modelos (pelo critério da métrica R^2) dos dois algoritmos (Random Forest Regressor e XGBRegressor), percebe-se que ambos os modelos foram baseados no dataset com o conjunto de features mais completo. Isso indica que as variáveis inicialmente consideradas menos correlacionadas com a variável target ainda contribuíram para o bom desempenho dos modelos.

A variável **engine_size** foi a mais importante em ambos os modelos, embora year_model também tenha mantido alta relevância. Já tempo_mercado apresentou menor importância, possivelmente por depender de year_model.

```

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
X = df[["engine_size", "year_model", "tempo_mercado"]]
y = df["avg_price"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=42)
rf = RandomForestRegressor()
rf.fit(X_train, y_train)
xgb = XGBRegressor()
xgb.fit(X_train, y_train)
rf.feature_importances_, xgb.feature_importances_

```

Com base nas métricas utilizadas, o melhor modelo para aplicar nos dados foi o **Random Forest Regressor com parâmetros default**, pois obteve menores valores de MSE e MAE e o maior valor de R^2 .

Logo depois aparece outro Random Forest ajustado por busca randômica de parâmetros, seguido do XGBoostRegressor também configurado. A diferença entre os três modelos foi pequena, mas o Random Forest default ainda apresentou o melhor desempenho geral.

```

y_pred_rf = rf.predict(X_test)
y_pred_xgb = xgb.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
mae_xgb = mean_absolute_error(y_test, y_pred_xgb)
r2_xgb = r2_score(y_test, y_pred_xgb)
print("RandomForest - MSE:", mse_rf, "MAE:", mae_rf, "R2:", r2_rf)
print("XGBoost - MSE:", mse_xgb, "MAE:", mae_xgb, "R2:", r2_xgb)

```

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. Classificação de Imagens de Satélite (Satellite)

Foi utilizada a base de dados Satellite do pacote *mlbench*.

O conjunto foi dividido em 80% treino e 20% teste.

Foram treinados três modelos: Random Forest, SVM e RNA.

A comparação foi feita com base nas matrizes de confusão.

O Random Forest apresentou melhor desempenho geral, com maior acurácia e precisão nas classes, mostrando-se o modelo mais adequado para esta tarefa.

```

library(mlbench)
library(caret)
library(randomForest)
library(e1071)
library(nnet)
data(Satellite)
set.seed(42)
index <- createDataPartition(Satellite$classes, p=0.8, list=FALSE)
train_data <- Satellite[index, ]
test_data <- Satellite[-index, ]
Random Forest
rf_model <- randomForest(classes ~ ., data=train_data)
rf_pred <- predict(rf_model, test_data)
confusionMatrix(rf_pred, test_data$classes)
SVM
svm_model <- svm(classes ~ ., data=train_data)
svm_pred <- predict(svm_model, test_data)
confusionMatrix(svm_pred, test_data$classes)
RNA
nn_model <- nnet(classes ~ ., data=train_data, size=10, maxit=200,
trace=FALSE)
nn_pred <- predict(nn_model, test_data, type="class")
confusionMatrix(nn_pred, test_data$classes)

```

2. Estimativa de Volumes de Árvores

Foi utilizada a base `Volumes.csv`, contendo as variáveis DAP (diâmetro à altura do peito), HT (altura total) e VOL (volume observado).

O conjunto foi dividido em 80% treino e 20% teste.

Foram testados quatro modelos:

- Random Forest (rf)
- SVM radial (svmRadial)
- Rede Neural (neuralnet)
- Modelo alométrico de Spurr

As métricas utilizadas foram: R^2 , Syx e Syx%.

O modelo com melhor desempenho foi o Random Forest, que apresentou maior R^2 e menores erros, sendo portanto o escolhido como final.

```

library(caret)
library(randomForest)
library(e1071)
library(neuralnet)
dados <- read.csv("Volumes.csv", sep=";", header=TRUE)

```

```

dados$NR <- NULL # remove coluna sequencial
set.seed(42)
index <- createDataPartition(dados$VOL, p=0.8, list=FALSE)
train_data <- dados[index, ]
test_data <- dados[-index, ]
Random Forest
rf_model <- randomForest(VOL ~ ., data=train_data)
rf_pred <- predict(rf_model, test_data)
SVM
svm_model <- train(VOL ~ ., data=train_data, method="svmRadial")
svm_pred <- predict(svm_model, test_data)
Rede Neural
nn_model <- neuralnet(VOL ~ DAP + HT, data=train_data, hidden=c(5,3),
linear.output=TRUE)
nn_pred <- compute(nn_model, test_data[, c("DAP", "HT")])$net.result
Modelo de Spurr (regressão não-linear)
alom <- nls(VOL ~ b0 + b1DAPDAP*HT, data=train_data, start=list(b0=0.5,
b1=0.5))
spurr_pred <- predict(alom, test_data)
Funções de avaliação
R2 <- function(obs, pred) { cor(obs, pred)^2 }
Syx <- function(obs, pred) { sqrt(sum((obs - pred)^2) / (length(obs) - 2)) }
SyxPerc <- function(obs, pred) { (Syx(obs,pred) / mean(obs)) * 100 }
Comparação dos modelos
resultados <- data.frame(
Modelo = c("RandomForest", "SVM", "Rede Neural", "Spurr"),
R2 = c(R2(test_data$VOL, rf_pred),
R2(test_data$VOL, svm_pred),
R2(test_data$VOL, nn_pred),
R2(test_data$VOL, spurr_pred)),
Syx = c(Syx(test_data$VOL, rf_pred),
Syx(test_data$VOL, svm_pred),
Syx(test_data$VOL, nn_pred),
Syx(test_data$VOL, spurr_pred)),
SyxPerc = c(SyxPerc(test_data$VOL, rf_pred),
SyxPerc(test_data$VOL, svm_pred),
SyxPerc(test_data$VOL, nn_pred),
SyxPerc(test_data$VOL, spurr_pred))
)
print(resultados)

```

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

Ao analisar os boxplots e histogramas, observou-se que:

- A idade mínima entre maridos e esposas é muito semelhante.
- A idade máxima difere: esposas chegam a ~60 anos, maridos até ~70 (com outliers acima disso).

- A mediana e os quartis dos maridos são ligeiramente maiores.
- As idades dos maridos são mais distribuídas na cauda direita, enquanto a das esposas interrompe em torno de 60 anos.
- Até aproximadamente 40 anos há predominância de esposas; a partir disso, os maridos apresentam maior frequência.

```
boxplot(salarios$age, salarios$husage, col=c("blue","red"),
names=c("Esposas","Maridos"),
main="Boxplot de Idades",
ylab="Idade")
hist(salarios$husage, col="red", main="Histograma Maridos",
xlab="Idade", ylab="Frequência")
hist(salarios$age, col="blue", main="Histograma Esposas",
xlab="Idade", ylab="Frequência")
```

As tabelas de frequência confirmam o padrão:

- Maridos apresentam maior dispersão, com valores até 86 anos.
- Esposas têm distribuição mais homogênea e concentrada.

2. Medidas de Posição e Dispersão

Os resultados mostraram:

- Média, mediana e moda dos maridos são maiores que das esposas.
- Moda: esposas = 37 anos; maridos = 44 anos.
- A diferença entre média e mediana é maior nos maridos, reforçando assimetria à direita.

```
media_age <- mean(salarios$age)
mediana_age <- median(salarios$age)
moda_age <- as.numeric(names(which.max(table(salarios$age))))
media_husage <- mean(salarios$husage)
mediana_husage <- median(salarios$husage)
moda_husage <- as.numeric(names(which.max(table(salarios$husage))))
comparacao <- data.frame(Variavel=c("age","husage"),
Media=c(media_age,media_husage),
Mediana=c(mediana_age,mediana_husage),
Moda=c(mod_a_age,moda_husage))
print(comparacao)
```

Quanto à dispersão:

- Variância, desvio padrão e coeficiente de variação foram maiores nos maridos.

- Isso reforça que a idade dos maridos é mais dispersa e assimétrica.

```
var_age <- var(salarios$age)
dp_age <- sd(salarios$age)
cv_age <- dp_age/media_age
var_husage <- var(salarios$husage)
dp_husage <- sd(salarios$husage)
cv_husage <- dp_husage/media_husage
comparacao2 <- data.frame(Variavel=c("age", "husage"),
  Variancia=c(var_age, var_husage),
  Desvio_Padrao=c(dp_age, dp_husage),
  Coeficiente_de_Variacao=c(cv_age, cv_husage))
print(comparacao2)
```

3. Testes Paramétricos ou Não Paramétricos

Para verificar se as idades medianas de maridos e esposas são estatisticamente iguais:

- Premissas de normalidade (Kolmogorov-Smirnov e Lilliefors) foram rejeitadas ($p < 0,001$).
- Assim, utilizou-se o teste não paramétrico de Mann-Whitney.
- Resultado: rejeição de H_0 ($p < 0,001$).
- Conclusão: as idades medianas não são estatisticamente iguais.
- Intervalo de confiança para a diferença das medianas ≈ -3 anos.

```
ks.test(idades$idade[idades$grupo=="esposa"], "pnorm",
  mean=mean(idades$idade[idades$grupo=="esposa"]),
  sd=sd(idades$idade[idades$grupo=="esposa"]))
ks.test(idades$idade[idades$grupo=="marido"], "pnorm",
  mean=mean(idades$idade[idades$grupo=="marido"]),
  sd=sd(idades$idade[idades$grupo=="marido"]))
lillie.test(idades$idade)
wilcox.test(idade ~ grupo, data=idades, exact=FALSE, conf.int=TRUE)
```

Análise final:

- O teste de Mann-Whitney mostrou que maridos têm idades medianas estatisticamente maiores.
- O resultado reforça o padrão observado nos gráficos e medidas descritivas.

APÊNDICE E - ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

Bibliotecas:

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

```
Lendo a base de dadosload("./trabalhosalarios.RData")
df <- trabalhosalarios
head(df, 10)
```

```
##      husage husunion husearns huseduc husblk hushisp hushrs kidge6 earns
age
## 3      56      0      1500      14      0      0      40      1      100
49
## 13     31      0      800      17      0      0      40      0      480
29
## 20     33      0      950      13      0      0      60      0      455
30
## 21     34      0     1000      14      0      0      50      1      102
31
## 22     42      0      730      14      0      0      40      1      300
41
## 25     45      0     1154      16      0      0      38      1      425
45
## 26     33      0     1350      16      0      0      40      0      770
32
## 27     31      0      769      18      0      0      55      0      125
27
## 29     31      0      340      12      0      0      40      0      245
30
## 31     44      0      750      12      0      0      40      1      539
42
##      black educ hispanic union exper kidlt6      lwage
## 3      0     12      0      0      31      0 1.897120
## 13     0     14      0      0      9      0 2.484907
## 20     0     12      0      0     12      1 2.431418
## 21     0     12      0      0     13      0 1.629241
## 22     0     12      0      0     23      0 2.302585
## 25     0     18      0      0     21      0 2.496741
## 26     0     12      0      0     14      0 2.957511
## 27     0     14      0      0      7      1 1.783791
## 29     0     15      0      0      9      1 1.945910
## 31     0     12      0      0     24      0 2.600836
```

```
rm(trabalhosalarios )
```

1 Fazer as regressões Ridge, Lasso e ElasticNet com a variável dependente "lwage" (salário-hora da esposa em logaritmo neperiano) e todas as demais variáveis da base de dados são variáveis explicativas (todas essas variáveis tentam explicar o salário-hora da esposa). No pdf você deve colocar a rotina utilizada, mostrar em uma tabela as estatísticas dos modelos (RMSE e R2) e concluir qual o melhor modelo entre os três, e mostrar o resultado da predição com intervalos de confiança para os valores acima

1.1) Pre processamento

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

```
set.seed(24)
```

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

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

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

```
# Padronizando a base de treinamento e teste
pre_proc_val <- preProcess(df_train[,cols],
```

```

                                method = c("center", "scale"))
df_train[,cols] = predict(pre_proc_val, df_train[,cols])
df_test[,cols] = predict(pre_proc_val, df_test[,cols])

cols_reg =
c('husage', 'husunion', 'husearns', 'huseduc', 'husblck', 'hushisp', 'hushrs', 'kidg
e6', 'age', 'black', 'educ', 'hispanic', 'union', 'exper', 'kidlt6', 'lwage')

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

## [1] 2061 15

## [1] 513 15

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

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

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

# dependente para o modelo em um objeto chamado "y_test"
y_test = df_test$lwage

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

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

```

```

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

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

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

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

## [1] 0.07943282

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

## Time difference of 0.003395081 secs

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

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

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

```

```

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

##          RMSE    Rsquare
## 1 0.8491541 0.2785873

# Predicao e avaliacao nos dados de teste:
predictions_test <- predict(ridge_reg,
                             s = best_lambda_ridge,
                             newx = x_test)

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

##          RMSE    Rsquare
## 1 0.8239824 0.3019891

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

# O resultado da predicao eh:
predict_our_ridge

##          s1
## [1,] -0.2049218

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

# O resultado eh:
wage_pred_ridge

##          s1
## [1,] 2.086919

exp(wage_pred_ridge) # valor real

##          s1
## [1,] 8.06004

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

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

# Criar um dataframe para armazenar as médias
Ridge_padronizadas <- data.frame(
  Model = 'Ridge Padronizada',
  Lambda = best_lambda_ridge,

```

```

RMSE_treino = eval_results(y_train, predictions_train, df_train)[1],
Rsquare_treino = eval_results(y_train, predictions_train, df_train)[2],
RMSE_teste = eval_results(y_test, predictions_test, df_test)[1],
Rsquare_teste = eval_results(y_test, predictions_test, df_test)[2],
Valor_predito = exp(wage_pred_ridge),
int.conf.inf = exp(CI_lwr_ridge), # esta com o valor padronizado, teria
que ajeitar aqui
int.conf.up = exp(CI_upr_ridge), # esta com o valor padronizado, teria que
ajeitar aqui
stringsAsFactors = FALSE
)

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

```

1.3) Regressao de Lasso :

```

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

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

## [1] 0.01258925

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

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

## 15 x 1 sparse Matrix of class "dgCMatrix"
##              s0
## husage      0.015648453
## husunion -0.050283412
## husearns   0.235385985
## huseduc    0.043459439
## husblk     .
## hushisp     .
## hushrs    -0.063395131
## kidge6    -0.068338201
## age       0.022140728
## black     .
## educ      0.307925589
## hispanic  .
## union     0.412547221
## exper     .
## kidlt6    -0.009068894

# Vamos fazer as predicoes na base de treinamento e
# avaliar a regressao Lasso
predictions_train <- predict(lasso_model,

```

```

s = best_lambda_lasso,
newx = x)

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

##          RMSE    Rsquare
## 1 0.849695 0.2776679

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

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

##          RMSE    Rsquare
## 1 0.8239465 0.3020499

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

predict_our_lasso

##          s1
## [1,] -0.1542719

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

##          s1
## [1,] 2.113432

# Portanto, o salario-hora da esposa eh US$10.36

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

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

```

```

que ajeitar aqui
  int.conf1.up = exp(CIupr_lasso), # esta com o valor padronizado, teria que
  ajeitar aqui
  stringsAsFactors = FALSE
)

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

df_final = rbind(Ridge_padronizadas, lasso_padronizadas)

```

1.4) Regressao ElasticNet

```

# Vamos configurar o treinamento do modelo por
# cross validation, com 10 folders, 5 repeticoes
# e busca aleatoria dos componentes das amostras
# de treinamento, o "verboseIter" eh soh para
# mostrar o processamento.
train_cont <- trainControl(method = "repeatedcv",
                           number = 15,
                           repeats = 8,
                           search = "random",
                           verboseIter = TRUE)

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

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

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

##          lambda
## 10 0.01003417

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

## 15 x 74 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 74 column names 's0', 's1', 's2' ... ]]

##
## husage      0.01789223  0.01789362  0.01789258  0.01789099  0.01790081
## husunion -0.08282503 -0.08299825 -0.08315535 -0.08329887 -0.08342685
## husearns   0.24445625  0.24450319  0.24454558  0.24458417  0.24461696
## huseduc    0.05418411  0.05425786  0.05433344  0.05440530  0.05443463
## husblck    0.08996805  0.09489086  0.09980678  0.10457290  0.10697346
## hushisp    0.14581108  0.14715617  0.14846130  0.14966826  0.15044834
## hushrs     -0.07639606 -0.07646142 -0.07652093 -0.07657504 -0.07662468

```

```
## kidge6      -0.11102771 -0.11124854 -0.11145340 -0.11164070 -0.11179035
## age         0.02355083  0.02357210  0.02359227  0.02361122  0.02362728
## black      -0.05703516 -0.06194916 -0.06683730 -0.07156522 -0.07401934
## educ        0.31351137  0.31352255  0.31352760  0.31353028  0.31355788
## hispanic   -0.08894316 -0.09013586 -0.09127902 -0.09233396 -0.09306831
## union       0.44647457  0.44664467  0.44679799  0.44693677  0.44706763
## exper       .           .           .           .           .
## kidlt6     -0.05320073 -0.05341459 -0.05361361 -0.05379533 -0.05393864

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

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

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

##           RMSE    Rsquare
## 1 0.8493444 0.278264

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

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

##           RMSE    Rsquare
## 1 0.8236324 0.3025818

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

## [1] -0.1682915

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

## [1] 2.106093

# Entao o salario-hora medio da esposa predito com base
# nas caracteristicas informadas eh US$9.28

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

# Criar um dataframe para armazenar as médias
```

```

eladticnet_padronizadas <- data.frame(
  Model = 'ElasticNet Padronizada',
  Lambda = elastic_reg$bestTune[2],
  RMSE_treino = eval_results(y_train, predictions_train, df_train)[1],
  Rsquare_treino = eval_results(y_train, predictions_train, df_train)[2],
  RMSE_teste = eval_results(y_test, predictions_test, df_test)[1],
  Rsquare_teste = eval_results(y_test, predictions_test, df_test)[2],
  Valor_predito = exp(wage_pred_elastic),
  int.conf.inf = exp(CI_lwr_elastic), # esta com o valor padronizado, teria
  que ajustar aqui
  int.conf.up = exp(CI_upr_elastic), # esta com o valor padronizado, teria
  que ajustar aqui
  stringsAsFactors = FALSE
)

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

df_final = rbind(df_final, eladticnet_padronizadas)

```

1.4) Conclusão

df_final

```

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

```

Adicionado Print para melhor visualização dos dados do Data Frame df_final:

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

Fonte: O autor (2025)

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

APÊNDICE 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

Exercício 1

```
<h1>1) Identificador automático de idioma</h1>
```

```
<p><b>Problema</b>: Dados um texto de entrada, é possível identificar em qual
língua o texto está escrito?</p>
```

```
<p>
```

```
Entrada: "texto qualquer" <br />
```

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

```
</p>
```

```
<h5>O processo de Reconhecimento de Padrões</h5>
```

```
<p>O objetivo desse trabalho é demonstrar o processo de "construção de
atributos" e como ele é fundamental para o <b>Reconhecimento de Padrões
(RP)</b>.</p>
```

```
<p>Primeiro um conjunto de "amostras" previamente conhecido
(classificado)</p>
```

```
#
```

```
# amostras de texto em diferentes línguas
```

```
#
```

```
ingles = [
```

```
"Hello, how are you?",
```

```
"I love to read books.",
```

```
"The weather is nice today.",
```

```
"Where is the nearest restaurant?",
```

```
"What time is it?",
```

```
"I enjoy playing soccer.",
```

```
"Can you help me with this?",
```

```
"I'm going to the movies tonight.",
```

```
"This is a beautiful place.",
```

```
"I like listening to music.",
```

```
"Do you speak English?",
```

```

"What is your favorite color?",
"I'm learning to play the guitar.",
"Have a great day!",
"I need to buy some groceries.",
"Let's go for a walk.",
"How was your weekend?",
"I'm excited for the concert.",
"Could you pass me the salt, please?",
"I have a meeting at 2 PM.",
"I'm planning a vacation.",
"She sings beautifully.",
"The cat is sleeping.",
"I want to learn French.",
"I enjoy going to the beach.",
"Where can I find a taxi?",
"I'm sorry for the inconvenience.",
"I'm studying for my exams.",
"I like to cook dinner at home.",
"Do you have any recommendations for restaurants?",
]

```

```

espanhol = [
"Hola, ¿cómo estás?",
"Me encanta leer libros.",
"El clima está agradable hoy.",
"¿Dónde está el restaurante más cercano?",
"¿Qué hora es?",
"Voy al parque todos los días.",
"¿Puedes ayudarme con esto?",
"Me gustaría ir de vacaciones.",
"Este es mi libro favorito.",
"Me gusta bailar salsa.",
"¿Hablas español?",

```

```

"¿Cuál es tu comida favorita?",
"Estoy aprendiendo a tocar el piano.",
"¡Que tengas un buen día!",
"Necesito comprar algunas frutas.",
"Vamos a dar un paseo.",
"¿Cómo estuvo tu fin de semana?",
"Estoy emocionado por el concierto.",
"¿Me pasas la sal, por favor?",
"Tengo una reunión a las 2 PM.",
"Estoy planeando unas vacaciones.",
"Ella canta hermosamente.",
"El perro está jugando.",
"Quiero aprender italiano.",
"Disfruto ir a la playa.",
"¿Dónde puedo encontrar un taxi?",
"Lamento las molestias.",
"Estoy estudiando para mis exámenes.",
"Me gusta cocinar la cena en casa.",
"¿Tienes alguna recomendación de restaurantes?",
]

```

```

portugues = [
"Estou indo para o trabalho agora.",
"Adoro passar tempo com minha família.",
"Preciso comprar leite e pão.",
"Vamos ao cinema no sábado.",
"Gosto de praticar esportes ao ar livre.",
"O trânsito está terrível hoje.",
"A comida estava deliciosa!",
"Você já visitou o Rio de Janeiro?",
"Tenho uma reunião importante amanhã.",
"A festa começa às 20h.",
"Estou cansado depois de um longo dia de trabalho.",

```

```

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

```

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

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

```
import random
```

```
pre_padroes = []
```

```
for frase in ingles:
```

```

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

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

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

random.shuffle(pre_padroes)
#print(pre_padroes)

```

O DataFrame do pandas facilita a visualização.

```

import pandas as pd
nomes_colunas = ['frase', 'idioma']
dados = pd.DataFrame(pre_padroes, columns=nomes_colunas)
dados

```

<h3>Construção dos atributos</h3>

<p>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.<p>

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

```

# Função para verificar se a frase contém "ing"
def contains_ing(text):
    if "ing" in text:
        return 1
    else:
        return 0

# Aplicando a função ao DataFrame
dados['Contem_ing'] = dados['frase'].apply(contains_ing)

```

```

# Função para verificar se a frase contém "oy"
def contains_oy(text):
    if "oy" in text:
        return 1
    else:
        return 0

# Aplicando a função ao DataFrame
dados['Contem_oy'] = dados['frase'].apply(contains_oy)

# Função para verificar se a frase contém "oy"
def contains_carc_especial(text):
    if ";" in text or ":" in text:
        return 1
    else:
        return 0

# Aplicando a função ao DataFrame
dados['Contem_carc_especial'] = dados['frase'].apply(contains_carc_especial)

dados.head(2)

##### Listas de palavras comuns em
cada idioma #####

ingles = set(["the", "is", "in", "and", "to", "a", "it", "you", "of", "for",
"on", "with", "this", "that", "by", "an", "be", "are", "was", "as"])

portugues = set(["o", "é", "em", "e", "para", "um", "ele", "você", "de",
"por", "com", "isso", "que", "um", "ser", "são", "foi", "como"])

espanhol = set(["el", "al", "la", "es", "en", "y", "para", "un", "él", "tú",
"de", "por", "con", "esto", "que", "un", "ser", "son", "fue", "como"])

def count_palavras(texto, palavras):
    words = texto.lower().split()
    count = sum(1 for word in words if word in palavras)

```

```

    return count

def detectar_lingua(text):
    ingles_count = count_palavras(text, ingles)
    portugues_count = count_palavras(text, portugues)
    espanhol_count = count_palavras(text, espanhol)

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

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

#####
#####

dados2 = dados.copy()

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

def tamanhoMedioFrases(texto):
    palavras = re.split("\s", texto)

    #print(palavras)

    tamanhos = [len(s) for s in palavras if len(s)>0]

    #print(tamanhos)

    soma = 0

    for t in tamanhos:

```

```

        soma=soma+t

    return soma / len(tamanhos)

def funcao2(frase):

    # faz alguma coisa, mede algo

    return 0

##### GABRIEL
#####

def extraiCaracteristicas(frase):

    # frase é um vetor [ 'texto', 'lingua' ]

    texto = frase[0]

    pattern_regex = re.compile('[^\w+]', re.UNICODE)

    texto = re.sub(pattern_regex, ' ', texto)

    #print(texto)

    caracteristica1=tamanhoMedioFrases(texto)

    caracteristica2=funcao2(texto)

    caracteristica3=funcao2(texto)

    # acrescente as suas funcoes no vetor padrao

    padrao = [caracteristica1, caracteristica2, caracteristica3, frase[1] ]

    return padrao

def geraPadroes(frases):

    padroes = []

    for frase in frases:

        padrao = extraiCaracteristicas(frase)

        padroes.append(padrao)

    return padroes

```

```

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

dados = pd.DataFrame(padroes)
dados

# Lista com os novos nomes das colunas
novos_nomes_colunas = ['cont_media', 'caracteristicas', 'padroes','idioma']

# Renomeando todas as colunas
dados.columns = novos_nomes_colunas

dados.head(2)

dados2.columns

##### Valerio

def freqCaracteres(texto, lower=True):
    freq = {}
    for caracter in texto:
        if lower:
            caracter = caracter.lower()
        key = f'freq_{caracter}'
        freq[key] = freq.get(key, 0) + 1
    return freq

def freqCaracteres_df(textos, lower=True):
    df = pd.DataFrame()
    for texto in textos:
        texto = texto[0]

```

```

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

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

    return df.fillna(0)

# começa com pontuação?
def começa_pontuacao(frase):
    if frase[0].strip() in ['¿', '¡']:
        return 1
    else:
        return 0

#encontros vocálicos e consonantais e outros padrões como ing.
def encontros(frase):
    lista_padroes =
["ch", "nh", "th", "sh", "ll", "rr", 'lh', 'nn', 'tr', 'ss', 'ie', 'ing', 'el']

    res = []

    for c in lista_padroes:
        res.append(frase.lower().count(c.lower()))

    return res

#base para a contagem ocorrência das diversas classes de caracteres
def qtde(frase, lista_caracteres):
    qtde = 0

    for c in frase.lower():
        if c.lower() in lista_caracteres:
            qtde = qtde + 1

    return qtde

#vogais
def qtde_vogais(frase):
    lista = ['a', 'e', 'i', 'o', 'u']

    return qtde(frase, lista)

```

```

#pontuacao
def qtde_pontuacao(frase):
    lista = ['.', ',', '?', '!']
    return qtdes(frase, lista)

#pontuacao especial
def qtde_pontuacao_especial(frase):
    lista = [';', ':']
    return qtdes(frase, lista)

#acentuados
def qtde_acentuados(frase):
    lista = [u'á', u'é', u'í', u'ó', u'ú', u'ã', u'â', u'ê', u'õ', u'ô']
    return qtdes(frase, lista)

#acentuados especiais
def qtde_acentuados_especiais(frase):
    lista = [u'ñ']
    return qtdes(frase, lista)

#consoantes
def qtde_consoantes(frase):
    lista =
['b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x',
'y', 'z']
    return qtdes(frase, lista)

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

```

```

for t in tamanhos:
    soma=soma+t
return soma / len(tamanhos)

def extraiCaracteristicas(frase):
    # frase é um vetor [ 'texto', 'lingua' ]
    texto = frase[0]
    caracteristicas = []
    caracteristicas = [comeca_pontuacao(texto)]
    pattern_regex = re.compile('[^\w+]', re.UNICODE)
    texto = re.sub(pattern_regex, ' ', texto)
    #print(texto)
    caracteristica1=tamanhoMedioFrases(texto)
    # acrescente as suas funcoes no vetor padrao
    caracteristicas = caracteristicas + [qtde_vogais(texto),
        qtde_pontuacao(texto),
        qtde_acentuados(texto),
        qtde_acentuados_especiais(texto),
        qtde_consoantes(texto)]
    caracteristicas = caracteristicas + encontrados(texto)
    padrao = caracteristicas+[caracteristica1, frase[1] ]
    return padrao

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

# converte o formato [frase classe] em

```

```

# [caracteristica_1, caracteristica_2,... caracteristica n, classe]
padroes = geraPadroes(pre_padroes)

#
# apenas para visualizacao
#print(padroes)

dados = pd.DataFrame(padroes)

#remove coluna com a target e depois reinsere como última coluna
ultima_coluna = dados.columns[-1]
df_target = dados[ultima_coluna]

dados =
dados[dados.columns[0:-1]].join(freqCaracteres_df(pre_padroes).reset_index())
dados = dados.join(df_target)

dados.columns = dados.columns.astype(str)
padroes = dados.values.tolist()
dados

dados2.head(2)

# juntando com a base que eu tenho inicial
dados = pd.concat([dados,dados2[['Contem_ing', 'Contem_oy',
'Contem_carc_especial',
'detector_lingua']] ], axis = 1)
dados

dados.columns

<h2>Treinando o modelo com SVM</h1>

<p>Separando o conjunto de treinamento do conjunto de testes</p>

vet = np.array(dados)

```

```

vet

from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report
import numpy as np

def treinaModelo(padroes):

    vet = np.array(padroes)

    classes = padroes['20']          # classes = [p[-1] for p in padroes]

    #print(classes)

    padroes_sem_classe = padroes.drop(['20'], axis = 1 )#np.delete(vet, 3,
axis=1)

    #print(padroes_sem_classe)

    X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe,
classes, test_size=0.25, stratify=classes)

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

    #

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

    #

    # melhor avaliar com a matriz de confusão
    y_pred = modelo.predict(X_train)
    print('_'*60)
    print('classes: ',modelo.classes_)

    cm = confusion_matrix(y_train, y_pred, labels=modelo.classes_)
    print(cm)

```

```

print(classification_report(y_train, y_pred))

ConfusionMatrixDisplay(confusion_matrix=cm,
                        display_labels=modelo.classes_).plot()

#

# com dados de teste que não foram usados no treinamento
print('métricas mais confiáveis')

acuracia = modelo.score(X_test, y_test)
print("Acurácia nos dados de teste: {:.2f}%".format(acuracia * 100))

y_pred2 = modelo.predict(X_test)
print('_'*60)

print('classes: ',modelo.classes_)

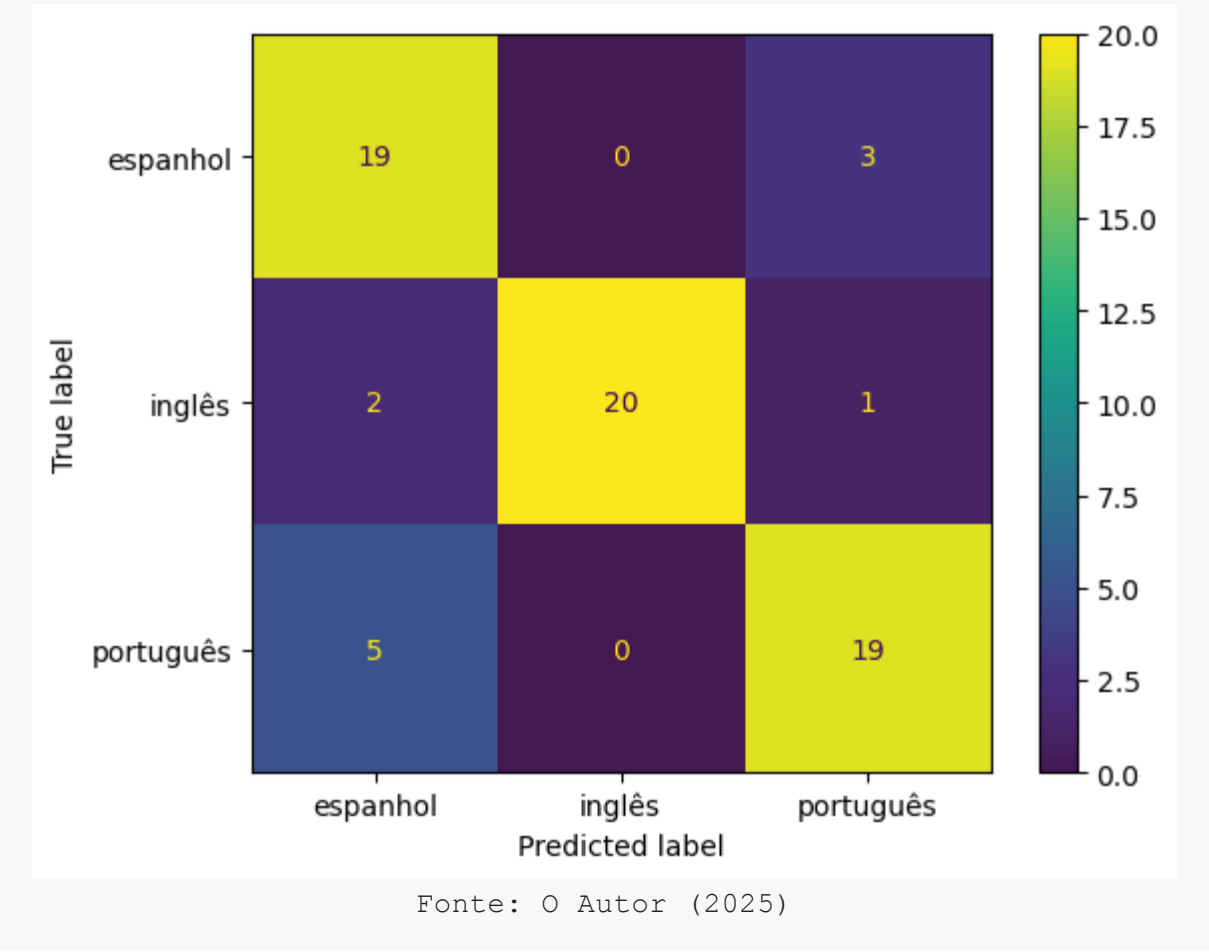
cm = confusion_matrix(y_test, y_pred2, labels=modelo.classes_)
print(cm)

print(classification_report(y_test, y_pred2))

ConfusionMatrixDisplay(confusion_matrix=cm,
                        display_labels=modelo.classes_).plot()

treinaModelo(dados)

```



Exercicio 2

<h1>2) Melhore uma base de dados ruim</h1>

<p>Problema: Pegar um dataset que esta com uma classificação ruim e fazer pre processamento para melhora-lo.</p>

<p>

<p>Objetivo: Se o percentual da

acurácia original estiver em até 85%, a meta será obter 5%. Para bases com mais de 90% de acurácia, a

meta será obter a melhora em pelo menos 2 pontos percentuais (92% ou mais .

Nessa atividade deverá ser entregue o script aplicado (o notebook e o DF correspondente</p>

```

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

!pip install ucimlrepo matplotlib matplotlib

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

#!pip install ucimlrepo
from ucimlrepo import fetch_ucirepo

import sklearn

import numpy as np

from sklearn.preprocessing import OneHotEncoder, LabelEncoder,
OrdinalEncoder, MinMaxScaler, StandardScaler, MaxAbsScaler

from sklearn.model_selection import train_test_split

from sklearn import svm

from sklearn.metrics import confusion_matrix, classification_report
, accuracy_score

import warnings

warnings.filterwarnings("ignore")

# lendo a base de dados da UCI
# fetch dataset
adult = fetch_ucirepo(id=2)

# data (as pandas dataframes)
X = adult.data.features
y = adult.data.targets

# juntando a base de dados
df = pd.concat([X,y], axis = 1)

del X

```

```

del y
df.head(3)

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

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

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

    return X_train, X_test, y_train, y_test

def print_metricas2(modelo, X_train, X_test, y_train, y_test):
    # score com os dados de treinamento
    acuracia = modelo.score(X_train, y_train)
    print("Acurácia nos dados de treinamento: {:.2f}%".format(acuracia * 100))

    #
    # melhor avaliar com a matriz de confusão
    y_pred = modelo.predict(X_train)
    print('_'*60)
    print('classes: ',modelo.classes_)

    cm = confusion_matrix(y_train, y_pred, labels=modelo.classes_)

```

```

print(cm)

print(classification_report(y_train, y_pred))

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

def get_acuracia(cm):
    total = sum(sum(cm)) # Soma de todos os elementos da matriz
    acertos = sum(np.diag(cm)) # Soma dos elementos da diagonal principal
    (verdadeiros positivos e verdadeiros negativos)
    acuracia = acertos / total
    return acuracia

def get_metricas(modelo, X_train, X_test, y_train, y_test):
    # Obtendo métricas sobre o conjunto de treinamento
    y_pred = modelo.predict(X_train)
    cm_train = confusion_matrix(y_train, y_pred, labels=modelo.classes_)
    cr_train = classification_report(y_train, y_pred)
    acuracia_train = get_acuracia(cm_train)

    # Obtendo métricas sobre o conjunto de teste
    y_pred2 = modelo.predict(X_test)
    cm_test = confusion_matrix(y_test, y_pred2, labels=modelo.classes_)
    cr_test = classification_report(y_test, y_pred2)
    acuracia_test = get_acuracia(cm_test)

```

```

return {'modelo' : modelo,
        'acuracia_train': acuracia_train,
        'acuracia_test': acuracia_test,
        'cm_train' : cm_train,
        'cm_test' : cm_test,
        'cr_train' : cr_train,
        'cr_test' : cr_test
    }

def print_metricas(metricas):
    #metricas = get_metricas(modelo, X_train, X_test, y_train, y_test)

    print('***** TREINAMENTO *****')

    print('Acurácia nos dados de treinamento:
{:.2f}%'.format(metricas['acuracia_train'] * 100))

    print('Confusion Matrix nos dados de treinamento')
    print(metricas['cm_train'])

    print('Classification Report nos dados de treinamento')
    print(metricas['cr_train'])

    print('***** TESTE *****')

    print('Acurácia nos dados de teste:
{:.2f}%'.format(metricas['acuracia_test'] * 100))

    print('Confusion Matrix nos dados de teste')
    print(metricas['cm_test'])

    print('Classification Report nos dados de teste')
    print(metricas['cr_test'])

    print('***** RESUMO *****')

    print('Acurácia nos dados de treinamento:
{:.2f}%'.format(metricas['acuracia_train'] * 100))

    print('Acurácia nos dados de teste:
{:.2f}%'.format(metricas['acuracia_test'] * 100))

#Separa df em X e Y
def preprocess(df, fillna=True):
    #nulls (inicialmente colocar -1 no lugar)

```

```

    if fillna:
        df = df.fillna('?')
    target = 'income'
    X = df.drop(target, axis=1)
    Y = df[target]
    return X, Y

def get_features_categoricas(X):
    features_categoricas = [coluna for coluna in
X.select_dtypes(include=['object']).columns ]
    return features_categoricas

#Aplica LabelEncoder nas features categóricas
def ordinal_encoder(X):
    enc = OrdinalEncoder()

    features_categoricas = get_features_categoricas(X)

    df_encoded = pd.DataFrame(enc.fit_transform(X[features_categoricas]),
columns=enc.get_feature_names_out(features_categoricas))

    df_final = pd.concat([X.drop(columns=features_categoricas), df_encoded],
axis=1)

    return df_final

    enc.fit(X)

    return enc.transform(X)

def ordinal_encoder_completo(X):
    enc = OrdinalEncoder()

    enc.fit(X)

    return enc.transform(X)

def one_hot_encoder_completo(X):
    enc = OneHotEncoder()

    enc.fit(X)

```

```

    return enc.transform(X)

def one_hot_encoder(X):
    features_categoricas = get_features_categoricas(X)
    enc = OneHotEncoder(sparse_output=False)
    df_encoded = pd.DataFrame(enc.fit_transform(X[features_categoricas]),
                              columns=enc.get_feature_names_out(features_categoricas))
    df_final = pd.concat([X.drop(columns=features_categoricas), df_encoded],
                          axis=1)
    return df_final

df.isnull().sum()

colunas_com_nulls = list(df.columns[df.isnull().sum()>0]) # pega quem nao
esta nulo

colunas_com_nulls

# Verificando quantidade de valores únicos e tipos de cada coluna
[[c, df[c].nunique(), df[c].dtype] for c in df.columns]

# Inicializado Dataframe que armazenará os resultados dos experimentos
df_resultados_geral = pd.DataFrame(columns=['descricao', 'modelo',
'acuracia_train', 'acuracia_test', 'cm_train', 'cm_test', 'cr_train',
'cr_test'])

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

# o que estiver nulo, preenche com a media
for c in colunas_com_nulls:
    print(c)
    X[c] = X[[c]].fillna(X[c].mean())

### 1 Inicial - Somente OrdinalEncoder em todos campos

```

```

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

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

df_resultados_geral

### 1.1 Inicial - Somente OrdinalEncoder nos campos categóricos

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

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

```

```

df_resultados_geral

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

# ajeitando a variavel resposta
df['income'].str.replace(".", " ", regex=False).value_counts().index

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

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

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

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

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

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

df_resultados_geral

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

```

```

# Vamos ver o efeito da aplicação de one-hot-encode no lugar de labelencoder
descricao = '2.1. one-hot-encode completo no lugar de labelencoder'

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

df_resultados_geral

### 2.1.1 one-hot-encode completo com MaxAbsScaler

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

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

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

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

modelo = train(X_train, y_train)

```

```

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

df_resultados_geral

### 2.2 MinMaxScaler sobre ordinalEncoder

#Aplicando MinMaxScaler sobre ordinalEncoder

descricao = '2.2 MinMaxScaler sobre ordinalEncoder'

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

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

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

```

```

df_resultados_geral

### 2.2.1 MinMaxScaler sobre ordinalEncoder - Dropando colunas

#Vamos veriricar o efeito de dropar as colunas com nulos.
#Aplicando MinMaxScaler sobre ordinalEncoder

descricao = '2.2.1 MinMaxScaler sobre ordinalEncoder - Dropando colunas'

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

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

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

df_resultados_geral

### 2.3 one-hot-encode somente em categoricas

```

```

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

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

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

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

df_resultados_geral

### Aplicando MinMaxScaler sobre one_hot_encoder

#Aplicando MinMaxScaler sobre one_hot_encoder

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

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

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

```

```

scaler = MinMaxScaler()

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

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

metricas['descricao'] = descricao

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

print_metricas(metricas)

df_resultados_geral

### Final - Avaliando os modelos

# Print the DataFrame with full descriptions
pd.set_option('display.max_colwidth', None)

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

```

descricao	acuracia_train	acuracia_test	
0	1 Inicial - Somente OrdinalEncoder em todos campos	0.506129	0.506101
1	1.1 Inicial - Somente OrdinalEncoder nos campos categóricos	0.529688	0.529604
2	2. Modelo com ajustes na target que estava com problemas (4 categorias)	0.797166	0.801163
3	2.1. one-hot-encode completo no lugar de labelencoder	0.887609	0.867087

4	2.1.1 one-hot-encode completo com MaxAbsScaler	0.887609	0.867087
5	2.2 MinMaxScaler sobre ordinalEncoder	0.847670	0.850381
6	2.2.1 MinMaxScaler sobre ordinalEncoder - Dropando colunas	0.847452	0.849562
7	2.3 one-hot-encode somente em categoricas	0.797166	0.800753
8	2.3.1 one-hot-encode somente em categoricas - com minmaxscaler	0.846278	0.841454

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

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

CLASSIFICAÇÃO

Para o experimento de Classificação:
Ordenar pela Acurácia (descendente), ou seja, a técnica de melhor acurácia ficará em primeiro na tabela.

Após o quadro colocar:

- Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia (criar um arquivo com novos casos à sua escolha)
- A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM - Hold-out	C=100 Sigma=0.01	82,35%	<pre>Prediction bus opel saab van bus 40 0 0 1 opel 0 31 3 1 saab 0 24 30 0 van 1 0 0 39</pre> Fonte: O Autor (2025)
SVM - CV	C=100 Sigma=0.015	81,76%	<pre>Prediction bus opel saab van bus 40 1 0 1 opel 0 33 4 2 saab 0 20 29 1 van 1 1 0 37</pre> Fonte: O Autor (2025)
RNA - CV	size=41 decay=0.7	81,76%	<pre>Prediction bus opel saab van bus 40 1 0 1 opel 0 32 6 0 saab 0 18 27 0 van 1 4 0 40</pre> Fonte: O Autor (2025)
RNA - Hold-out	size=41 decay=0.7	74,71%	<pre>Prediction bus opel saab van bus 39 1 0 1 opel 0 28 13 0 saab 1 24 20 0 van 1 2 0 40</pre> Fonte: O Autor (2025)
RF - Hold-out	mtry=2	72,94%	<pre>Prediction bus opel saab van bus 41 1 0 0 opel 0 19 8 0 saab 0 33 24 1 van 0 2 1 40</pre> Fonte: O Autor (2025)
RF - CV	mtry=5	72,35%	

			Prediction bus opel saab van bus 41 0 1 0 opel 0 19 9 0 saab 0 34 23 1 van 0 2 0 40 Fonte: O Autor (2025)
KNN	k= 1	61,76%	Prediction bus opel saab van bus 38 4 3 3 opel 3 15 11 2 saab 0 32 18 2 van 0 4 1 34 Fonte: O Autor (2025)

Resultado da predição com os dados novos:

Comp	Circ	DCirc	RadRa	PrAxisRa	MastRa	ScatRa	Elong	PrAxisRect	MastRect	ScVarMast	ScVarMastis	RaGyr	SkewMast	SkewMastis	KurtMast	KurtMastis	HolRa	predicao
65	34	88	120	61	10	156	40	21	140	170	320	167	67	4	8	190	200	bus
99	55	73	180	59	6	180	51	17	139	190	420	200	77	10	12	188	191	opel
113	51	91	190	67	9	233	43	24	161	200	550	189	73	7	20	191	199	bus

Fonte: O Autor (2025)

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM - Hold-out	C=0.01 Sigma=1	79,22%	Prediction neg pos neg 86 25 pos 7 36 Fonte: O Autor (2025)
SVM - CV	C=0.01 Sigma=1	79,22%	Prediction neg pos neg 86 25 pos 7 36 Fonte: O Autor (2025)
RF - CV	mtry=2	78,57%	Prediction neg pos neg 82 22 pos 11 39 Fonte: O Autor (2025)
RF - Hold-out	mtry=2	77,92%	Prediction neg pos neg 83 24 pos 10 37 Fonte: O Autor (2025)
RNA - CV	size=21 decay=0.4	73,38%	Prediction neg pos neg 79 27 pos 14 34 Fonte: O Autor (2025)
KNN	k= 9	72,08%	Prediction neg pos neg 77 27 pos 16 34 Fonte: O Autor (2025)
RNA - Hold-out	size=31 decay=0.7	70,13%	

			Prediction neg pos neg 76 29 pos 17 32 Fonte: O Autor (2025)
--	--	--	--

Resultado da predição com os dados novos:

preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	predicao
2	100	59	25	90	36	0.378	27	neg
4	127	80	30	120	40	0.500	40	neg
5	151	91	27	480	32	0.470	55	pos

Fonte: O Autor (2025)

REGRESSÃO

Para o experimento de Regressão:

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

Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM - Hold-out	C=2 Sigma=0.01	82,4%	6,4%	91,0%		4,2%
SVM - CV	C=10 Sigma=0.01	82,3%	6,4%	90,9%		4,2%
RNA - CV	size=41 decay=0.1	81,6%	6,6%	90,5%		4,5%
RF - CV	mtry=2	81,3%	6,6%	90,4%		4,6%
RF - Hold-out	mtry=2	81,3%	6,6%	90,4%		4,6%
RNA - Hold-out	size=11 decay=0.1	77,4%	7,3%	88,4%		5,3%
KNN	K=9	73,6%	7,9%	85,9%		5,8%

Resultado da predição com os dados novos:

GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predicao
300	100		3	2	4	7.2	0.00 0.5266368
390	109		1	4	2	8.8	0.50 0.7872084
344	103		5	4	3	8.3	0.67 0.7569174

Fonte: O Autor (2025)

Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
KNN	K=1	0,90595	419,072	0,96275		136,614
RNA - Hold-out	size=21 decay=0.7	0,89035	452,507	0,95497		143,813
RNA - CV	size=31 decay=0.7	0,8107	594,555	0,90729		222,924

SVM - Hold-out	C=100 Sigma=0.01	0,8107	594,55 5	0,9072 9		222,92 4
SVM - CV	C=100 Sigma=0.01	0,7900 4	626,15 4	0,9038 4		195,57 2
RF - Hold-out	mtry=2	0,6480 9	810,64 8	0,8053 6		261,61 9
RF - CV	mtry=2	-0,483 6	1664,4 8	0,3307		460,41 2

Resultado da predição com os dados novos:

```
dap h Me predicao
16 10 0.70 99.80
20 19 0.34 210.31
19 14 0.43 154.67
```

Fonte: O Autor (2025)

AGRUPAMENTO

Veículo

Lista de Clusters gerados:

10 primeiras linhas do arquivo com o cluster correspondente.

Usa 10 clusters no experimento.

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

REGRAS DE ASSOCIAÇÃO

Musculação

Regras geradas com uma configuração de Suporte e Confiança.

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

Resolução:

```
library(caret)
library(nnet)
library(randomForest)
run_classification <- function(dados, response_var) {

  # Separando em treino e teste a base de dados
  set.seed(202485)
  ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
  treino <- dados[ran,]
  teste <- dados[-ran,]

  results <- list()

  ##### 1- KNN #####
  #print('Avaliando o modelo KNN:')
  cat('----- Avaliando o modelo KNN:\n\n' )

  tuneGrid_knn <- expand.grid(k = c(1,3,5,7,9))
  set.seed(202485)
  formula <- as.formula(paste(response_var, "~ ."))
```

```

knn <- train(formula, data = treino, method = "knn", tuneGrid =
tuneGrid_knn)
best_k <- knn$bestTune$k
cat(paste("Melhor K para o modelo KNN:", best_k ))
cat('\n\n')

predict_knn <- predict(knn, teste)
cm_knn <- confusionMatrix(predict_knn, as.factor(teste[[response_var]]))
print(cm_knn)
results$KNN <- list(Model = knn, BestK = best_k, ConfusionMatrix = cm_knn)

##### 2- RNA #####
cat('\n\n')
cat('----- Avaliando o modelo RNA:\n\n')

grid_rna <- expand.grid(size = seq(from = 1, to = 45, by = 10), decay =
seq(from = 0.1, to = 0.9, by = 0.3))

# Hold-out
cat('\n\n')
cat('##### Modelo RNA Hold-out:\n\n')
set.seed(202485)
rna_ho <- train(formula, data = treino, method = "nnet", trace = FALSE,
tuneGrid = grid_rna)
print(rna_ho)

predict_rna_ho <- predict(rna_ho, teste)
cm_rna_ho <- confusionMatrix(predict_rna_ho,
as.factor(teste[[response_var]]))
print(cm_rna_ho)
results$RNA_HO <- list(Model = rna_ho, ConfusionMatrix = cm_rna_ho)

# Cross Validation
cat('\n\n')
cat('##### Modelo RNA Cross Validation: \n\n')
ctrl_cv <- trainControl(method = "cv", number = 10)
set.seed(202485)
rna_cv <- train(formula, data = treino, method = "nnet", tuneGrid =
grid_rna, trControl = ctrl_cv, maxit = 2000, trace = FALSE)
print(rna_cv)

predict_rna_cv <- predict(rna_cv, teste)
cm_rna_cv <- confusionMatrix(predict_rna_cv,
as.factor(teste[[response_var]]))
print(cm_rna_cv)
results$RNA_CV <- list(Model = rna_cv, ConfusionMatrix = cm_rna_cv)

##### 3- SVM #####
cat('\n\n')
cat('----- Avaliando o modelo SVM:\n\n')

tuneGrid_svm <- expand.grid(C = c(1, 2, 10, 50, 100), sigma = c(.01, .015,
0.2))

```

```

# Hold-out
cat('\n\n')
cat('##### Modelo SVM Hold-out:\n\n')
set.seed(202485)
svm_ho <- train(formula, data = treino, method = "svmRadial", tuneGrid =
tuneGrid_svm)
print(svm_ho)

predict_svm_ho <- predict(svm_ho, teste)
cm_svm_ho <- confusionMatrix(predict_svm_ho,
as.factor(teste[[response_var]]))
print(cm_svm_ho)
results$SVM_HO <- list(Model = svm_ho, ConfusionMatrix = cm_svm_ho)

# Cross Validation
cat('\n\n')
cat('##### Modelo SVM Cross Validation:\n\n')
set.seed(202485)
svm_cv <- train(formula, data = treino, method = "svmRadial", trControl =
ctrl_cv, tuneGrid = tuneGrid_svm)
print(svm_cv)

predict_svm_cv <- predict(svm_cv, teste)
cm_svm_cv <- confusionMatrix(predict_svm_cv,
as.factor(teste[[response_var]]))
print(cm_svm_cv)
results$SVM_CV <- list(Model = svm_cv, ConfusionMatrix = cm_svm_cv)

##### 4- Random Forest #####
cat('\n\n')
cat('----- Avaliando o modelo Random Forest (RF):\n\n')

tuneGrid_rf <- expand.grid(mtry = c(2, 5, 7, 9))

# Hold-out
cat('\n\n')
cat('##### Modelo RF Hold-out:\n\n')
set.seed(202485)
rf_ho <- train(formula, data = treino, method = "rf", tuneGrid =
tuneGrid_rf)
print(rf_ho)

predict_rf_ho <- predict(rf_ho, teste)
cm_rf_ho <- confusionMatrix(predict_rf_ho,
as.factor(teste[[response_var]]))
print(cm_rf_ho)
results$RF_HO <- list(Model = rf_ho, ConfusionMatrix = cm_rf_ho)

# Cross Validation
cat('\n\n')
cat('##### Modelo RF Cross Validation:\n\n')
set.seed(202485)

```

```

    rf_cv <- train(formula, data = treino, method = "rf", trControl = ctrl_cv,
tuneGrid = tuneGrid_rf)
    print(rf_cv)

    predict_rf_cv <- predict(rf_cv, teste)
    cm_rf_cv <- confusionMatrix(predict_rf_cv,
as.factor(teste[[response_var]]))
    print(cm_rf_cv)
    results$RF_CV <- list(Model = rf_cv, ConfusionMatrix = cm_rf_cv)

    return(results)
}
# Classificação - Veículos -----
### Obter os dados
setwd("C:/Users/AlmeidaGabriel(BipGr/Desktop/outros/Pos/08. Aprendizado de
Máquina/Bases de Dados/06 - Veículos")
df <- read.csv("6 - Veiculos - Dados.csv")
df$a <- NULL # pois e o numero de linha, nao faz sentido para modelagem
resultados <- run_classification(df, "tipo")
#---- fazendo a predição
summary(df)
# Criando o data frame diretamente com as variáveis definidas
df_novo <- data.frame(
  Comp = c(65, 99, 113),
  Circ = c(34, 55, 51),
  DCirc = c(88, 73, 91),
  RadRa = c(120, 180, 190),
  PrAxisRa = c(61, 59, 67),
  MaxLRa = c(10, 6, 9),
  ScatRa = c(156, 180, 233),
  Elong = c(40, 51, 43),
  PrAxisRect = c(21, 17, 24),
  MaxLRect = c(140, 139, 161),
  ScVarMaxis = c(170, 190, 200),
  ScVarmaxis = c(320, 420, 550),
  RaGyr = c(167, 200, 189),
  SkewMaxis = c(67, 77, 73),
  Skewmaxis = c(4, 10, 7),
  Kurtmaxis = c(8, 12, 20),
  KurtMaxis = c(190, 188, 191),
  HollRa = c(200, 191, 199)
)
# aplicando o modelo
predicao <- predict(predict_svm_ho, df_novo) # esta dizendo que nao foi
encotrado o modelo, pode ser por nao ficar salvo e termos que fazer a
modelagem
resultado <- cbind(df_novo, predicao)
#resultado$Class <- NULL
View(resultado)
# Classificação - Diabetes -----
### Obter os dados
setwd("C:/Users/AlmeidaGabriel(BipGr/Desktop/outros/Pos/08. Aprendizado de
Máquina/Bases de Dados/10 - Diabetes")

```

```

df <- read.csv("10 - Diabetes - Dados.csv")
df$num <- NULL # pois e o numero de linha, nao faz sentido para modelagem
resultados <- run_classification(df, "diabetes")
#---- fazendo a predição
summary(df)
# Criando o data frame diretamente com as variáveis definidas
df_novo <- data.frame(
  preg0nt = c(2, 4, 5),
  glucose = c(100, 127, 151),
  pressure = c(59, 80, 91),
  triceps = c(25, 30, 27),
  insulin = c(90, 120, 480),
  mass = c(36, 40, 32),
  pedigree = c(0.378, 0.5, 0.47),
  age = c(27, 40, 55)
)
# aplicando o modelo
predicao <- predict(predict_svm_ho, df_novo) # esta dizendo que nao foi
encotrado o modelo, pode ser por nao ficar salvo e termos que fazer a
modelagem
resultado <- cbind(df_novo, predicao)
#resultado$Class <- NULL
View(resultado)
resultados <- run_classification(df, "tipo")
##### REGRESSAO
run_regression <- function(dados, response_var) {

  # Separando em treino e teste a base de dados
  set.seed(202485)
  ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
  treino <- dados[ran,]
  teste <- dados[-ran,]

  results <- list()

  ### Funcoes para avaliar o modelo

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

  #MAE
  # Função MAE
  MAE <- function(actual, predicted) {
    mean(abs(actual - predicted))}

  # Coeficiente de Pearson
  Pearson <- function(x, y) {
    cor(x, y, method = "pearson")}

  #Syx - Erro padrão da Estimativa
  Syx <- function(actual, predicted, df) {

```

```

n <- length(actual)
p <- ncol(df) - 1# Número de colunas do data frame, que corresponde ao
número de parâmetros no modelo
residuals <- actual - predicted
sqrt(sum(residuals^2) / (n - p))

##### 1- KNN #####
#print('Avaliando o modelo KNN:')
cat('----- Avaliando o modelo KNN:\n\n' )

tuneGrid_knn <- expand.grid(k = c(1,3,5,7,9))
set.seed(202485)
formula <- as.formula(paste(response_var, "~ ."))
knn <- train(formula, data = treino, method = "knn", tuneGrid =
tuneGrid_knn)
best_k <- knn$bestTune$k
cat(paste("Melhor K para o modelo KNN:", best_k ))
cat('\n\n')

predict_knn <- predict(knn, teste)

# avaliando o modelo

cat('----- R^2:\n\n' )
print( r2(predict_knn,teste[[response_var]]) )

print(paste("O erro padrão da estimativa (Syx) é:",
Syx(teste[[response_var]], predict_knn, teste)))

cat(paste("O coeficiente de correlação de Pearson é:",
Pearson(teste[[response_var]], predict_knn)))

cat('----- RMSE:\n\n' )
print( rmse(teste[[response_var]], predict_knn) )

cat(paste("O valor do MAE é:", MAE(teste[[response_var]], predict_knn)))

##### 2- RNA #####
cat('\n\n')
cat('----- Avaliando o modelo RNA:\n\n')

grid_rna <- expand.grid(size = seq(from = 1, to = 45, by = 10), decay =
seq(from = 0.1, to = 0.9, by = 0.3))

# Hold-out
cat('\n\n')
cat('##### Modelo RNA Hold-out:\n\n')
set.seed(202485)
rna_ho <- train(formula, data = treino, method = "nnet", trace = FALSE,
tuneGrid = grid_rna)
print(rna_ho)

```

```

predict_rna_ho <- predict(rna_ho, teste)
cm_rna_ho <- confusionMatrix(predict_rna_ho,
as.factor(teste[[response_var]]))
print(cm_rna_ho)
results$RNA_HO <- list(Model = rna_ho, ConfusionMatrix = cm_rna_ho)

# Cross Validation
cat('\n\n')
cat('##### Modelo RNA Cross Validation: \n\n')
ctrl_cv <- trainControl(method = "cv", number = 10)
set.seed(202485)
rna_cv <- train(formula, data = treino, method = "nnet", tuneGrid =
grid_rna, trControl = ctrl_cv, maxit = 2000, trace = FALSE)
print(rna_cv)

predict_rna_cv <- predict(rna_cv, teste)
cm_rna_cv <- confusionMatrix(predict_rna_cv,
as.factor(teste[[response_var]]))
print(cm_rna_cv)
results$RNA_CV <- list(Model = rna_cv, ConfusionMatrix = cm_rna_cv)

##### 3- SVM #####
cat('\n\n')
cat('----- Avaliando o modelo SVM:\n\n')

tuneGrid_svm <- expand.grid(C = c(1, 2, 10, 50, 100), sigma = c(.01, .015,
0.2))

# Hold-out
cat('\n\n')
cat('##### Modelo SVM Hold-out:\n\n')
set.seed(202485)
svm_ho <- train(formula, data = treino, method = "svmRadial", tuneGrid =
tuneGrid_svm)
print(svm_ho)

predict_svm_ho <- predict(svm_ho, teste)
cm_svm_ho <- confusionMatrix(predict_svm_ho,
as.factor(teste[[response_var]]))
print(cm_svm_ho)
results$SVM_HO <- list(Model = svm_ho, ConfusionMatrix = cm_svm_ho)

# Cross Validation
cat('\n\n')
cat('##### Modelo SVM Cross Validation:\n\n')
set.seed(202485)
svm_cv <- train(formula, data = treino, method = "svmRadial", trControl =
ctrl_cv, tuneGrid = tuneGrid_svm)
print(svm_cv)

predict_svm_cv <- predict(svm_cv, teste)

```

```

    cm_svm_cv <- confusionMatrix(predict_svm_cv,
as.factor(teste[[response_var]]))
    print(cm_svm_cv)
    results$SVM_CV <- list(Model = svm_cv, ConfusionMatrix = cm_svm_cv)

##### 4- Random Forest #####
cat('\n\n')
cat('----- Avaliando o modelo Random Forest (RF):\n\n')

tuneGrid_rf <- expand.grid(mtry = c(2, 5, 7, 9))

# Hold-out
cat('\n\n')
cat('##### Modelo RF Hold-out:\n\n')
set.seed(202485)
rf_ho <- train(formula, data = treino, method = "rf", tuneGrid =
tuneGrid_rf)
print(rf_ho)

predict_rf_ho <- predict(rf_ho, teste)
cm_rf_ho <- confusionMatrix(predict_rf_ho,
as.factor(teste[[response_var]]))
print(cm_rf_ho)
results$RF_HO <- list(Model = rf_ho, ConfusionMatrix = cm_rf_ho)

# Cross Validation
cat('\n\n')
cat('##### Modelo RF Cross Validation:\n\n')
set.seed(202485)
rf_cv <- train(formula, data = treino, method = "rf", trControl = ctrl_cv,
tuneGrid = tuneGrid_rf)
print(rf_cv)

predict_rf_cv <- predict(rf_cv, teste)
cm_rf_cv <- confusionMatrix(predict_rf_cv,
as.factor(teste[[response_var]]))
print(cm_rf_cv)
results$RF_CV <- list(Model = rf_cv, ConfusionMatrix = cm_rf_cv)

return(results)
}

```

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 a arquitetura de RNN vista no curso.

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

# Carga da base
cifar10 = tf.keras.datasets.cifar10
# Já está separado em dados de treino e teste
# Não precisa separar
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalização os dados
# Imagens em pixels de 0 - 255
```

```

# / 255.0 transforma em 0 - 1
x_train, x_test = x_train / 255.0, x_test / 255.0
# O dado y é a classe a qual faz parte
# O flatten torna os dados vetorizados
y_train, y_test = y_train.flatten(), y_test.flatten()
# Dimensão dos dados
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)

# A rede completa fica:
K = len(set(y_train))
# Aqui começa o Estágio 1
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
# Todas as imagens são do mesmo tamanho, não precisa de Global Pooling
x = Flatten()(x)
# Aqui começa o Estágio 2
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)
# Model ( lista entrada, lista saída)
model = Model(i, x)

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

# Compilar o modelo
model.compile(optimizer="adam",
loss="sparse_categorical_crossentropy", metrics=["accuracy"])
# Treinar o modelo
r = model.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=15)

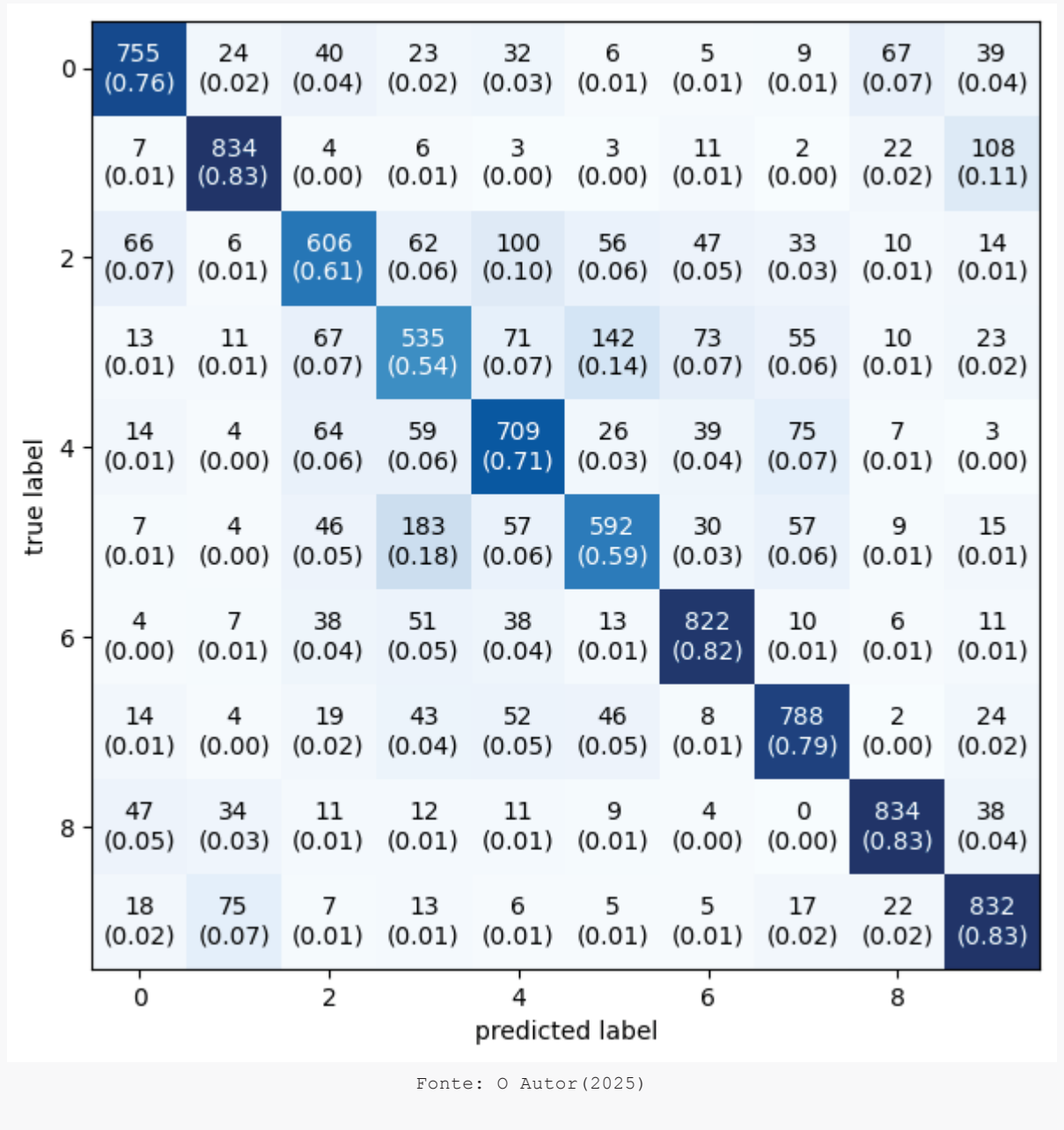
#Plotar a função de perda, treino e validação
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

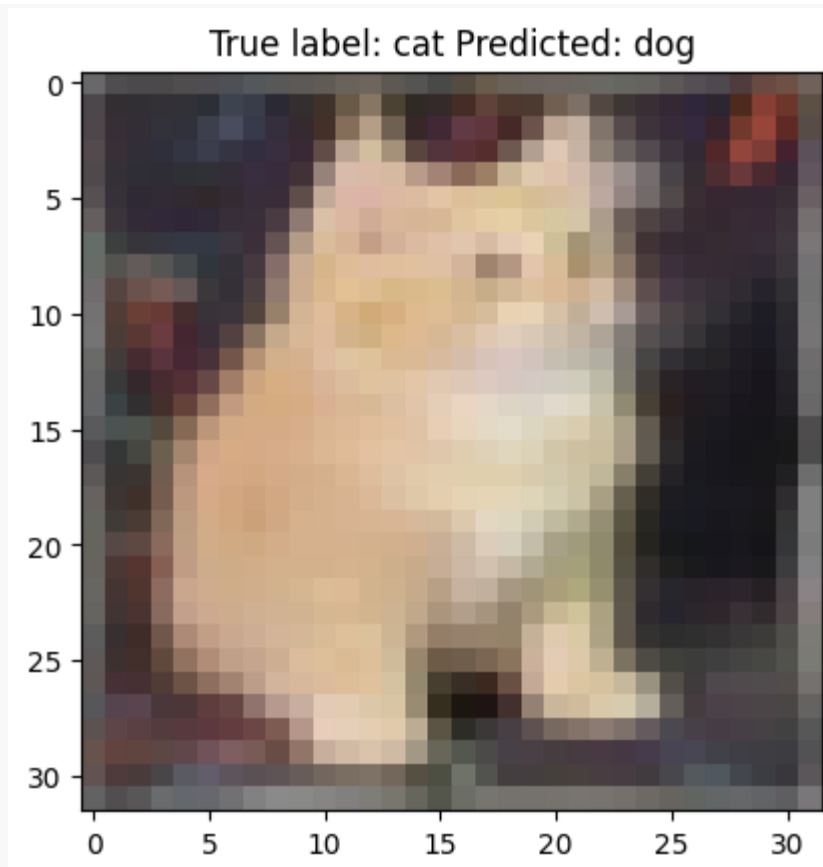
# Plotar acurácia, treino e validação
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
plt.show()

# Efetuar previsões na base de teste
# argmax é usado pois a função de ativação da saída é softmax

```

```
# argmax pega o neurônio que deu o maior resultado, isto é,  
# a maior probabilidade de saída  
y_pred = model.predict(x_test).argmax(axis=1)  
  
# Mostrar a matriz de confusão  
cm = confusion_matrix(y_test, y_pred)  
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),  
show_normed=True)  
  
# 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]]))
```





Fonte: O Autor (2025)

Exercício 2

```
# 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

# Separa a base em treino e teste
```

```

x_train, x_test, y_train, y_test = train_test_split(df["data"], y,
test_size=0.33)

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

# Acerta o tamanho das sequências (padding)
data_train = pad_sequences(sequences_train) # usa o tamanho da maior seq.
T = data_train.shape[1] # tamanho da sequência
data_test = pad_sequences(sequences_test, maxlen=T)
print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)

# 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) # Sigmoide pois só tem 2 valores
model = Model(i, x)

# Resumo do modelo
model.summary()

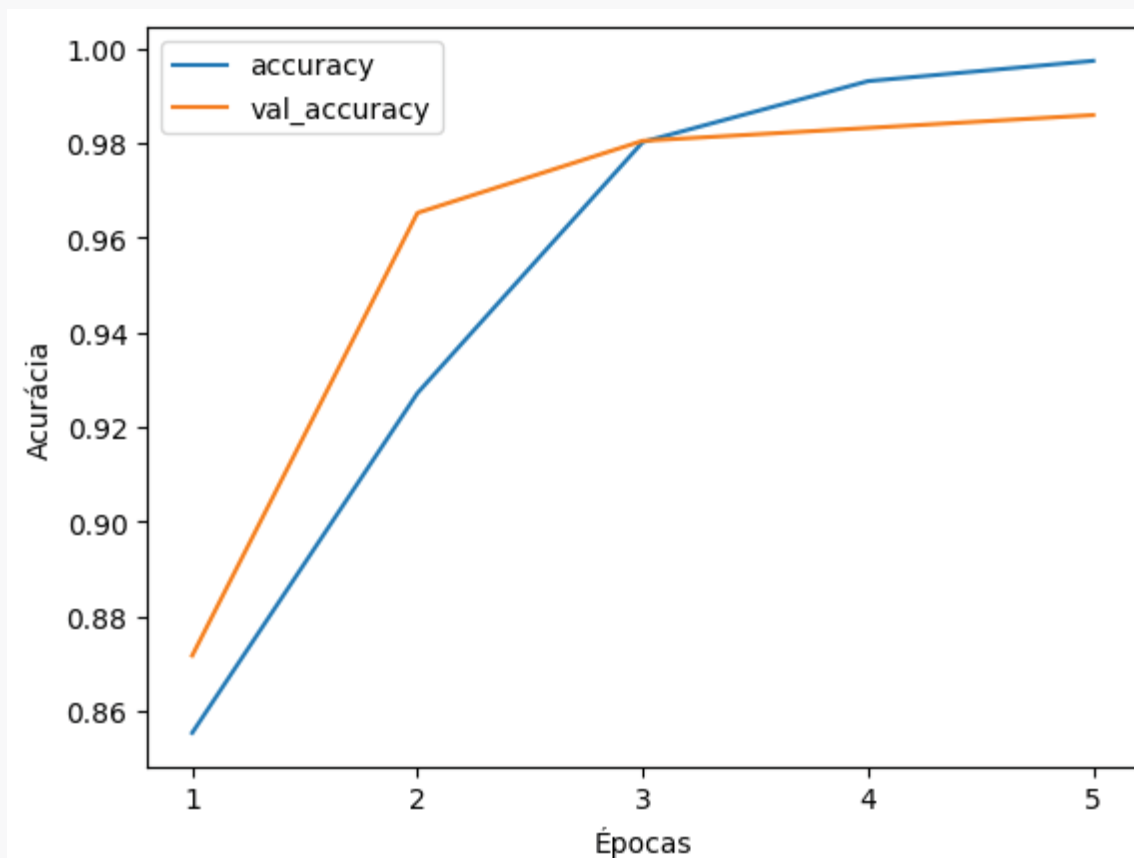
# Compila e treina o modelo
model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])
epochs = 5
r = model.fit(data_train, y_train, epochs=epochs, validation_data=(data_test,
y_test))

# 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()
```



Fonte: O Autor (2025)

```
# 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")
```

Exercício 3

```
# Importações
import tensorflow as tf
import glob
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
```

```

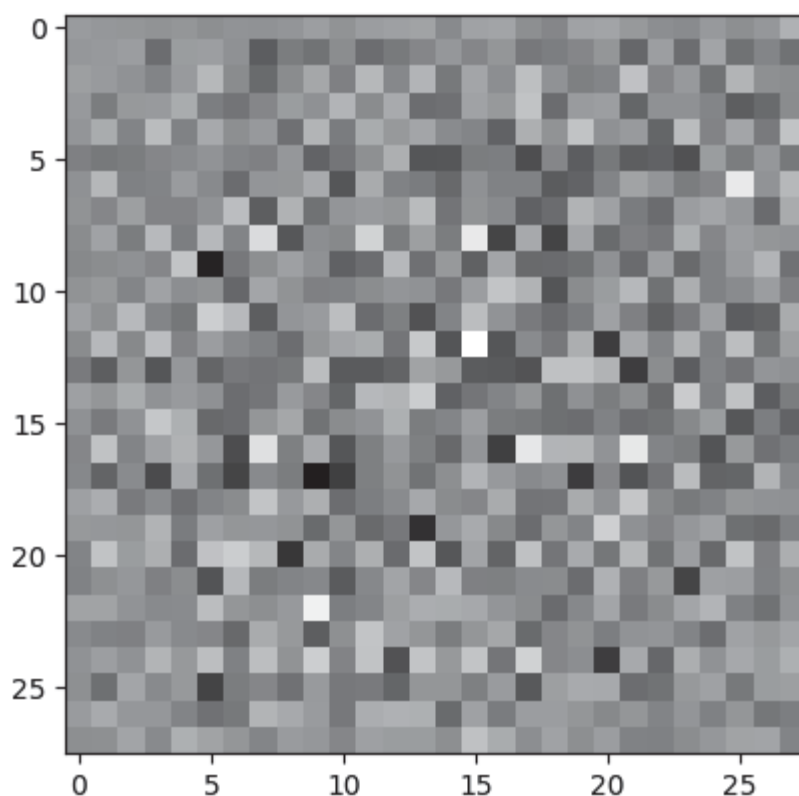
import PIL
from tensorflow.keras import layers
import time
from IPython import display

# Carregar a base de dados
(train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()
# Normalização
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5 # Normaliza entre [-1, 1]
# Gera o banco em partes e randomiza
BUFFER_SIZE = 60000
BATCH_SIZE = 256
# Cria o dataset (from_tensor_slices)
# Randomiza (shuffle)
# Combina elementos consecutivos em lotes (batch)
train_dataset =
tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).batch(B
ATCH_SIZE)

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

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

```



Fonte: O Autor (2025)

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

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

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

```

    total_loss = real_loss + fake_loss
    return total_loss
# Perda do GERADOR
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)

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

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

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

# Antes do passo de treinamento, vamos ver como são gerados os gradientes
# Seja a função  $3x^2 + 2xy$ 
# Para os valores  $x=5$  e  $y=3$  queremos as derivadas de  $f$  com respeito a  $x$  e  $y$ 
def f(x, y):
    return  $3x^2 + 2xy$ 
x, y = tf.Variable(5.), tf.Variable(3.)
with tf.GradientTape() as tape:
    z = f(x, y)
gradients = tape.gradient(z, [x, y])
print(gradients)

# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)
        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)
        gen_loss = generator_loss(fake_output)
        disc_loss = discriminator_loss(real_output, fake_output)
        gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
        gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)

```

```

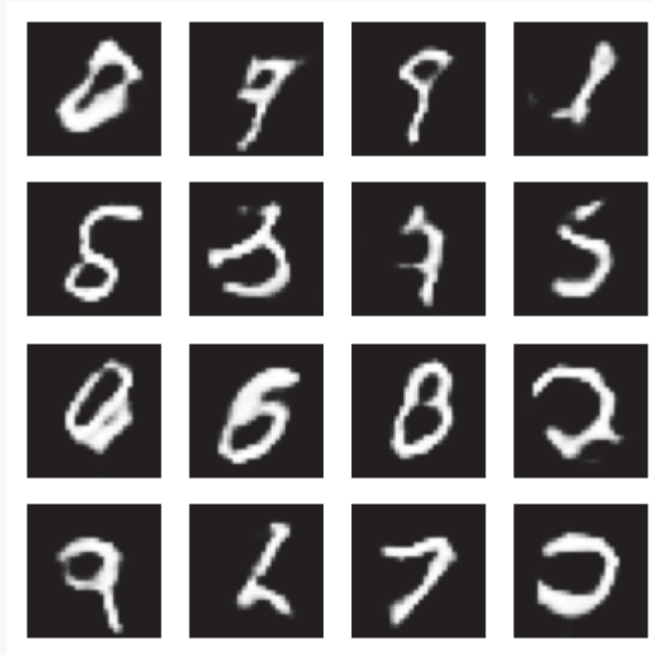
    generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
discriminator.trainable_variables))

# Treinamento completo/laço
def train(dataset, epochs):
    for epoch in range(epochs):
        start = time.time()
        for image_batch in dataset:
            train_step(image_batch)
        # Produce images for the GIF as you go
        display.clear_output(wait=True)
        generate_and_save_images(generator, epoch + 1, seed)
        # Save the model every 15 epochs
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)
        print ('Time for epoch {} is {} sec'.format(epoch + 1,
time.time()-start))
        # Generate after the final epoch
        display.clear_output(wait=True)
        generate_and_save_images(generator, epochs, seed)

# Gerar e salvar imagens
def generate_and_save_images(model, epoch, test_input):
    # Notice `training` is set to False.
    # This is so all layers run in inference mode (batchnorm).
    predictions = model(test_input, training=False)
    fig = plt.figure(figsize=(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))

```



Fonte: O Autor(2025)

Exercício 4

```
import collections
import logging
import os
import pathlib
import re
import string
import sys
import time
import numpy as np
import matplotlib.pyplot as plt
import tensorflow_datasets as tfds
import tensorflow_text as text
import tensorflow as tf
logging.getLogger('tensorflow').setLevel(logging.ERROR) # suppress warnings

# Carregar a base de dados
examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en',
                               with_info=True, as_supervised=True)
train_examples, val_examples = examples['train'], examples['validation']

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

# Tokenização e Destokenização do texto
model_name = "ted_hrlr_translate_pt_en_converter"
```

```

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

f"https://storage.googleapis.com/download.tensorflow.org/models/{model_name}.zip",

                        cache_dir='.', cache_subdir='', extract=True)
# Tem 2 tokenizers: um pt outro em en
# tokenizers.en tokeniza e detokeniza
tokenizers = tf.saved_model.load(model_name)

# PIPELINE DE ENTRADA
# Codificar/tokenizar lotes de texto puro
def tokenize_pairs(pt, en):
    pt = tokenizers.pt.tokenize(pt)
    # Converte ragged (irregular, tam variável) para dense
    # Faz padding com zeros.
    pt = pt.to_tensor()
    en = tokenizers.en.tokenize(en)
    # ragged -> dense
    en = en.to_tensor()
    return pt, en

# Pipeline simples: processa, embaralha, agrupa os dados, prefetch
# Datasets de entrada terminam com prefetch
BUFFER_SIZE = 20000
BATCH_SIZE = 64
def make_batches(ds):
    return (
        ds
        .cache()
        .shuffle(BUFFER_SIZE)
        .batch(BATCH_SIZE)
        .map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)
        .prefetch(tf.data.AUTOTUNE))

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

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

def positional_encoding(position, d_model):
    angle_rads = get_angles(np.arange(position)[:, np.newaxis],
                             np.arange(d_model)[np.newaxis, :], d_model)
    # sin em índices pares no array; 2i
    angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
    # cos em índices ímpares no array; 2i+1
    angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
    # newaxis, aumenta a dimensão [] -> [ [] ]
    pos_encoding = angle_rads[np.newaxis, ...]
    return tf.cast(pos_encoding, dtype=tf.float32)

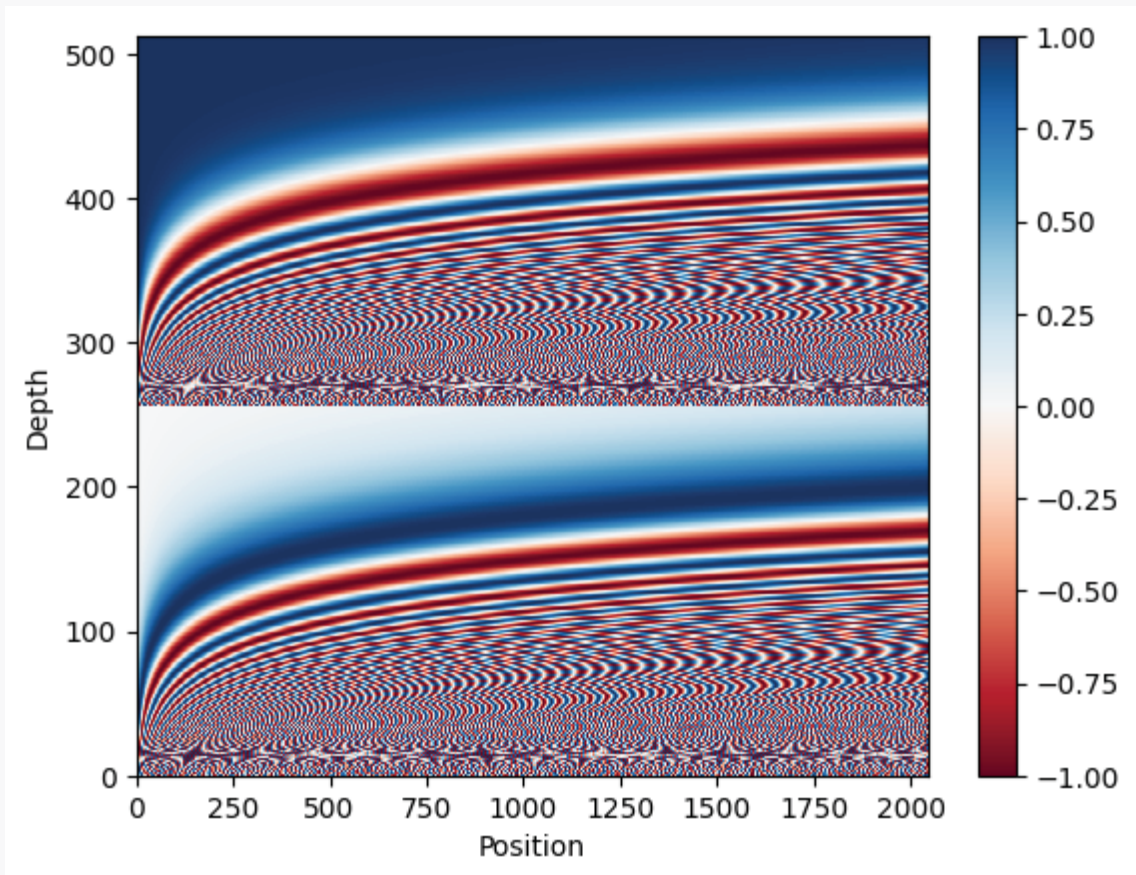
# CODIFICAÇÃO POSICIONAL

```

```

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

```



Fonte: O Autor(2025)

```

# Cria uma máscara de 0 e 1, 0 para quando há valor e 1 quando não há
def create_padding_mask(seq):
    seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
    # add extra dimensions to add the padding
    # to the attention logits.
    return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_len)

# Máscara futura, usada no decoder
def create_look_ahead_mask(size):
    # zera o triângulo inferior
    mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
    return mask # (seq_len, seq_len)

```

```

# Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):
    # Q K^T
    matmul_qk = tf.matmul(q, k, transpose_b=True) # (... , seq_len_q, seq_len_k)
    # converte matmul_qk para float32
    dk = tf.cast(tf.shape(k)[-1], tf.float32)
    # divide por sqrt(d_k)
    scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
    # Soma a máscara, e os valores faltantes serão um número próximo a -inf
    if mask is not None:
        scaled_attention_logits += (mask * -1e9)
    # softmax normaliza os dados, somam 1. // (... , seq_len_q, seq_len_k)
    attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
    output = tf.matmul(attention_weights, v) # (... , seq_len_q, depth_v)
    return output, attention_weights

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

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

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

        q = self.split_heads(q, batch_size) # (batch_size, num_heads, seq_len_q, depth)
        k = self.split_heads(k, batch_size) # (batch_size, num_heads, seq_len_k, depth)
        v = self.split_heads(v, batch_size) # (batch_size, num_heads, seq_len_v, depth)

        # Calcula a atenção para cada cabeça (de forma matricial)
        # scaled_attention.shape == (batch_size, num_heads, seq_len_q, depth)
        # attention_weights.shape == (batch_size, num_heads, seq_len_q, seq_len_k)

```

```

        scaled_attention, attention_weights = scaled_dot_product_attention(q, k,
v, mask)

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

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

class EncoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.1):
        super(EncoderLayer, self).__init__()
        self.mha = MultiHeadAttention(d_model, num_heads)
        self.ffn = point_wise_feed_forward_network(d_model, dff)
        self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = tf.keras.layers.Dropout(rate)
        self.dropout2 = tf.keras.layers.Dropout(rate)
    def call(self, x, training, mask):
        attn_output, _ = self.mha(x, x, x, mask) # (batch_size, input_seq_len,
d_model)
        attn_output = self.dropout1(attn_output, training=training)
        out1 = self.layernorm1(x + attn_output) # (batch_size, input_seq_len,
d_model)
        ffn_output = self.ffn(out1) # (batch_size, input_seq_len, d_model)
        ffn_output = self.dropout2(ffn_output, training=training)
        out2 = self.layernorm2(out1 + ffn_output) # (batch_size, input_seq_len,
d_model)
        return out2

class DecoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.1):
        super(DecoderLayer, self).__init__()
        self.mha1 = MultiHeadAttention(d_model, num_heads)
        self.mha2 = MultiHeadAttention(d_model, num_heads)
        self.ffn = point_wise_feed_forward_network(d_model, dff)
        self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
        self.dropout1 = tf.keras.layers.Dropout(rate)
        self.dropout2 = tf.keras.layers.Dropout(rate)
        self.dropout3 = tf.keras.layers.Dropout(rate)
    def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
        # enc_output.shape == (batch_size, input_seq_len, d_model)

```

```

    # (batch_size, target_seq_len, d_model)
    attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
    attn1 = self.dropout1(attn1, training=training)
    out1 = self.layer_norm1(attn1 + x)
    # (batch_size, target_seq_len, d_model)
    attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1,
padding_mask)
    attn2 = self.dropout2(attn2, training=training)
    out2 = self.layer_norm2(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.layer_norm3(ffn_output + out2) # (batch_size, target_seq_len,
d_model)
    return out3, attn_weights_block1, attn_weights_block2

class Encoder(tf.keras.layers.Layer):
    def __init__(self, num_layers, d_model, num_heads, dff,
input_vocab_size, maximum_position_encoding, rate=0.1):
        super(Encoder, self).__init__()
        self.d_model = d_model
        self.num_layers = num_layers
        self.embedding = tf.keras.layers.Embedding(input_vocab_size, d_model)
        self.pos_encoding = positional_encoding(maximum_position_encoding,
self.d_model)
        self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate) for _ in
range(num_layers)]
        self.dropout = tf.keras.layers.Dropout(rate)
    def call(self, x, training, mask):
        seq_len = tf.shape(x)[1]
        # adding embedding and position encoding.
        x = self.embedding(x) # (batch_size, input_seq_len, d_model)
        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
        x += self.pos_encoding[:, :seq_len, :]
        x = self.dropout(x, training=training)
        for i in range(self.num_layers):
            x = self.enc_layers[i](x, training, mask)
        return x # (batch_size, input_seq_len, d_model)

class Decoder(tf.keras.layers.Layer):
    def __init__(self, num_layers, d_model, num_heads, dff, target_vocab_size,
maximum_position_encoding, rate=0.1):
        super(Decoder, self).__init__()
        self.d_model = d_model
        self.num_layers = num_layers
        self.embedding = tf.keras.layers.Embedding(target_vocab_size, d_model)
        self.pos_encoding = positional_encoding(maximum_position_encoding,
d_model)
        self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate)
for _ in range(num_layers)]
        self.dropout = tf.keras.layers.Dropout(rate)
    def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
        seq_len = tf.shape(x)[1]

```

```

attention_weights = {}
x = self.embedding(x) # (batch_size, target_seq_len, d_model)
x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
x += self.pos_encoding[:, :seq_len, :]
x = self.dropout(x, training=training)
for i in range(self.num_layers):
    x, block1, block2 = self.dec_layers[i](x, enc_output, training,
                                           look_ahead_mask, padding_mask)

    attention_weights[f'decoder_layer{i+1}_block1'] = block1
    attention_weights[f'decoder_layer{i+1}_block2'] = block2
# x.shape == (batch_size, target_seq_len, d_model)
return x, attention_weights

class Transformer(tf.keras.Model):
    def __init__(self, num_layers, d_model, num_heads, dff, input_vocab_size,
                 target_vocab_size, pe_input, pe_target, rate=0.1):
        super().__init__()
        self.encoder = Encoder(num_layers, d_model, num_heads, dff,
                                input_vocab_size,
                                pe_input, rate)
        self.decoder = Decoder(num_layers, d_model, num_heads, dff,
                                target_vocab_size,
                                pe_target, rate)
        self.final_layer = tf.keras.layers.Dense(target_vocab_size)
    def call(self, inputs, training):
        # Keras models prefer if you pass all your inputs in the first argument
        inp, tar = inputs
        enc_padding_mask, look_ahead_mask, dec_padding_mask =
self.create_masks(inp, tar)
        # (batch_size, inp_seq_len, d_model)
        enc_output = self.encoder(inp, training, enc_padding_mask)
        # dec_output.shape == (batch_size, tar_seq_len, d_model)
        dec_output, attention_weights = self.decoder(tar, enc_output,
                                                    training, look_ahead_mask,
dec_padding_mask)
        # (batch_size, tar_seq_len, target_vocab_size)
        final_output = self.final_layer(dec_output)
        return final_output, attention_weights
    def create_masks(self, inp, tar):
        # Encoder padding mask
        enc_padding_mask = create_padding_mask(inp)
        # Used in the 2nd attention block in the decoder.
        # This padding mask is used to mask the encoder outputs.
        dec_padding_mask = create_padding_mask(inp)
        # Used in the 1st attention block in the decoder.
        # It is used to pad and mask future tokens in the input received by
        # the decoder.
        look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])
        dec_target_padding_mask = create_padding_mask(tar)
        look_ahead_mask = tf.maximum(dec_target_padding_mask, look_ahead_mask)
        return enc_padding_mask, look_ahead_mask, dec_padding_mask

# Hiperparâmetros

```

```

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

class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
    def __init__(self, d_model, warmup_steps=4000):
        super(CustomSchedule, self).__init__()
        self.d_model = d_model
        self.d_model = tf.cast(self.d_model, tf.float32)
        self.warmup_steps = warmup_steps
    def __call__(self, step):
        step = tf.cast(step, tf.float32) # Adicionado para evitar ERRO
        arg1 = tf.math.rsqrt(step)
        arg2 = step * (self.warmup_steps ** -1.5)
        return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

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

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask
    return tf.reduce_sum(loss_) / tf.reduce_sum(mask)
def accuracy_function(real, pred):
    accuracies = tf.equal(real, tf.argmax(pred, axis=2))
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    accuracies = tf.math.logical_and(mask, accuracies)
    accuracies = tf.cast(accuracies, dtype=tf.float32)
    mask = tf.cast(mask, dtype=tf.float32)
    return tf.reduce_sum(accuracies) / tf.reduce_sum(mask)

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

transformer = Transformer(num_layers=num_layers,
                           d_model=d_model,
                           num_heads=num_heads,
                           dff=dff,

input_vocab_size=tokenizers.pt.get_vocab_size().numpy(),

target_vocab_size=tokenizers.en.get_vocab_size().numpy(),
                           pe_input=1000,
                           pe_target=1000,
                           rate=dropout_rate)

```

```

checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(transformer=transformer,
                           optimizer=optimizer)
ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path,
max_to_keep=5)
# if a checkpoint exists, restore the latest checkpoint.
if ckpt_manager.latest_checkpoint:
    ckpt.restore(ckpt_manager.latest_checkpoint)
    print('Latest checkpoint restored!!')

EPOCHS = 20
train_step_signature = [
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]
@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
    tar_inp = tar[:, :-1]
    tar_real = tar[:, 1:]
    with tf.GradientTape() as tape:
        predictions, _ = transformer([inp, tar_inp],
                                     training = True)
        loss = loss_function(tar_real, predictions)
    gradients = tape.gradient(loss, transformer.trainable_variables)
    optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))
    train_loss(loss)
    train_accuracy(accuracy_function(tar_real, predictions))

for epoch in range(EPOCHS):
    start = time.time()
    train_loss.reset_state()
    train_accuracy.reset_state()
    # inp -> portuguese, tar -> english
    for (batch, (inp, tar)) in enumerate(train_batches):
        train_step(inp, tar)
        if batch % 50 == 0:
            print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f}
Accuracy {train_accuracy.result():.4f}')
        if (epoch + 1) % 5 == 0:
            ckpt_save_path = ckpt_manager.save()
            print(f'Saving checkpoint for epoch {epoch+1} at {ckpt_save_path}')
            print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}')
            print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')

class Translator(tf.Module):
    def __init__(self, tokenizers, transformer):
        self.tokenizers = tokenizers
        self.transformer = transformer
    def __call__(self, sentence, max_length=20):
        # input sentence is portuguese, hence adding the start and end token
        assert isinstance(sentence, tf.Tensor)
        if len(sentence.shape) == 0:

```

```

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

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

```

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

Projeto de sistema de Big Data para investimentos no mercado de ações brasileiro (B3) .

1. Introdução

O presente trabalho tem como objetivo apresentar um projeto para um sistema de Big Data para que contemple as etapas mínimas para coletar, transformar, estruturar, analisar, modelar e visualizar dados do mercado de ações nacional com o objetivo de construir um sistema de apoio à decisão para auxiliar na melhor alocação de recursos financeiros.

1. Fontes de Dados

Dados de Mercado:

- Preços em tempo real (tickers).
- Preços históricos (OHLCV).
- Book de ofertas.

Dados Financeiros:

- Balanços patrimoniais.
- Demonstrações de resultados.
- Indicadores fundamentalistas.
- Dados macroeconômicos: Taxa de juros, Inflação, PIB, etc.

Notícias e Sentimento de Mercado:

- Notícias financeiras.
 - Dados de mídias sociais.
 - Dados Macroeconômicos:
- #### 1. Arquitetura Geral do Sistema

A arquitetura geral do sistema é apresentada no diagrama abaixo. A seguir são apresentados os componentes principais da arquitetura geral do sistema proposto.

Camada de Ingestão de Dados:

Coleta de dados de múltiplas fontes em tempo real.

Ferramentas:

- Apache Kafka: Plataforma de streaming para ingestão de dados em tempo real.
- Apache Flume: Plataforma para coletar, agregar e mover grandes quantidades de dados.
- Apache NiFi: Automatização de fluxos de dados e ETL.
- APIs e Feeds de Dados: Integração com provedores como a B3, Bloomberg, Reuters.

Camada de Armazenamento:

Data Lake para armazenamento dos dados brutos.

Data Warehouse para dados estruturados e processados.

Ferramentas:

- Data Lake: Utilizando Hadoop HDFS ou Amazon S3 para armazenamento de dados brutos.
- Data Warehouse: PostgreSQL, Amazon Redshift, Google BigQuery ou Snowflake para dados estruturados.
- Bancos de Dados SQL: PostgreSQL.
- Bancos de Dados Temporais: InfluxDB, TimeScaleDB.
- Bancos de Dados NoSQL: Apache HBase, Cassandra, MongoDB.

Processamento:

Processamento em lote e em tempo real.

Limpeza, transformação e enriquecimento de dados.

Ferramentas:

- Apache Spark: Processamento distribuído para grandes volumes de dados.
- Apache Flink: Processamento em tempo real de fluxos de dados.
- Airflow: Orquestração de workflows e agendamento de tarefas.

Análise dados e Machine Learning:

Descoberta de padrões.

Treinamento e validação de modelos.

Ferramentas:

- Linguagens: Python, R.
- Bibliotecas: Pandas, NumPy, Scikit-learn, TensorFlow, Keras, PyTorch.
- Plataformas de ML: MLflow para gerenciamento de modelos.
- Notebooks: Jupyter Notebooks para análise de dados interativa.

Aplicação, GUI e Visualização:

Interface para usuários finais.

Dashboards, relatórios e APIs.

Ferramentas:

- Dashboards: Power BI, Tableau, Power BI, Grafana.
 - Aplicações Web: Django, Flask para desenvolvimento de GUI personalizadas.
1. Modelo de Dados
 1. Esquema de Dados Estruturados

Tabela de Preços Históricos:

Campos: Data/Hora, Ação, Preço de Abertura, Máximo, Mínimo, Preço de Fechamento, Volume.

Tabela de Dados Financeiros:

Campos: Ação, Período, Receita, Lucro Líquido, EBITDA, etc.

Tabela de Notícias e Sentimento:

Campos: Data/Hora, Fonte, Título, Corpo, Sentimento (positivo, negativo, neutro).

1. Metadados e Catálogo de Dados

Catálogo de Dados: Utilização de um catálogo para gestão e descoberta de dados.

Schemas e Validações: Definição de schemas para validação de dados durante a ingestão.

1. Dados não estruturados

Os textos de notícias, posts de redes sociais, entre outros dados não estruturados, serão armazenados em sistema de arquivos distribuído Hadoop HDFS ou Amazon S3.

1. Pipeline de Dados

Ingestão:

- Coleta de dados em tempo real via Kafka.
- Ingestão de dados históricos em lote.

Armazenamento Inicial:

- Dados brutos armazenados no Data Lake.

Processamento Primário:

- Limpeza e transformação usando Spark.
- Cálculo de indicadores técnicos e fundamentalistas.

Armazenamento Processado:

- Dados transformados armazenados no Data Warehouse.

Análise e Modelagem:

- Análise exploratória de dados.
- Treinamento de modelos de machine learning.

Distribuição e Visualização:

- Dados disponibilizados para aplicações e dashboards.
- APIs para acesso a modelos preditivos.

6. Desenvolvimento de Estratégias e Portfólios

Análise Técnica:

- Identificação de padrões gráficos.
- Utilização de indicadores como RSI, MACD.

Análise Fundamentalista:

- Avaliação de empresas com base em indicadores financeiros.

Machine Learning e IA:

- Modelos de previsão de preços.
- Clusterização de ativos.
- Análise de sentimento com NLP.

Backtesting:

- Teste de estratégias em dados históricos.
- Avaliação de desempenho e risco.

Conclusão

Foram apresentados a concepção e projeto geral para um sistema de apoio à decisão de investimentos no mercado financeiro, especificamente do mercado de ações brasileiro (B3). O projeto contempla uma arquitetura básica, ferramentas a serem utilizadas, um pipeline de dados e produtos e serviços finais.

Os dados envolvidos incluem dados financeiros de alta frequência (ticks), books de ofertas, notícias financeiras, redes sociais, relatórios trimestrais, dados relacionados ao cenário do mercado financeiro local e global, entre outros. Novas fontes de dados podem ser adicionadas sempre que houver necessidade. Como trata-se de uma proposta ampla, estão envolvidos todos os V's do Big Data.

A seguir, são feitos breves comentários sobre cada um dos V's e seu impacto no projeto desse trabalho.

- **Volume:** O histórico dos preços, volumes e books de ofertas embora sejam relativamente compactos são de uma grande variedade de ativos e são coletados e armazenados de forma cumulativa, o que leva a um volume significativo e com tendência a aumentar ao longo do tempo. As informações relacionadas às menções aos ativos e cenário financeiro coletadas nas redes sociais tendem a gerar um volume mais significativo, pois são dados não estruturados e que também serão coletados de forma cumulativa ao longo do tempo. Além disso, o fato de o sistema estar aberto a novas fontes de dados, faz com que o volume seja um ponto muito importante no projeto.
- **Velocidade:** Os preços e livros de ofertas são atualizados praticamente em tempo real e o tempo é importante para algumas estratégias que exploram oportunidades que oferecem uma janela de tempo muito estreita. Além dos dados financeiros propriamente ditos, as fontes auxiliares de dados, como as menções nas redes sociais também são produzidas em uma velocidade alta. Assim, há necessidade de ingestão e processamento em tempo praticamente real para identificar oportunidades de investimento.
- **Variedade:** Como o sistema pretende ser um sistema completo que agregue qualquer tipo de informação relevante para apoiar a tomada de decisão, vários tipos de dados serão utilizados, como os dados estruturados de

preços, indicadores financeiros, assim como dados não estruturados vindos de notícias, posts em redes sociais, entre outros.

- **Veracidade:** É importante garantir a confiabilidade dos dados para minimizar o risco de decisões erradas.
- **Valor:** O mais importante dos V's, pois sem ele nenhum dos outros V's teria sentido. Aqui o valor é oferecer aos investidores informações que subsidiem a tomada de decisões de investimentos que levem ao atingimento dos seus objetivos de forma racional e objetiva.

APÊNDICE 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*
- Aplice 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

```
"""
* Ajustar dataset validação para ter 1 exemplo de cada classe
"""

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

# Baixando e descompactando datasets

!wget
https://github.com/valeriow/IAA-PUB/raw/refs/heads/main/Test_Warwick.zip

!wget
https://github.com/valeriow/IAA-PUB/raw/refs/heads/main/Train_Warwick.zip

!unzip -o Test_Warwick.zip

!unzip -o Train_Warwick.zip

import random

import glob

import os

from collections import defaultdict

import shutil
```

```
import math

import cv2

import numpy as np

from skimage.feature import local_binary_pattern

import matplotlib.pyplot as plt


import pandas as pd


from tensorflow.keras.models import Model


#from tensorflow.keras.applications.resnet50 import preprocess_input
#from tensorflow.keras.applications.resnet50 import ResNet50


from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Dense, Flatten


from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.applications import VGG16


from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.neural_network import MLPClassifier


from sklearn.metrics import classification_report, accuracy_score


# Obtendo a lista de pacientes para fazer a separação da base de treinamento
em treino e validação
```

```

# Obtém a lista de arquivos
all_train_images = glob.glob("/content/Train_4cls_amostra/**/*.png")

# Itera sobre a lista de arquivos obtendo o id do paciente
patients_images = defaultdict(list)

for img_path in all_train_images:
    filename = os.path.basename(img_path)
    patient_id = filename.split('_')[0] # Pega o XX (número do paciente)
    patients_images[patient_id].append(img_path)

# Obter a lista de pacientes (as chaves do defaultdict)
patients = list(patients_images.keys())

random.seed(42)
random.shuffle(patients)

# Dividir 80% para treino e 20% para validação
num_patients = len(patients)
num_train = math.ceil(0.8 * num_patients)

train_patients = patients[:num_train] # Primeiro 80% dos pacientes
val_patients = patients[num_train:] # Último 20% dos pacientes

# Função para mover os arquivos para as respectivas pastas
def move_images(patients_list, destination_folder):
    if not os.path.exists(destination_folder):
        os.makedirs(destination_folder)

```

```

for patient_id in patients_list:

    for img_path in patients_images[patient_id]:

        if img_path: # Verifica se há um caminho de imagem

            subdir = img_path.split('/')[-2]

            # Define o caminho de destino

            destination_path = os.path.join(destination_folder, subdir)

            # Criar o diretório de destino se não existir

            if not os.path.exists(destination_path):

                os.makedirs(destination_path)

            # Move o arquivo para o diretório de destino

            #print(img_path,destination_path)

            shutil.copy(img_path, destination_path)


# Diretórios para armazenar treino e validação
train_dir = '/content/train/'
val_dir = '/content/valid/'
test_dir = '/content/test/'


# Exclui diretórios antes da cópia se existirem
for dir in [train_dir, val_dir, test_dir]:

    if os.path.exists(dir):

        shutil.rmtree(dir)


# Copiar 80% dos pacientes para a pasta train
move_images(train_patients, train_dir)


# Copiar 20% dos pacientes para a pasta valid

```

```

move_images(val_patients, val_dir)

# Copiar os pacientes para a pasta test
shutil.copytree("/content/Test_4cl_amostra", f"{test_dir}")

# Função para calcular LBP de uma imagem
def calculate_lbp(image_path, radius=1, n_points=8):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    lbp = local_binary_pattern(img, n_points, radius, method='uniform')
    return lbp

# Função que calcula o histograma lbp de uma imagem.
def calculate_lbp_hist(image_path, radius=1, n_points=8):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    height, width = img.shape
    lbp = local_binary_pattern(img, n_points, radius, method='uniform')
    # (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3),
    range=(0, n_points + 2))
    (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, height+1),
    range=(0, width + 1))
    hist = hist.astype("float")
    hist /= hist.sum() # Normalizar o histograma
    return hist

# Função auxiliar que separa caminho, nome e extensão de um caminho completo
def separa_caminho(img_path):
    len_img_path = len(img_path)
    img_path_split = img_path.split('/')
    nome_arquivo = img_path_split[-1].split('.')[0]

```

```

extensao_arquivo = img_path_split[-1].split('.')[1]

caminho_arquivo = "/".join(img_path_split[:len(img_path_split)-1])+"/"

return (nome_arquivo, extensao_arquivo, caminho_arquivo)

# Percorre todas as imagens de treinamento e extrai LPB de cada salva num
arquivo com o mesmo nome da imagem + _lbp.csv

def percorre_arquivos_extraindo_caracteristicas_lbp(base_dir="train",
radius=1, n_points=8):

    #base_dir = train_dir

    files = glob.glob(f"{base_dir}/**/*.png")

    for img_path in files:

        lbp_hist = calculate_lbp_hist(img_path, radius=1, n_points=8)

        lbp_hist = lbp_hist.reshape(1, -1)

        nome_arquivo, extensao_arquivo, caminho_arquivo =
separa_caminho(img_path)

        #novo_arquivo = f"{caminho_arquivo}{nome_arquivo}_lbp.npy"

        #np.save(novo_arquivo, lbp_hist)

        novo_arquivo = f"{caminho_arquivo}{nome_arquivo}_lbp.csv"

        np.savetxt(novo_arquivo, lbp_hist, delimiter=',', fmt='%f')

# Percorre todas as imagens de treinamento e extrai LPB de cada salva num
arquivo com o mesmo nome da imagem + _lbp.npy

percorre_arquivos_extraindo_caracteristicas_lbp("train", radius=1,
n_points=8)

percorre_arquivos_extraindo_caracteristicas_lbp("valid", radius=1,
n_points=8)

percorre_arquivos_extraindo_caracteristicas_lbp("test", radius=1, n_points=8)

```

```

classes = [classe.split("/")[-1] for classe in
glob.glob("/content/Train_4cls_amostra/*")]

classes

#Modelo que será utilizado, sem a camada top (adicionada manualmente depois)

vgg16_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))

vgg16_model.trainable = False

vgg16_model.summary()

def extrai_caracteristicas_vgg16(img_path, model):

    img = cv2.imread(img_path)

    img = cv2.resize(img, (224, 224))

    img = np.expand_dims(img, axis=0)

    return model.predict(img)

def percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir, model):

    files = glob.glob(f"{base_dir}/*/*.png")

    for img_path in files:

        nome_arquivo, extensao_arquivo, caminho_arquivo =
separa_caminho(img_path)

        novo_arquivo = f"{caminho_arquivo}{nome_arquivo}_vgg16.csv"

        data = extrai_caracteristicas_vgg16(img_path, model)

        data = data.reshape(1, -1)

        np.savetxt(novo_arquivo, data, delimiter=',', fmt='%f')

        #np.save(novo_arquivo, extrai_caracteristicas_vgg16(img_path))

```

```

"""base_dir = 'train'

train_files = glob.glob(f"{base_dir}/**/*.png")

base_dir = 'valid'

val_files = glob.glob(f"{base_dir}/**/*.png") """

percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir= "train", model =
vgg16_model)

percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir= "valid", model =
vgg16_model)

percorre_arquivos_extraindo_caracteristicas_vgg16(base_dir= "test", model =
vgg16_model)


base_dir = 'train'

paths_lbp_features_train = glob.glob(f"{base_dir}/**/*.lbp.csv")

paths_vgg16_features_train = glob.glob(f"{base_dir}/**/*.vgg16.csv")


base_dir = 'valid'

paths_lbp_features_valid = glob.glob(f"{base_dir}/**/*.lbp.csv")

paths_vgg16_features_valid = glob.glob(f"{base_dir}/**/*.vgg16.csv")


base_dir = 'test'

paths_lbp_features_test = glob.glob(f"{base_dir}/**/*.lbp.csv")

paths_vgg16_features_test = glob.glob(f"{base_dir}/**/*.vgg16.csv")


# Percorre os arquivos csv com as características LBP e VGG16 extraídas antes
para cada dataset (train, valid, test)


#LBP

```

```

lbp_features_train = []
lbp_labels_train = []
for file_path in paths_lbp_features_train:
    data = np.loadtxt(file_path , delimiter=',')
    lbp_features_train.append(data)
    classe = separa_caminho(file_path)[2].split("/") [1]
    lbp_labels_train.append(classe)

lbp_features_valid = []
lbp_labels_valid = []
for file_path in paths_lbp_features_valid:
    data = np.loadtxt(file_path , delimiter=',')
    lbp_features_valid.append(data)
    classe = separa_caminho(file_path)[2].split("/") [1]
    lbp_labels_valid.append(classe)

lbp_features_test = []
lbp_labels_test = []
for file_path in paths_lbp_features_test:
    data = np.loadtxt(file_path , delimiter=',')
    lbp_features_test.append(data)
    classe = separa_caminho(file_path)[2].split("/") [1]
    lbp_labels_test.append(classe)

#VGG16
vgg16_features_train = []
vgg16_labels_train = []

```

```

for file_path in paths_vgg16_features_train:

    data = np.loadtxt(file_path , delimiter=',')

    vgg16_features_train.append(data)

    classe = separa_caminho(file_path)[2].split("/") [1]

    vgg16_labels_train.append(classe)


vgg16_features_valid = []
vgg16_labels_valid = []
for file_path in paths_vgg16_features_valid:

    data = np.loadtxt(file_path , delimiter=',')

    vgg16_features_valid.append(data)

    classe = separa_caminho(file_path)[2].split("/") [1]

    vgg16_labels_valid.append(classe)


vgg16_features_test = []
vgg16_labels_test = []
for file_path in paths_vgg16_features_test:

    data = np.loadtxt(file_path , delimiter=',')

    vgg16_features_test.append(data)

    classe = separa_caminho(file_path)[2].split("/") [1]

    vgg16_labels_test.append(classe)


print(len(lbp_features_valid), len(lbp_labels_valid),len(lbp_labels_test))


from sklearn.metrics import confusion_matrix


def calculate_metrics(cm):

```

```

# Calcular a matriz de confusão

#cm = confusion_matrix(y_val, val_predictions)

# Inicializar contadores para micro F1-Score

TP_total = 0

FP_total = 0

FN_total = 0

TN_total = 0

f1_scores = []

sensibilities = []

specificities = []

# Número de classes

num_classes = cm.shape[0]

# Calcular precisão, recall e F1-Score para cada classe

for i in range(num_classes):

    TP = cm[i, i]

    FP = cm[:, i].sum() - TP # Falsos positivos

    FN = cm[i, :].sum() - TP # Falsos negativos

    TN = cm.sum() - (TP + FP + FN) # Verdadeiros negativos

# Calcular Precisão e Recall

precision = TP / (TP + FP) if (TP + FP) > 0 else 0

recall = TP / (TP + FN) if (TP + FN) > 0 else 0

sensibilidade = TP / (TP + FN) if (TP + FN) > 0 else 0

especificidade = TN / (TN + FP) if (TN + FP) > 0 else 0

```

```

    f1 = 2 * (precision * recall) / (precision + recall) if (precision +
recall) > 0 else 0

    # Acumular valores

    TP_total += TP

    FP_total += FP

    FN_total += FN

    TN_total += TN

    f1_scores.append(f1)

    sensibilities.append(sensibilidade)

    specificities.append(especificidade)

    accuracy = (TP_total + TN_total) / (TP_total + TN_total + FP_total +
FN_total) if (TP_total + TN_total + FP_total + FN_total) > 0 else 0

    n_classes = [cm[i, :].sum() for i in range(num_classes)] # Número de
instâncias por classe

    # Cálculo do F1-Score Micro

    #precision_micro = TP_total / (TP_total + FP_total) if (TP_total +
FP_total) > 0 else 0

    #recall_micro = TP_total / (TP_total + FN_total) if (TP_total + FN_total) >
0 else 0

    #f1_micro = 2 * (precision_micro * recall_micro) / (precision_micro +
recall_micro) if (precision_micro + recall_micro) > 0 else 0

    # Cálculo do F1-Score Weighted

    f1_weighted = sum(f1 * n for f1, n in zip(f1_scores, n_classes)) /
sum(n_classes) if sum(n_classes) > 0 else 0

```

```

# Sensibilidad ponderada

sensitivity_weighted = sum(sens * n for sens, n in zip(sensibilities,
n_classes)) / sum(n_classes) if sum(n_classes) > 0 else 0

# Especificidad ponderada

specificity_weighted = sum(spec * n for spec, n in zip(specificities,
n_classes)) / sum(n_classes) if sum(n_classes) > 0 else 0

return accuracy, sensitivity_weighted, specificity_weighted, f1_weighted

def train_evaluate(model, X_train, y_train, X_val, y_val):

    model.fit(X_train, y_train)

    train_predictions = model.predict(X_train)

    score_train = accuracy_score(y_train, train_predictions)

    val_predictions = model.predict(X_val)

    score_val = accuracy_score(y_val, val_predictions)

    return model, train_predictions, val_predictions, score_train, score_val

#Modelos para características LBP

rf_lbp = RandomForestClassifier(n_estimators=100)

svm_lbp = SVC(kernel='linear')

mlp_lbp = MLPClassifier(hidden_layer_sizes=(40, 40, 40, 4), max_iter=10000,
random_state=42)

#Modelos para características VGG16

rf_vgg16 = RandomForestClassifier(n_estimators=100)

```

```

svm_vgg16 = SVC(kernel='linear')

mlp_vgg16 = MLPClassifier(hidden_layer_sizes=(40, 40, 40, 4), max_iter=10000,
random_state=42)

# Vamos usar inicialmente as características lbp

X_train = lbp_features_train
y_train = lbp_labels_train

X_val = lbp_features_valid
y_val = lbp_labels_valid

X_test = lbp_features_test
y_test = lbp_labels_test

def metricas(tipo, df_resultados, y_val, val_predictions):

    cm = confusion_matrix(y_val, val_predictions)

    accuracy, sensibility_weighted, specificity_weighted, f1_weighted =
calculate_metrics(cm)

    data = { 'Tipo' : tipo,

            'Acurária': [accuracy],

            'Sensibilidade Ponderada': [sensibility_weighted],

            'Especificidade Ponderada': [specificity_weighted],

            'F1 Ponderada': [f1_weighted]}

    df_resultados = pd.concat([df_resultados, pd.DataFrame(data)],
ignore_index=True)

    return df_resultados

```

```

df_resultados_valid = pd.DataFrame()

df_resultados_test = pd.DataFrame()


#Random Forest

model, train_predictions, val_predictions,score_train, score_val =
train_evaluate(model=rf_lbp, X_train=X_train, y_train=y_train, X_val=X_val,
y_val=y_val)

tipo = 'rf_lbp'

#Resultados Validação

df_resultados_valid = metricas(tipo, df_resultados_valid, y_val,
val_predictions)

df_resultados_valid

#Resultados Test

test_predictions = model.predict(X_test)

df_resultados_test = metricas(tipo, df_resultados_test, y_test,
test_predictions)


#SVM

model, train_predictions, val_predictions,score_train, score_val =
train_evaluate(model=svm_lbp, X_train=X_train, y_train=y_train, X_val=X_val,
y_val=y_val)

tipo = 'svm_lbp'

#Resultados Validação

df_resultados_valid = metricas(tipo, df_resultados_valid, y_val,
val_predictions)

df_resultados_valid

#Resultados Test

test_predictions = model.predict(X_test)

```

```

df_resultados_test = metricas(tipo, df_resultados_test, y_test,
test_predictions)

#RNA

model, train_predictions, val_predictions, score_train, score_val =
train_evaluate(model=mlp_lbp, X_train=X_train, y_train=y_train, X_val=X_val,
y_val=y_val)

tipo = 'rna_lbp'

#Resultados Validação

df_resultados_valid = metricas(tipo, df_resultados_valid, y_val,
val_predictions)

df_resultados_valid

#Resultados Test

test_predictions = model.predict(X_test)

df_resultados_test = metricas(tipo, df_resultados_test, y_test,
test_predictions)

# Vamos usar agora as características extraídas da última camada de
convolução da Vgg16

X_train = vgg16_features_train
y_train = vgg16_labels_train

X_val = vgg16_features_valid
y_val = vgg16_labels_valid

X_test = vgg16_features_test
y_test = vgg16_labels_test

#Random Forest

```

```

model, train_predictions, val_predictions, score_train, score_val =
train_evaluate(model=rf_vgg16, X_train=X_train, y_train=y_train, X_val=X_val,
y_val=y_val)

tipo = 'rf_vgg16'

#Resultados Validação

df_resultados_valid = metricas(tipo, df_resultados_valid, y_val,
val_predictions)

df_resultados_valid

#Resultados Test

test_predictions = model.predict(X_test)

df_resultados_test = metricas(tipo, df_resultados_test, y_test,
test_predictions)


#SVM

model, train_predictions, val_predictions, score_train, score_val =
train_evaluate(model=svm_vgg16, X_train=X_train, y_train=y_train,
X_val=X_val, y_val=y_val)

tipo = 'svm_vgg16'

#Resultados Validação

df_resultados_valid = metricas(tipo, df_resultados_valid, y_val,
val_predictions)

df_resultados_valid

#Resultados Test

test_predictions = model.predict(X_test)

df_resultados_test = metricas(tipo, df_resultados_test, y_test,
test_predictions)


#RNA

model, train_predictions, val_predictions, score_train, score_val =
train_evaluate(model=mlp_vgg16, X_train=X_train, y_train=y_train,
X_val=X_val, y_val=y_val)

tipo = 'rna_vgg16'

#Resultados Validação

```

```

df_resultados_valid = metricas(tipo, df_resultados_valid, y_val,
val_predictions)

df_resultados_valid

#Resultados Test

test_predictions = model.predict(X_test)

df_resultados_test = metricas(tipo, df_resultados_test, y_test,
test_predictions)


#Ordenando os resultados de validação por 'F1 Ponderada'

df_resultados_valid = df_resultados_valid.sort_values('F1 Ponderada',
ascending=False)

df_resultados_valid


#Ordenando os resultados de teste por 'F1 Ponderada'

df_resultados_test = df_resultados_test.sort_values('F1 Ponderada',
ascending=False)

df_resultados_test


TipoAcuráriaSensibilidade PonderadaEspecificidade PonderadaF1
Ponderada4svm_vgg160.9285710.8571430.9514900.8495913rf_vgg160.8611860.7223720
.9051360.6416262rna_lbp0.7843670.5687330.8523030.5785300rf_lbp0.7803230.56064
70.8534830.5662961svm_lbp0.6212940.2425880.7574120.0947205rna_vgg160.6212940.
2425880.7574120.094720

```

****Conclusão:****

Verificando as métricas obtidas tanto no conjunto de validação quanto no conjunto de testes percebe-se a melhor F1 Ponderada foi do Modelo SVM utilizando características extraídas da rede VGG16.

A F1 Ponderada é uma métrica importante para problemas de classificação com múltiplas classes, especialmente quando há um desbalanceamento entre as classes. Ela considera tanto a precisão quanto a sensibilidade (recall) e

fornece uma medida balanceada do desempenho do modelo em todas as classes. Um valor de F1 ponderada próximo de 1 indica um bom equilíbrio entre precisão e recall.

Além disso, percebe-se que as demais métricas também seguem o mesmo padrão. A alta acurácia e a especificidade ponderada indicam que o modelo SVM/VGG16 possui uma boa capacidade de identificar corretamente as amostras negativas, sem perder muitas positivas, o que é crucial em contextos médicos para evitar falsos negativos.

```
# Gerando os arquivos zip com os csvs vgg16 e lbp

import zipfile

def zipa(tipo):

    arquivos = glob.glob(f'{train_dir}/*/*_{tipo}.csv') # +
    glob.glob(f'{val_dir}/*/*_vgg16.csv') +
    glob.glob(f'{test_dir}/*/*_vgg16.csv')

    nome_arquivo_zip = f'csv_{tipo}.zip'

    with zipfile.ZipFile(nome_arquivo_zip, 'w') as zipf:

        for arquivo in arquivos:

            zipf.write(arquivo, arcname=arquivo.split('/')[-1]) # arcname
mantém apenas o nome do arquivo dentro do zip

#lbp

zipa('lbp')

#vgg16

zipa('vgg16')
```

```

[4]: import random

import glob

import zipfile

23

import os

from collections import defaultdict

import shutil

import math

import cv2

import numpy as np

from skimage.feature import local_binary_pattern

import matplotlib.pyplot as plt

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.applications.vgg16 import preprocess_input

from tensorflow.keras.applications.resnet50 import preprocess_input as _preprocess_input_resnet

from tensorflow.keras.applications.resnet50 import ResNet50

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.applications.vgg16 import preprocess_input as _preprocess_input_vgg16

from tensorflow.keras.applications import VGG16

[5]: # Aqui utilizamos o mesmo código do exercício anterior para separar a
base de treino apropriadamente

# Função para mover os arquivos para as respectivas pastas

```

```

def move_images(patients_list, destination_folder):
    if not os.path.exists(destination_folder):
        os.makedirs(destination_folder)
    for patient_id in patients_list:
        for img_path in patients_images[patient_id]:
            if img_path: # Verifica se há um caminho de imagem
                subdir = img_path.split('/')[2]
                # Define o caminho de destino
                destination_path = os.path.join(destination_folder, subdir)
                # Criar o diretório de destino se não existir
                if not os.path.exists(destination_path):
                    os.makedirs(destination_path)
                # Move o arquivo para o diretório de destino
                #print(img_path, destination_path)
                shutil.copy(img_path, destination_path)

24

# Obtendo a lista de pacientes para fazer a separação da base de treinamento
em treino e validação

# Obtém a lista de arquivos
all_train_images = glob.glob("/content/Train_4cls_amostra/*/*.png")
# Itera sobre a lista de arquivos obtendo o id do paciente
patients_images = defaultdict(list)
for img_path in all_train_images:
    filename = os.path.basename(img_path)
    patient_id = filename.split('_')[0] # Pega o XX (número do paciente)
    patients_images[patient_id].append(img_path)
# Obter a lista de pacientes (as chaves do defaultdict)
patients = list(patients_images.keys())
random.seed(42)
random.shuffle(patients)
# Dividir 80% para treino e 20% para validação
num_patients = len(patients)
num_train = math.ceil(0.8 * num_patients)
train_patients = patients[:num_train] # Primeiro 80% dos pacientes

```

```

val_patients = patients[num_train:] # Último 20% dos pacientes
# Diretórios para armazenar treino e validação
train_dir = '/content/train/'
val_dir = '/content/valid/'
test_dir = '/content/test/'
# Exclui diretórios antes da cópia se existirem
for dir in [train_dir, val_dir, test_dir]:
    if os.path.exists(dir):
        shutil.rmtree(dir)
# Copiar 80% dos pacientes para a pasta train
move_images(train_patients, train_dir)
# Copiar 20% dos pacientes para a pasta valid
move_images(val_patients, val_dir)
# Copiar os pacientes para a pasta test
shutil.copytree("/content/Test_4cl_amostra", f"{test_dir}")
classes = [classe.split("/")[-1] for classe in
glob.glob("/content/Train_4cls_amostra/*")]
classes

[6]: ['1', '0', '3', '2']

[7]: # Com data augmentation

BATCH_SIZE = 32 # quantidade de imagens criadas em cada ciclo
#Vgg16
train_generator_dataaug_vgg16 = ImageDataGenerator(
    rescale=1 / 255.,
    rotation_range=90,
    brightness_range=[0.1, 0.7],
    width_shift_range=0.5,
    height_shift_range=0.5,
    horizontal_flip=True,
    vertical_flip=True,

preprocessing_function=preprocess_input_vgg16)
train_generator_vgg16 = ImageDataGenerator(rescale=1 / 255.,

preprocessing_function=preprocess_input_vgg16)
valid_generator_vgg16 = ImageDataGenerator(rescale=1 / 255.,

preprocessing_function=preprocess_input_vgg16)
test_generator_vgg16 = ImageDataGenerator(rescale=1 / 255.,

preprocessing_function=preprocess_input_vgg16)
#REsnet
train_generator_dataaug_resnet = ImageDataGenerator(
    #rescale=1 / 255.,
    rotation_range=90,
    brightness_range=[0.1, 0.7],
    width_shift_range=0.5,
    height_shift_range=0.5,
    horizontal_flip=True,
    vertical_flip=True,

```

```

preprocessing_function=preprocess_input_resnet)
train_generator_resnet = ImageDataGenerator(rescale=1 / 255.,

preprocessing_function=preprocess_input_resnet)
valid_generator_resnet = ImageDataGenerator(rescale=1 / 255.,

preprocessing_function=preprocess_input_resnet)
test_generator_resnet = ImageDataGenerator(rescale=1 / 255.,

preprocessing_function=preprocess_input_resnet)

[8]: #VGG16

#VGG16
print("VGG16")
print("DataAug")
traingen_dataaug_vgg16 =
train_generator_dataaug_vgg16.flow_from_directory("/content/train/",
                                                    target_size=(224, 224),
                                                    batch_size=BATCH_SIZE,
                                                    class_mode='categorical',
                                                    classes=classes,
                                                    shuffle=True,
                                                    seed=42)

print("No DataAug")
traingen_vgg16 = train_generator_vgg16.flow_from_directory("/content/train/",
                                                            target_size=(224, 224),
                                                            batch_size=BATCH_SIZE,
                                                            class_mode='categorical',
                                                            classes=classes,
                                                            shuffle=True,
                                                            seed=42)

print("Valid")
validgen_vgg16 = valid_generator_vgg16.flow_from_directory("/content/valid/",
                                                            target_size=(224, 224),
                                                            batch_size=BATCH_SIZE,
                                                            class_mode='categorical',
                                                            classes=classes,
                                                            shuffle=True,
                                                            seed=42)

print("Test")
testgen_vgg16 = test_generator_vgg16.flow_from_directory("/content/test/",
                                                           target_size=(224, 224),
                                                           batch_size=BATCH_SIZE,
                                                           class_mode='categorical',
                                                           classes=classes,
                                                           shuffle=True,
                                                           seed=42)

print("Resnet")
print("DataAug")

```

```

traingen_dataaug_resnet =
train_generator_dataaug_resnet.flow_from_directory("/content/train/",
                                                    target_size=(224, 224),
                                                    batch_size=BATCH_SIZE,
                                                    class_mode='categorical',
                                                    classes=classes,
                                                    shuffle=True,
                                                    seed=42)

print("No DataAug")
VGG16
DataAug
Found 476 images belonging to 4 classes.
No DataAug
Found 476 images belonging to 4 classes.
Valid
Found 117 images belonging to 4 classes.
Test
Found 371 images belonging to 4 classes.
Resnet
DataAug
Found 476 images belonging to 4 classes.
No DataAug
Found 476 images belonging to 4 classes.
Valid
Found 117 images belonging to 4 classes.
Test
Found 371 images belonging to 4 classes.

[9]: # # Modelo VGG16 sem top

vgg16_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,
224, 3))
# não treinar os pesos existentes
for layer in vgg16_model.layers:
    layer.trainable = False
vgg16_model.summary()

Model: "vgg16"

```

Layer	Type	Output Shape	Param #
input_layer	InputLayer	(None, 224, 224, 3)	0
block1_conv1	Conv2D	(None, 224, 224, 64)	1,792
block1_conv2	Conv2D	(None, 224, 224, 64)	36,928
block1_pool	MaxPooling2D	(None, 112, 112, 64)	0

block2_conv1	Conv2D	(None, 112, 112, 128)	73,856
block2_conv2	Conv2D	(None, 112, 112, 128)	147,584
block2_pool	MaxPooling2D	(None, 56, 56, 128)	0
block3_conv1	Conv2D	(None, 56, 56, 256)	295,168
block3_conv2	Conv2D	(None, 56, 56, 256)	590,080
block3_conv3	Conv2D	(None, 56, 56, 256)	590,080
block3_pool	MaxPooling2D	(None, 28, 28, 256)	0
block4_conv1	Conv2D	(None, 28, 28, 512)	1,180,160
block4_conv2	Conv2D	(None, 28, 28, 512)	2,359,808
block4_conv3	Conv2D	(None, 28, 28, 512)	2,359,808
block4_pool	MaxPooling2D	(None, 14, 14, 512)	0
block5_conv1	Conv2D	(None, 14, 14, 512)	2,359,808
block5_conv2	Conv2D	(None, 14, 14, 512)	2,359,808
block5_conv3	Conv2D	(None, 14, 14, 512)	2,359,808
block5_pool	MaxPooling2D	(None, 7, 7, 512)	0

Total params: 14,714,688 (56.13 MB)

Trainable params: 0 (0.00 B)

30 Non-trainable params: 14,714,688 (56.13 MB)

[10]: # Definindo as novas camadas

```
x = vgg16_model.output
x = Flatten()(x) # Camada Flatten para ser aplicada na saída da camada convolucional
x = Dense(256, activation='relu')(x) # Camada totalmente conectada com 256 neurônios
x = Dense(len(classes), activation='softmax')(x) # Camada de saída para 4 classes com ativação softmax
# Define the new model
vgg16_model = Model(inputs=vgg16_model.input, outputs=x)
vgg16_model.summary()
```

Model: "functional"

Layer	Type	Output Shape	Param #
input_layer	InputLayer	(None, 224, 224, 3)	0
block1_conv1	Conv2D	(None, 224, 224, 64)	1,792
block1_conv2	Conv2D	(None, 224, 224, 64)	36,928
block1_pool	MaxPooling2D	(None, 112, 112, 64)	0
block2_conv1	Conv2D	(None, 112, 112, 128)	73,856
block2_conv2	Conv2D	(None, 112, 112, 128)	147,584
block2_pool	MaxPooling2D	(None, 56, 56, 128)	0
block3_conv1	Conv2D	(None, 56, 56, 256)	295,168
block3_conv2	Conv2D	(None, 56, 56, 256)	590,080
block3_conv3	Conv2D	(None, 56, 56, 256)	590,080
block3_pool	MaxPooling2D	(None, 28, 28, 256)	0
block4_conv1	Conv2D	(None, 28, 28, 512)	1,180,160
block4_conv2	Conv2D	(None, 28, 28, 512)	2,359,808
block4_conv3	Conv2D	(None, 28, 28, 512)	2,359,808
block4_pool	MaxPooling2D	(None, 14, 14, 512)	0
block5_conv1	Conv2D	(None, 14, 14, 512)	2,359,808
block5_conv2	Conv2D	(None, 14, 14, 512)	2,359,808
block5_conv3	Conv2D	(None, 14, 14, 512)	2,359,808
block5_pool	MaxPooling2D	(None, 7, 7, 512)	0
flatten	Flatten	(None, 25088)	0
dense	Dense	(None, 256)	6,422,784
dense_1	Dense	(None, 4)	1,028

Total params: 21,138,500 (80.64 MB)

Trainable params: 6,423,812 (24.50 MB)

Non-trainable params: 14,714,688 (56.13 MB)

[11]: #Copia o modelo anterior para treinar com data augmentation

```
vgg16_model_dataaug = Model(inputs=vgg16_model.input, outputs=vgg16_model.  
output)
```

[12]: #Modelo ResNet50

A opção include_top=False não inclui as camadas de aprendizado da rede original

Utiliza os pesos treinados na base imagenet

```
resnet_model = ResNet50(input_shape=(224,224,3), weights='imagenet',  
include_top=False)
```

não treinar os pesos existentes

```
for layer in resnet_model.layers:  
    layer.trainable = False
```

[13]: resnet_model.summary()

Model: "resnet50"

Layer	Type	Output Shape	Param #	Connected to
input_layer_1	InputLayer	(None, 224, 224, 3)	0	-
conv1_pad	ZeroPadding2D	(None, 230, 230, 3)	0	input_layer_1[0][0]
conv1_conv	Conv2D	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn	BatchNormalization	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu	Activation	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad	ZeroPadding2D	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool	MaxPooling2D	(None, 56, 56, 64)	0	pool1_pad[0][0]

conv2_block1_1_conv	Conv2D	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn	BatchNormalization	(None, 56, 56, 64)	256	conv2_block1_1_conv[0]...
conv2_block1_1_relu	Activation	(None, 56, 56, 64)	0	conv2_block1_1_bn[0][0]...
conv2_block1_2_conv	Conv2D	(None, 56, 56, 64)	36,928	conv2_block1_1_relu[0]...
conv2_block1_2_bn	BatchNormalization	(None, 56, 56, 64)	256	conv2_block1_2_conv[0]...
conv2_block1_2_relu	Activation	(None, 56, 56, 64)	0	conv2_block1_2_bn[0][0]...
conv2_block1_0_conv	Conv2D	(None, 56, 56, 256)	16,640	pool1_pool[0][0]
conv2_block1_3_conv	Conv2D	(None, 56, 56, 256)	16,640	conv2_block1_2_relu[0]...
conv2_block1_0_bn	BatchNormalization	(None, 56, 56, 256)	1,024	conv2_block1_0_conv[0]...
conv2_block1_3_bn	BatchNormalization	(None, 56, 56, 256)	1,024	conv2_block1_3_conv[0]...
conv2_block1_add	Add	(None, 56, 56, 256)	0	conv2_block1_0_bn[0][0] + conv2_block1_3_bn[0][0]...
conv2_block1_out	Activation	(None, 56, 56, 256)	0	conv2_block1_add[0][0]
conv2_block2_1_conv	Conv2D	(None, 56, 56, 64)	16,448	conv2_block1_out[0][0]
conv2_block2_1_bn	BatchNormalization	(None, 56, 56, 64)	256	conv2_block2_1_conv[0]...
conv2_block2_1_relu	Activation	(None, 56, 56, 64)	0	conv2_block2_1_bn[0][0]...
conv2_block2_2_conv	Conv2D	(None, 56, 56, 64)	36,928	conv2_block2_1_relu[0]...

conv2_block2_2_bn	BatchNormalization	(None, 56, 56, 64)	256	conv2_block2_2_conv[0 ...]
conv2_block2_2_relu	Activation	(None, 56, 56, 64)	0	conv2_block2_2_bn[0] [...]
conv2_block2_3_conv	Conv2D	(None, 56, 56, 256)	16,640	conv2_block2_2_relu[0 ...]
conv2_block2_3_bn	BatchNormalization	(None, 56, 56, 256)	1,024	conv2_block2_3_conv[0 ...]
conv2_block2_add	Add	(None, 56, 56, 256)	0	conv2_block1_out[0] [0 ...] + conv2_block2_3_bn[0] [...]
conv2_block2_out	Activation	(None, 56, 56, 256)	0	conv2_block2_add[0] [0]
conv2_block3_1_conv	Conv2D	(None, 56, 56, 64)	16,448	conv2_block2_out[0] [0]
conv2_block3_1_bn	BatchNormalization	(None, 56, 56, 64)	256	conv2_block3_1_conv[0 ...]
conv2_block3_1_relu	Activation	(None, 56, 56, 64)	0	conv2_block3_1_bn[0] [...]
conv2_block3_2_conv	Conv2D	(None, 56, 56, 64)	36,928	conv2_block3_1_relu[0 ...]
conv2_block3_2_bn	BatchNormalization	(None, 56, 56, 64)	256	conv2_block3_2_conv[0 ...]
conv2_block3_2_relu	Activation	(None, 56, 56, 64)	0	conv2_block3_2_bn[0] [...]
conv2_block3_3_conv	Conv2D	(None, 56, 56, 256)	16,640	conv2_block3_2_relu[0 ...]
conv2_block3_3_bn	BatchNormalization	(None, 56, 56, 256)	1,024	conv2_block3_3_conv[0 ...]
conv2_block3_add	Add	(None, 56, 56, 256)	0	conv2_block2_out[0] [0 ...] + conv2_block3_3_bn[0] [...]
conv2_block3_out	Activation	(None, 56, 56, 256)	0	conv2_block3_add[0] [0]

conv3_block1_1_conv	Conv2D	(None, 28, 28, 128)	32,896	conv2_block3_out[0][0]
conv3_block1_1_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block1_1_conv[0]...
conv3_block1_1_relu	Activation	(None, 28, 28, 128)	0	conv3_block1_1_bn[0][0]...
conv3_block1_2_conv	Conv2D	(None, 28, 28, 128)	147,584	conv3_block1_1_relu[0]...
conv3_block1_2_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block1_2_conv[0]...
conv3_block1_2_relu	Activation	(None, 28, 28, 128)	0	conv3_block1_2_bn[0][0]...
conv3_block1_0_conv	Conv2D	(None, 28, 28, 512)	131,584	conv2_block3_out[0][0]
conv3_block1_3_conv	Conv2D	(None, 28, 28, 512)	66,048	conv3_block1_2_relu[0]...
conv3_block1_0_bn	BatchNormalization	(None, 28, 28, 512)	2,048	conv3_block1_0_conv[0]...
conv3_block1_3_bn	BatchNormalization	(None, 28, 28, 512)	2,048	conv3_block1_3_conv[0]...
conv3_block1_add	Add	(None, 28, 28, 512)	0	conv3_block1_0_bn[0][0] + conv3_block1_3_bn[0][0]...
conv3_block1_out	Activation	(None, 28, 28, 512)	0	conv3_block1_add[0][0]
conv3_block2_1_conv	Conv2D	(None, 28, 28, 128)	65,664	conv3_block1_out[0][0]
conv3_block2_1_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block2_1_conv[0]...
conv3_block2_1_relu	Activation	(None, 28, 28, 128)	0	conv3_block2_1_bn[0][0]...
conv3_block2_2_conv	Conv2D	(None, 28, 28, 128)	147,584	conv3_block2_1_relu[0]...

conv3_block2_2_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block2_2_conv[0 ...]
conv3_block2_2_relu	Activation	(None, 28, 28, 128)	0	conv3_block2_2_bn[0] [...]
conv3_block2_3_conv	Conv2D	(None, 28, 28, 512)	66,048	conv3_block2_2_relu[0 ...]
conv3_block2_3_bn	BatchNormalization	(None, 28, 28, 512)	2,048	conv3_block2_3_conv[0 ...]
conv3_block2_add	Add	(None, 28, 28, 512)	0	conv3_block1_out[0] [0 ...] + conv3_block2_3_bn[0] [...]
conv3_block2_out	Activation	(None, 28, 28, 512)	0	conv3_block2_add[0] [0]
conv3_block3_1_conv	Conv2D	(None, 28, 28, 128)	65,664	conv3_block2_out[0] [0]
conv3_block3_1_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block3_1_conv[0 ...]
conv3_block3_1_relu	Activation	(None, 28, 28, 128)	0	conv3_block3_1_bn[0] [...]
conv3_block3_2_conv	Conv2D	(None, 28, 28, 128)	147,584	conv3_block3_1_relu[0 ...]
conv3_block3_2_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block3_2_conv[0 ...]
conv3_block3_2_relu	Activation	(None, 28, 28, 128)	0	conv3_block3_2_bn[0] [...]
conv3_block3_3_conv	Conv2D	(None, 28, 28, 512)	66,048	conv3_block3_2_relu[0 ...]
conv3_block3_3_bn	BatchNormalization	(None, 28, 28, 512)	2,048	conv3_block3_3_conv[0 ...]
conv3_block3_add	Add	(None, 28, 28, 512)	0	conv3_block2_out[0] [0 ...] + conv3_block3_3_bn[0] [...]
conv3_block3_out	Activation	(None, 28, 28, 512)	0	conv3_block3_add[0] [0]

conv3_block4_1_conv	Conv2D	(None, 28, 28, 128)	65,664	conv3_block3_out[0][0]
conv3_block4_1_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block4_1_conv[0]...
conv3_block4_1_relu	Activation	(None, 28, 28, 128)	0	conv3_block4_1_bn[0][0]...
conv3_block4_2_conv	Conv2D	(None, 28, 28, 128)	147,584	conv3_block4_1_relu[0]...
conv3_block4_2_bn	BatchNormalization	(None, 28, 28, 128)	512	conv3_block4_2_conv[0]...
conv3_block4_2_relu	Activation	(None, 28, 28, 128)	0	conv3_block4_2_bn[0][0]...
conv3_block4_3_conv	Conv2D	(None, 28, 28, 512)	66,048	conv3_block4_2_relu[0]...
conv3_block4_3_bn	BatchNormalization	(None, 28, 28, 512)	2,048	conv3_block4_3_conv[0]...
conv3_block4_add	Add	(None, 28, 28, 512)	0	conv3_block3_out[0][0]... + conv3_block4_3_bn[0][0]...
conv3_block4_out	Activation	(None, 28, 28, 512)	0	conv3_block4_add[0][0]
conv4_block1_1_conv	Conv2D	(None, 14, 14, 256)	131,328	conv3_block4_out[0][0]
conv4_block1_1_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block1_1_conv[0]...
conv4_block1_1_relu	Activation	(None, 14, 14, 256)	0	conv4_block1_1_bn[0][0]...
conv4_block1_2_conv	Conv2D	(None, 14, 14, 256)	590,080	conv4_block1_1_relu[0]...
conv4_block1_2_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block1_2_conv[0]...
conv4_block1_2_relu	Activation	(None, 14, 14, 256)	0	conv4_block1_2_bn[0][0]...

conv4_block1_0_conv	Conv2D	(None, 14, 14, 1024)	525,312	conv3_block4_out[0][0]
conv4_block1_3_conv	Conv2D	(None, 14, 14, 1024)	263,168	conv4_block1_2_relu[0...]
conv4_block1_0_bn	BatchNormalization	(None, 14, 14, 1024)	4,096	conv4_block1_0_conv[0...]
conv4_block1_3_bn	BatchNormalization	(None, 14, 14, 1024)	4,096	conv4_block1_3_conv[0...]
conv4_block1_add	Add	(None, 14, 14, 1024)	0	conv4_block1_0_bn[0][...] + conv4_block1_3_bn[0][...]
conv4_block1_out	Activation	(None, 14, 14, 1024)	0	conv4_block1_add[0][0]
conv4_block2_1_conv	Conv2D	(None, 14, 14, 256)	262,400	conv4_block1_out[0][0]
conv4_block2_1_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block2_1_conv[0...]
conv4_block2_1_relu	Activation	(None, 14, 14, 256)	0	conv4_block2_1_bn[0][...]
conv4_block2_2_conv	Conv2D	(None, 14, 14, 256)	590,080	conv4_block2_1_relu[0...]
conv4_block2_2_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block2_2_conv[0...]
conv4_block2_2_relu	Activation	(None, 14, 14, 256)	0	conv4_block2_2_bn[0][...]
conv4_block2_3_conv	Conv2D	(None, 14, 14, 1024)	263,168	conv4_block2_2_relu[0...]
conv4_block2_3_bn	BatchNormalization	(None, 14, 14, 1024)	4,096	conv4_block2_3_conv[0...]
conv4_block2_add	Add	(None, 14, 14, 1024)	0	conv4_block1_out[0][0...] + conv4_block2_3_bn[0][...]
conv4_block2_out	Activation	(None, 14, 14, 1024)	0	conv4_block2_add[0][0]

conv4_block3_1_conv	Conv2D	(None, 14, 14, 256)	262,400	conv4_block2_out[0][0]
conv4_block3_1_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block3_1_conv[0]...
conv4_block3_1_relu	Activation	(None, 14, 14, 256)	0	conv4_block3_1_bn[0][0]...
conv4_block3_2_conv	Conv2D	(None, 14, 14, 256)	590,080	conv4_block3_1_relu[0]...
conv4_block3_2_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block3_2_conv[0]...
conv4_block3_2_relu	Activation	(None, 14, 14, 256)	0	conv4_block3_2_bn[0][0]...
conv4_block3_3_conv	Conv2D	(None, 14, 14, 1024)	263,168	conv4_block3_2_relu[0]...
conv4_block3_3_bn	BatchNormalization	(None, 14, 14, 1024)	4,096	conv4_block3_3_conv[0]...
conv4_block3_add	Add	(None, 14, 14, 1024)	0	conv4_block2_out[0][0]... + conv4_block3_3_bn[0][0]...
conv4_block3_out	Activation	(None, 14, 14, 1024)	0	conv4_block3_add[0][0]
conv4_block4_1_conv	Conv2D	(None, 14, 14, 256)	262,400	conv4_block3_out[0][0]
conv4_block4_1_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block4_1_conv[0]...
conv4_block4_1_relu	Activation	(None, 14, 14, 256)	0	conv4_block4_1_bn[0][0]...
conv4_block4_2_conv	Conv2D	(None, 14, 14, 256)	590,080	conv4_block4_1_relu[0]...
conv4_block4_2_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block4_2_conv[0]...
conv4_block4_2_relu	Activation	(None, 14, 14, 256)	0	conv4_block4_2_bn[0][0]...

conv4_block4_3_conv	Conv2D	(None, 14, 14, 1024)	263,168	conv4_block4_2_relu[0 ...]
conv4_block4_3_bn	BatchNormalization	(None, 14, 14, 1024)	4,096	conv4_block4_3_conv[0 ...]
conv4_block4_add	Add	(None, 14, 14, 1024)	0	conv4_block3_out[0][0 ...] + conv4_block4_3_bn[0][...]
conv4_block4_out	Activation	(None, 14, 14, 1024)	0	conv4_block4_add[0][0]
conv4_block5_1_conv	Conv2D	(None, 14, 14, 256)	262,400	conv4_block4_out[0][0]
conv4_block5_1_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block5_1_conv[0 ...]
conv4_block5_1_relu	Activation	(None, 14, 14, 256)	0	conv4_block5_1_bn[0][...]
conv4_block5_2_conv	Conv2D	(None, 14, 14, 256)	590,080	conv4_block5_1_relu[0 ...]
conv4_block5_2_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block5_2_conv[0 ...]
conv4_block5_2_relu	Activation	(None, 14, 14, 256)	0	conv4_block5_2_bn[0][...]
conv4_block5_3_conv	Conv2D	(None, 14, 14, 1024)	263,168	conv4_block5_2_relu[0 ...]
conv4_block5_3_bn	BatchNormalization	(None, 14, 14, 1024)	4,096	conv4_block5_3_conv[0 ...]
conv4_block5_add	Add	(None, 14, 14, 1024)	0	conv4_block4_out[0][0 ...] + conv4_block5_3_bn[0][...]
conv4_block5_out	Activation	(None, 14, 14, 1024)	0	conv4_block5_add[0][0]
conv4_block6_1_conv	Conv2D	(None, 14, 14, 256)	262,400	conv4_block5_out[0][0]
conv4_block6_1_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block6_1_conv[0 ...]

conv4_block6_1_relu	Activation	(None, 14, 14, 256)	0	conv4_block6_1_bn[0][...]
conv4_block6_2_conv	Conv2D	(None, 14, 14, 256)	590,080	conv4_block6_1_relu[0][...]
conv4_block6_2_bn	BatchNormalization	(None, 14, 14, 256)	1,024	conv4_block6_2_conv[0][...]
conv4_block6_2_relu	Activation	(None, 14, 14, 256)	0	conv4_block6_2_bn[0][...]
conv4_block6_3_conv	Conv2D	(None, 14, 14, 1024)	263,168	conv4_block6_2_relu[0][...]
conv4_block6_3_bn	BatchNormalization	(None, 14, 14, 1024)	4,096	conv4_block6_3_conv[0][...]
conv4_block6_add	Add	(None, 14, 14, 1024)	0	conv4_block5_out[0][0][...] + conv4_block6_3_bn[0][...]
conv4_block6_out	Activation	(None, 14, 14, 1024)	0	conv4_block6_add[0][0]
conv5_block1_1_conv	Conv2D	(None, 7, 7, 512)	524,800	conv4_block6_out[0][0]
conv5_block1_1_bn	BatchNormalization	(None, 7, 7, 512)	2,048	conv5_block1_1_conv[0][...]
conv5_block1_1_relu	Activation	(None, 7, 7, 512)	0	conv5_block1_1_bn[0][...]
conv5_block1_2_conv	Conv2D	(None, 7, 7, 512)	2,359,808	conv5_block1_1_relu[0][...]
conv5_block1_2_bn	BatchNormalization	(None, 7, 7, 512)	2,048	conv5_block1_2_conv[0][...]
conv5_block1_2_relu	Activation	(None, 7, 7, 512)	0	conv5_block1_2_bn[0][...]
conv5_block1_0_conv	Conv2D	(None, 7, 7, 2048)	2,099,200	conv4_block6_out[0][0]
conv5_block1_3_conv	Conv2D	(None, 7, 7, 2048)	1,050,624	conv5_block1_2_relu[0][...]

conv5_block1_0_bn	BatchNormalization	(None, 7, 7, 2048)	8,192	conv5_block1_0_conv[0 ...]
conv5_block1_3_bn	BatchNormalization	(None, 7, 7, 2048)	8,192	conv5_block1_3_conv[0 ...]
conv5_block1_add	Add	(None, 7, 7, 2048)	0	conv5_block1_0_bn[0] [...] + conv5_block1_3_bn[0] [...]
conv5_block1_out	Activation	(None, 7, 7, 2048)	0	conv5_block1_add[0] [0]
conv5_block2_1_conv	Conv2D	(None, 7, 7, 512)	1,049,088	conv5_block1_out[0] [0]
conv5_block2_1_bn	BatchNormalization	(None, 7, 7, 512)	2,048	conv5_block2_1_conv[0 ...]
conv5_block2_1_relu	Activation	(None, 7, 7, 512)	0	conv5_block2_1_bn[0] [...]
conv5_block2_2_conv	Conv2D	(None, 7, 7, 512)	2,359,808	conv5_block2_1_relu[0 ...]
conv5_block2_2_bn	BatchNormalization	(None, 7, 7, 512)	2,048	conv5_block2_2_conv[0 ...]
conv5_block2_2_relu	Activation	(None, 7, 7, 512)	0	conv5_block2_2_bn[0] [...]
conv5_block2_3_conv	Conv2D	(None, 7, 7, 2048)	1,050,624	conv5_block2_2_relu[0 ...]
conv5_block2_3_bn	BatchNormalization	(None, 7, 7, 2048)	8,192	conv5_block2_3_conv[0 ...]
conv5_block2_add	Add	(None, 7, 7, 2048)	0	conv5_block1_out[0] [0 ...] + conv5_block2_3_bn[0] [...]
conv5_block2_out	Activation	(None, 7, 7, 2048)	0	conv5_block2_add[0] [0]
conv5_block3_1_conv	Conv2D	(None, 7, 7, 512)	1,049,088	conv5_block2_out[0] [0]
conv5_block3_1_bn	BatchNormalization	(None, 7, 7, 512)	2,048	conv5_block3_1_conv[0 ...]

conv5_block3_1_relu	Activation	(None, 7, 7, 512)	0	conv5_block3_1_bn[0][...]
conv5_block3_2_conv	Conv2D	(None, 7, 7, 512)	2,359,808	conv5_block3_1_relu[0][...]
conv5_block3_2_bn	BatchNormalization	(None, 7, 7, 512)	2,048	conv5_block3_2_conv[0][...]
conv5_block3_2_relu	Activation	(None, 7, 7, 512)	0	conv5_block3_2_bn[0][...]
conv5_block3_3_conv	Conv2D	(None, 7, 7, 2048)	1,050,624	conv5_block3_2_relu[0][...]
conv5_block3_3_bn	BatchNormalization	(None, 7, 7, 2048)	8,192	conv5_block3_3_conv[0][...]
conv5_block3_add	Add	(None, 7, 7, 2048)	0	conv5_block2_out[0][0][...] + conv5_block3_3_bn[0][...]
conv5_block3_out	Activation	(None, 7, 7, 2048)	0	conv5_block3_add[0][0]

Total params: 23,989,124 (91.51 MB)

Trainable params: 401,412 (1.53 MB)

Non-trainable params: 23,587,712 (89.98 MB)

[16]: *#Copia o modelo para treinar com data augmentation*

```
resnet_model_dataaug = Model(inputs=resnet_model.input,
                             outputs=resnet_model.output)
```

[17]: *from keras.optimizers import RMSprop*

```
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras
```

```
def train(training, validgen, model_tl, modelname):
    steps_per_epoch = training.samples // BATCH_SIZE
    val_steps = validgen.samples // BATCH_SIZE
    n_epochs = 100
    optimizer = RMSprop(learning_rate=0.0001)
    model_tl.compile(loss='categorical_crossentropy', optimizer=optimizer,
                    metrics=['accuracy'])
    # Salva o modelo Keras após cada época, porém só o de melhor resultado
    checkpointer =
    ModelCheckpoint(filepath=f'img_model_{modelname}.weights.best.keras',
```

```

        verbose=1,
        save_best_only=True)
# Para o treinamento para prevenir o overfitting
# Não utilizei aqui, pois queria que rodasse todas as 30 épocas
early_stop = EarlyStopping(monitor='val_loss',
                           patience=10,
                           restore_best_weights=True,
                           mode='min')

# Treinamento do Modelo
history_tl = model_tl.fit(traingen,
                          epochs=n_epochs,
                          steps_per_epoch=steps_per_epoch,
                          validation_data=validgen,
                          validation_steps=val_steps,
                          callbacks=[checkpointer, PlotLossesKeras()],
                          #callbacks=[early_stop, checkpointer,
PlotLossesKeras()],
                          verbose=False)
return model_tl, history_tl

[18]: # VGG16 com data augmentation

trained_vgg16_dataaug = train(traingen_dataaug_vgg16, validgen_vgg16,
vgg16_model_dataaug, "traingen_dataaug_vgg16")

accuracy
      training      (min:    0.493, max:    0.969, cur:    0.938)
  validation      (min:    0.238, max:    0.762, cur:    0.524)
Loss
      training      (min:    0.054, max:    4.626, cur:    0.078)
  validation      (min:    1.728, max:   15.903, cur:    6.541)

[19]: # VGG16 sem data augmentation

trained_vgg16 = train(traingen_vgg16, validgen_vgg16, vgg16_model,
"traingen_vgg16")

accuracy
      training      (min:    0.955, max:    1.000, cur:    1.000)
  validation      (min:    0.524, max:    0.905, cur:    0.762)
Loss
      training      (min:    0.000, max:    0.255, cur:    0.000)
  validation      (min:    0.607, max:    4.448, cur:    1.821)

[20]: # ResNet com data augmentation

trained_resnet_dataaug = train(traingen_dataaug_resnet, validgen_resnet,
resnet_model_dataaug, "traingen_dataaug_resnet")

```

```

accuracy
    training          (min:    0.568, max:    1.000, cur:    0.969)
    validation        (min:    0.381, max:    0.905, cur:    0.524)
Loss
    training          (min:    0.011, max:    1.990, cur:    0.036)
    validation        (min:    0.811, max:    4.066, cur:    2.611)

```

```
[21]: from sklearn.metrics import confusion_matrix
```

```

def calculate_metrics(cm):
    # Calcular a matriz de confusão
    #cm = confusion_matrix(y_val, val_predictions)
    # Inicializar contadores para micro F1-Score
    TP_total = 0
    FP_total = 0
    FN_total = 0
    TN_total = 0
    f1_scores = []
    sensibilities = []
    specificities = []
    # Número de classes
    num_classes = cm.shape[0]
    # Calcular precisão, recall e F1-Score para cada classe
    for i in range(num_classes):
        TP = cm[i, i]
        FP = cm[:, i].sum() - TP # Falsos positivos
        FN = cm[i, :].sum() - TP # Falsos negativos
        TN = cm.sum() - (TP + FP + FN) # Verdadeiros negativos
        # Calcular Precisão e Recall
        precision = TP / (TP + FP) if (TP + FP) > 0 else 0
        recall = TP / (TP + FN) if (TP + FN) > 0 else 0
        sensibilidade = TP / (TP + FN) if (TP + FN) > 0 else 0
        especificidade = TN / (TN + FP) if (TN + FP) > 0 else 0
        f1 = 2 * (precision * recall) / (precision + recall) if (precision +
recall) > 0 else 0
        # Acumular valores
        TP_total += TP
        FP_total += FP
        FN_total += FN
        TN_total += TN
        f1_scores.append(f1)
        sensibilities.append(sensibilidade)
        specificities.append(especificidade)
    accuracy = (TP_total + TN_total) / (TP_total + TN_total + FP_total +
FN_total) if (TP_total + TN_total + FP_total + FN_total) > 0 else 0
    n_classes = [cm[i, :].sum() for i in range(num_classes)] # Número de
instâncias por classe
    # Cálculo do F1-Score Micro
    #precision_micro = TP_total / (TP_total + FP_total) if (TP_total +
FP_total) > 0 else 0

```

```

#recall_micro = TP_total / (TP_total + FN_total) if (TP_total + FN_total)
> 0 else 0
#f1_micro = 2 * (precision_micro * recall_micro) / (precision_micro +
recall_micro) if (precision_micro + recall_micro) > 0 else 0
# Cálculo do F1-Score Weighted
f1_weighted = sum(f1 * n for f1, n in zip(f1_scores, n_classes)) /
sum(n_classes) if sum(n_classes) > 0 else 0
# Sensibilidade ponderada
sensitivity_weighted = sum(sens * n for sens, n in zip(sensibilities,
n_classes)) / sum(n_classes) if sum(n_classes) > 0 else 0
# Especificidade ponderada
specificity_weighted = sum(spec * n for spec, n in zip(specificities,
n_classes)) / sum(n_classes) if sum(n_classes) > 0 else 0
return accuracy, sensitivity_weighted, specificity_weighted, f1_weighted

[23]: trained_resnet[0].predict(testgen_resnet)

[23]: array([[9.87282634e-01, 1.06999648e-18, 8.83815066e-09,
1.27173057e-02],

[6.90599024e-01, 4.13748193e-15, 2.10919060e-09, 3.09400976e-01],

[7.84158647e-01, 2.26755489e-07, 1.25245325e-08, 2.15841204e-01],

...,

[8.28316353e-27, 6.70703358e-29, 1.00000000e+00, 3.06785082e-19],

[1.60936883e-13, 1.87543801e-20, 6.52806702e-05, 9.99934673e-01],

[2.23787396e-15, 9.99999523e-01, 1.54875877e-15, 4.57421493e-07]],

dtype=float32)

[24]: traingen_dataaug_resnet

[24]: <keras.src.legacy.preprocessing.image.DirectoryIterator at
0x7ab968c334c0>

[25]: import pandas as pd

import matplotlib.pyplot as plt

from sklearn.metrics import precision_recall_fscore_support, accuracy_score

def display_results(y_true, y_preds, class_labels):

# Converte para labels das classes para valores discretos se elas forem
valores contínuos

# Assuming y_preds contains probabilities or raw scores

```

```

y_preds = np.argmax(y_preds, axis=1)

# Ensure both y_true and y_preds are of the same data type (e.g., int)
y_true = y_true.astype(int)
y_preds = y_preds.astype(int)

results = pd.DataFrame(precision_recall_fscore_support(y_true, y_preds),

columns=class_labels).T

results.rename(columns={0: 'Precision', 1: 'Recall',

2: 'F-Score', 3: 'Support'}, inplace=True)

results.sort_values(by='F-Score', ascending=False, inplace=True)

global_acc = accuracy_score(y_true, y_preds)

print("Acurácia: {:.2f}%".format(global_acc*100))

return results

def plot_predictions(y_true, y_preds, test_generator, class_indices):
    fig = plt.figure(figsize=(20, 10))

    for i, idx in enumerate(np.random.choice(test_generator.samples, size=20, _
replace=False)):

        ax = fig.add_subplot(4, 5, i + 1, xticks=[], yticks=[])

        ax.imshow(np.squeeze(test_generator[idx]))

        pred_idx = y_preds[idx]

        true_idx = y_true[idx]

        plt.tight_layout()

        ax.set_title("{}\n({})".format(class_indices[pred_idx], _
class_indices[true_idx]),

color=("green" if pred_idx == true_idx else "red"))

```

```

def plot_predictions(y_true, y_preds, test_generator, class_indices):

fig = plt.figure(figsize=(20, 10))

# Get the actual number of samples in the test generator
num_samples = len(test_generator)

# Ensure you only select indices within the valid range
indices = np.random.choice(np.arange(num_samples), size=min(num_samples, 20), replace=False)

for i, idx in enumerate(indices): # Use the valid indices here
ax = fig.add_subplot(4, 5, i + 1, xticks=[], yticks=[])

# Accessing data from the generator might require a different approach
# depending on how your generator is implemented.
# If your generator yields (image, label) pairs, you can use:

80

# image, _ = test_generator[idx]
# ax.imshow(np.squeeze(image))

# However, if you're using ImageDataGenerator.flow_from_directory,
# you need to iterate through the generator to get data:

batch_index = idx // test_generator.batch_size
index_within_batch = idx % test_generator.batch_size

# Iterate to the desired batch
for _ in range(batch_index + 1):
batch_x, batch_y = test_generator.next()

ax.imshow(np.squeeze(batch_x[index_within_batch]))

pred_idx = np.argmax(y_preds[idx]) # Get the predicted class index
true_idx = y_true[idx]

plt.tight_layout()

```

```

ax.set_title("{}\n{}".format(class_indices[pred_idx],
class_indices[true_idx]),

color=("green" if pred_idx == true_idx else "red"))

[26]: import numpy as np

import pandas as pd
from sklearn.metrics import accuracy_score, confusion_matrix
def calculate_metrics(cm):
    # Converte para labels das classes para valores discretos se elas forem
    valores contínuos
    #y_preds = np.argmax(y_preds, axis=1)
    # Garante que ambos y_true e y_preds são do mesmo tipo de dados (e.g.,
    int)
    #y_true = y_true.astype(int)
    #y_preds = y_preds.astype(int)
    # Calcula a matriz de confusão
    #cm = confusion_matrix(y_true, y_preds)
    #print("Matriz de Confusão:")
    #print(cm)
    # Inicializa listas para armazenar as métricas por classe
    precisions = []
    recalls = []
    f1_scores = []
    specificities = []
    supports = []
    TP_total = 0
    FP_total = 0
    FN_total = 0
    TN_total = 0
    # Número de classes
    num_classes = cm.shape[0]
    # Calcula métricas para cada classe
    for i in range(num_classes):
        tp = cm[i, i]
        fn = np.sum(cm[i, :]) - tp
        fp = np.sum(cm[:, i]) - tp
        tn = np.sum(cm) - (tp + fn + fp)
        # Evita divisão por zero
        precision = tp / (tp + fp) if (tp + fp) > 0 else 0
        recall = tp / (tp + fn) if (tp + fn) > 0 else 0
        specificity = tn / (tn + fp) if (tn + fp) > 0 else 0
        f1 = 2 * precision * recall / (precision + recall) if (precision +
recall) > 0 else 0
        support = np.sum(cm[i, :]) # Número de amostras reais da classe i
        # Acumular valores
        TP_total += tp
        FP_total += fp

```

```

        FN_total += fn
        TN_total += tn
        # Adiciona as métricas calculadas às listas
        precisions.append(precision)
        recalls.append(recall)
        f1_scores.append(f1)
        specificities.append(specificity)
        supports.append(support)
    # Calcula métricas ponderadas
    f1_weighted = np.average(f1_scores, weights=supports)
    recall_weighted = np.average(recalls, weights=supports)
    specificity_weighted = np.average(specificities, weights=supports)
    accuracy = (TP_total + TN_total) / (TP_total + TN_total + FP_total +
FN_total) if (TP_total + TN_total + FP_total + FN_total) > 0 else 0
    # Acurácia global
    print("\nAcurácia: {:.2f}%".format(accuracy * 100))
    print(f"F1-Score (Weighted): {f1_weighted:.2f}")
    print(f"Sensibilidade (Recall Weighted): {recall_weighted:.2f}")
    print(f"Especificidade (Weighted): {specificity_weighted:.2f}")
    return accuracy, recall_weighted, specificity_weighted, f1_weighted
def metricas(tipo, df_resultados, y_true, y_preds):
    # Converte para labels das classes para valores discretos se elas forem
valores contínuos
    y_preds = np.argmax(y_preds, axis=1)
    # Garante que ambos y_true e y_preds são do mesmo tipo de dados (e.g.,
int)
    y_true = y_true.astype(int)
    y_preds = y_preds.astype(int)
    cm = confusion_matrix(y_true, y_preds)
    accuracy, sensibility_weighted, specificity_weighted, f1_weighted =
calculate_metrics(cm)
    data = { 'Tipo' : tipo,
            'Acurácia': [accuracy],
            'Sensibilidade Ponderada': [sensibility_weighted],
            'Especificidade Ponderada': [specificity_weighted],
            'F1 Ponderada': [f1_weighted]}
    df_resultados = pd.concat([df_resultados, pd.DataFrame(data)],
ignore_index=True)
    return df_resultados

[27]: # Carrega os melhores modelos

resnet_model.load_weights('/content/img_model_traingen_resnet.weights.best.ke
ras')
resnet_model_dataaug.load_weights('/content/img_model_traingen_dataaug_resnet
.weights.best.keras')
vgg16_model.load_weights('/content/img_model_traingen_vgg16.weights.best.kera
s')
vgg16_model_dataaug.load_weights('/content/img_model_traingen_dataaug_vgg16.w
eights.best.keras')

[28]: i#Preparando os dataframes para receber os resultados

```

```

df_resultados_test = pd.DataFrame()
# Calculando as métricas
#Resnet
predicted_classes = resnet_model.predict(testgen_resnet)
class_indices = traingen_resnet.class_indices
true_classes = testgen_resnet.classes
#labels = traingen_resnet.class_indices
df_resultados_test = metricas("resnet", df_resultados_test, true_classes,
predicted_classes)
#Resnet Data Augmentation
predicted_classes = resnet_model_dataaug.predict(testgen_resnet)
class_indices = traingen_resnet.class_indices
true_classes = testgen_resnet.classes
df_resultados_test = metricas("resnet_dataaug", df_resultados_test,
true_classes, predicted_classes)
#Vgg16 Data
predicted_classes = vgg16_model.predict(testgen_vgg16)
class_indices = traingen_vgg16.class_indices
true_classes = testgen_vgg16.classes
df_resultados_test = metricas("vgg16", df_resultados_test, true_classes,
predicted_classes)
#Vgg16 Data Augmentation
predicted_classes = vgg16_model_dataaug.predict(testgen_vgg16)
class_indices = traingen_vgg16.class_indices
true_classes = testgen_vgg16.classes
df_resultados_test = metricas("vgg16_dataaug", df_resultados_test,
true_classes, predicted_classes)

```

Acurácia: 62.80%

F1-Score (Weighted): 0.25

Sensibilidade (Recall Weighted): 0.26

Especificidade (Weighted): 0.75

12/12

11s 688ms/step

Acurácia: 63.21%

F1-Score (Weighted): 0.26

Sensibilidade (Recall Weighted): 0.26

Especificidade (Weighted): 0.75

Acurácia: 64.29%

F1-Score (Weighted): 0.28

Sensibilidade (Recall Weighted): 0.29

Especificidade (Weighted): 0.76

12/12

3s 181ms/step

Acurácia: 62.67%

F1-Score (Weighted): 0.25

Sensibilidade (Recall Weighted): 0.25

Especificidade (Weighted): 0.75

[29]: #Ordenando os resultados de teste por 'F1 Ponderada'

```
df_resultados_test = df_resultados_test.sort_values('F1 Ponderada',
ascending=False)
```

```
df_resultados_test
```

[29]: Tipo Acurária Sensibilidade Ponderada \

2 vgg16 0.642857 0.285714

1 resnet_dataaug 0.632075 0.264151

0 resnet 0.628032 0.256065

3 vgg16_dataaug 0.626685 0.253369

Especificidade Ponderada F1 Ponderada

2 0.761511 0.284902

1 0.753735 0.264135

0 0.750855 0.254400

3 0.750717 0.252442

Conclusão

O modelo que melhor métricas apresentou no conjunto de teste foi o VGG16 sem Data Augmentation. Logo após temos o Resnet50 com Data Augmentation. Seguido pelo Resnet50 Sem Data Augmentation e por último, mas com métricas muito semelhantes, o VGG16 com Data Augmentation.

Os valores das métricas foi muito baixo, o que atribuímos ao dataset possuir poucos exemplos de cada classe para cada paciente, assim o treinamento e validação durante o treinamento não foram adequados. O dataset de treinamento possuía 20 pacientes apenas e dividindo em 80%/20%, conforme o que solicitava o enunciado do trabalho, acabamos com 16 pacientes para treinamento e 4 para validação. Como fizemos a separação aleatoriamente, ficamos com apenas 2 classes de exemplo no dataset de validação.

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

O uso das novas tecnologias de Inteligência Artificial (IA), como o ChatGPT, apresenta um grande salto na evolução da chamada sociedade informacional, possibilitando acesso rápido à informação e conhecimento sobre diversos temas. É natural que uma ferramenta tão poderosa fosse também utilizada de maneira que levasse a questionamentos éticos e legais por diversas partes. Esse tipo de debate deve se tornar a tônica das futuras discussões envolvendo o uso da IA.

O presente trabalho apresenta uma reflexão sobre as implicações legais e éticas na aplicação do ChatGPT em alguns cenários específicos como na educação, atendimento ao cliente, criação de conteúdo, privacidade, consentimento de dados e tomada de decisões. Além de buscar soluções, o artigo visa ser um chamamento à reflexão sobre os valores primordiais das interações sociais humanas e até que ponto os ganhos devem justificar os meios utilizados para o seu atingimento. Os benefícios também não devem ser ignorados, mas é preciso entender as implicações, onde, como, e a que custo queremos chegar obtê-los. Como toda nova ferramenta tecnológica, é natural que surjam essas questões e é importante que elas sejam debatidas de forma ampla e aberta para que sejamos capazes de entender os caminhos percorridos e propor eventuais ajustes ao longo desses caminhos.

Palavras-chave: ChatGPT, ética, inteligência artificial, vieses algorítmicos, privacidade.

ABSTRACT

The use of new Artificial Intelligence (AI) technologies, such as ChatGPT, represents a significant leap in the evolution of the so-called information society, enabling rapid access to information and knowledge on a wide range of topics. It is natural that such a powerful tool would also be used in ways that raise ethical and legal questions from various parties. This type of debate should become the focus of future discussions involving the use of AI.

This paper presents a reflection on the legal and ethical implications of applying ChatGPT in specific scenarios such as education, customer service, content creation, privacy, data consent, and decision-making. In addition to seeking solutions, the article aims to be a call for reflection on the fundamental values of human social interactions and to what extent the benefits should justify the means used to achieve them. The benefits should not be ignored either, but it is necessary to understand the implications, where, how, and at what cost we want to achieve them.

As with any new technological tool, it is natural that these questions arise, and it is important that they are debated widely and openly so that we are able to understand the paths taken and propose any necessary adjustments along the way.

Keywords: ChatGPT, ethics, artificial intelligence, algorithmic biases, privacy.

1. INTRODUÇÃO

A Inteligência Artificial (IA) é cada vez mais presente na sociedade, impactando diversos setores, como o trabalho, a educação e a comunicação. O ChatGPT foi uma das ferramentas de IA que mais tiveram destaque nos últimos anos por sua capacidade de gerar textos coerentes, contextualizados e com informações relevantes.

Diversas atividades manuais e demoradas, agora podem ser executadas de forma rápida e relativamente automatizada com o ChatGPT. Mas todo benefício traz consigo potenciais problemas. Uma das preocupações, cada vez mais relevantes, é quanto aos impactos do seu uso em questões legais e éticas dentro das sociedades. Como lidar com assuntos que envolvem privacidade, direitos autorais, disseminação de falsas informações, manipulação da opinião pública, entre outros? Apesar desses assuntos não serem novidade, a capacidade de geração e disseminação em escala industrial a um custo relativamente baixo, é que os torna ainda mais relevantes nesse novo cenário.

O presente trabalho tem como objetivo discutir algumas implicações éticas e legais do uso do ChatGPT em diferentes cenários e em estudos de caso específicos, tentando identificar questões relevantes e eventuais soluções e posicionamentos para lidar com essas novas situações viabilizadas pelo atual estágio de desenvolvimento da IA. Esse tipo de reflexão é de extrema importância para que a sociedade busque utilizar essas novas tecnologias de forma responsável e que gere mais benefícios do que prejuízos no seu uso.

2. CHATGPT NA EDUCAÇÃO

O ChatGPT na educação pode facilitar a aprendizagem e a pesquisa, mas, se mal utilizado, pode impactar em questões importantes como plágio e a falta de preocupação com o desenvolvimento do pensamento analítico e crítico. Segundo CHEN (2019):

"A inteligência artificial na educação tem o potencial de transformar a forma como os estudantes aprendem, permitindo que os professores se concentrem em atividades de alto valor agregado, como a criação de currículos e o planejamento de aulas, enquanto a IA lida com tarefas mais repetitivas e administrativas."

Mais do que olhar para os estudantes como potenciais fraudadores, é preciso repensar a educação e seu papel, incluindo as formas de avaliação. Isso inclui ensinar boas práticas e técnicas de uso de ferramentas como o ChatGPT. Em vez de solicitar que o ChatGPT gere um artigo completo sobre determinado assunto, o aluno pode alimentar a ferramenta com fontes bibliográficas relevantes sobre o assunto, solicitar resumos e fazer questionamentos sobre o conteúdo para que vá desenvolvendo sua capacidade de análise crítica. Isso não exclui a necessidade de ler trechos das fontes bibliográficas e a pesquisa em outras fontes para confirmar as informações. Do ponto de vista das avaliações, o surgimento dessas ferramentas acaba, paradoxalmente, valorizando cada vez mais o ser humano e as interações humanas, pois para termos uma avaliação um pouco mais efetiva, enriquecedora e isonômica, cada vez mais será necessária a interação humana, para que o estudante seja capaz de demonstrar realmente o que absorveu. E se ele for capaz de demonstrar isso sem grandes aparatos tecnológicos, pouco importa se durante o processo fez ou não uso desses aparatos, pois os objetivos de desenvolver autonomia e capacidade de pensamento crítico foram atingidos.

Analisando do ponto de vista do custo, as interações humanas de qualidade na educação têm um custo relativamente alto e talvez não acessível a todos. Por outro lado, o custo de criar cursos e materiais didáticos autoinstrucionais e automatizados, com pouca interação humana, inclusive nas avaliações, deve acabar sendo reduzido pelo uso de ferramentas, como o ChatGPT, por parte dos criadores de conteúdo e professores. Nesses casos, cabe ao estudante e ao seu código de moral fazer um bom uso das ferramentas de IA, para que sua experiência resulte no aprimoramento de suas capacidades e não somente em um certificado, com valor questionável. De certa forma, é como se a avaliação do estudante ocorra, de fato, na utilização dos conhecimentos, seja na vida profissional, numa entrevista de emprego ou na sua rotina diária. O grande ganho desse tipo de educação é o acesso cada vez mais amplo ao conhecimento e, de certa forma, paradoxalmente, uma importância

menor dos certificados. As certificações profissionais, obtidas em ambiente controlado e com questões e situações práticas elaboradas por especialistas, podem preencher essa lacuna e acabarem sendo uma prova mais contundente de domínio de alguns tipos de conhecimento, o que já é um movimento em andamento, especialmente no âmbito das TICs. No final, tudo é uma questão de rever os objetivos e desmistificar o uso das novas tecnologias. Mal comparando, as calculadoras eletrônicas não substituíram os matemáticos e engenheiros e não foram abolidas das disciplinas. Talvez em uma ou outra disciplina, de acordo com o objetivo, o uso delas não seja encorajado, como para uma criança que aprende as primeiras operações aritméticas, mas quando apropriado, não há prejuízo, pelo contrário, acelera a realização de cálculos manuais em favor do aprendizado de novos conceitos e relações.

Durante as aulas síncronas da presente disciplina, um colega mencionou que “conversou” com o ChatGPT sobre o Mito da Caverna de Platão como analogia às bolhas informacionais e Fake News da atualidade e isso possibilitou que ele argumentasse sobre diversos pontos de vista durante a aula síncrona sem o auxílio, naquele exato momento, do ChatGPT, mostrando que a ferramenta possibilitou que ele refletisse sobre diversos aspectos de forma crítica, que é exatamente o que a educação objetiva. Muito diferente seria se ele tivesse apenas solicitado um artigo ao ChatGPT e sem ler, sem refletir sobre ele, fizesse a entrega. Nosso grupo frequentemente discute alguns conceitos e dúvidas e com muita naturalidade compartilhamos nosso processo de perguntas e respostas com ferramentas de IA, como o ChatGPT, para subsidiar nossas convicções e argumentos, tudo isso, sempre buscando construir uma visão crítica e reflexiva.

3. CHATGPT NO ATENDIMENTO AO CLIENTE

O ChatGPT também pode melhorar a eficiência do atendimento ao cliente, mas pode levar à desumanização das interações e à perda de empregos. De acordo com o relatório da McKinsey (2023), “Um em cada 16 trabalhadores poderá ter de mudar de profissão até 2030.” e “O crescimento do emprego estará mais concentrado em empregos de alta qualificação (por exemplo, nas áreas de saúde ou ciência, tecnologia, engenharia e matemática [STEM]), enquanto empregos de média e baixa qualificação (como serviços de alimentação, trabalho de produção ou funções de suporte de escritório) diminuirão”. Novamente vemos na história um momento disruptivo, com grande ganho de produtividade pelo uso de ferramentas tecnológicas e isso representa um problema social, pois talvez nem todos os cidadãos estejam preparados para

assumirem outras atividades laborais, que tendem a ficar cada vez mais complexas e especializadas. Isso remete à educação e a necessidade de sua reformulação e universalização do acesso. Como abordado anteriormente, a tendência é que haja uma redução no custo de produção de conteúdos educacionais, o que é importante para uma ampliação do acesso e também para que seja possível que a educação acompanhe o ritmo cada vez mais acelerado de desenvolvimento tecnológico.

Sobre a desumanização das interações, já existem iniciativas de IA para levar em consideração questões mais sutis como as emoções nas interações, mas à medida que seja possível resolver alguma demanda rapidamente, essa questão acaba sendo cultural e as necessidades de interação mais humana podem ser satisfeitas de outras formas. Provavelmente, cada vez menos pessoas trocariam as facilidades de resolver suas demandas bancárias por meio de aplicativos em favor de passar horas em uma fila pela simples interação social. O tempo antigamente gasto nas filas dos bancos, pode ser aproveitado passeando no parque e interagindo com outras pessoas com muito mais qualidade do que em uma fila em um ambiente fechado e com forte esquema de segurança.

Uma questão importante no atendimento automatizado que vem sendo discutida é a possibilidade de intervenção humana a qualquer momento, para seja possível a resolução dos problemas, já que as ferramentas não têm toda autonomia e capacidade para resolver todo tipo de situação. Muitas empresas têm apostado em solução automatizadas de atendimento, restringido totalmente o acesso a operadores humanos, mas não conseguem oferecer a solução dos problemas com essas soluções e isso deixa os usuários, que demandam alguma solução, reféns dos chatbots e suas limitações naturais ou impostas intencionalmente pelos seus proprietários. É preciso tomar cuidado para que isso não viole direitos fundamentais dos cidadãos.

4. CHATGPT NA CRIAÇÃO DE CONTEÚDO

Na criação de conteúdo, o ChatGPT pode auxiliar, mas também pode ser usado para gerar conteúdo enganoso ou tendencioso. Conforme apontado por Radford et al. (2019), "É importante que os modelos de linguagem sejam treinados com dados diversos e representativos para evitar a geração de conteúdo discriminatório ou ofensivo.". Nesse aspecto o uso do aprendizado por reforço humano tem grande importância, pois visa balizar em termos de objetivos e também em termos éticos as saídas dos modelos de IA para que não ofendam e discriminem pessoas e grupos. Esse tipo de ajuste é muito sensível ao momento e ao local, assim como pode ser bastante polêmico, assim como é a

ética. Outra questão ética e legal que surge é sobre a responsabilização sobre a geração de conteúdos que firam direitos fundamentais das pessoas e grupos. Quem é o responsável? Quem treinou o modelo? Quem conduziu o aprendizado por reforço humano? Quem escreveu o "prompt"? Quem divulgou? São questões que estarão cada vez mais presentes nos debates da sociedade e deverão ter alguma regulamentação mínima para balizar o uso dessas ferramentas.

Como apontado por Floridi (2020): "A inteligência artificial generativa, como o ChatGPT, pode ser usada para criar notícias falsas e outros tipos de conteúdo enganoso em escala industrial." Com a proximidade das eleições municipais esse ano, essa questão ganha maior relevância, pois potenciais efeitos da geração de conteúdo falso, como Deep Fakes e Fake News, podem ter um impacto sobre uma cidade, um estado ou até mesmo um país, como já se presenciou nas eleições norte-americanas em 2016, com o escândalo da assessoria Cambridge Analytica, posteriormente na eleição norte-americana seguinte e até mesmo nas eleições brasileiras.

5. CHATGPT E A PROTEÇÃO AOS DADOS PESSOAIS

O caso das eleições norte-americanas de 2016 e da assessoria Cambridge Analytica leva a outro questionamento ético e legal que diz respeito à propriedade dos dados pessoais gerados nas redes sociais e aplicativos, na privacidade deles e aos limites de uso desses dados por terceiros. Recentemente no Brasil saiu uma série de reportagens de Rossi (2023) sobre a comercialização dos dados de compras de clientes de uma grande rede de farmácias nacional que, para obter dados sensíveis, se utiliza de incentivos como descontos em medicamentos, que são questionáveis. Em nenhum momento fica claro para o cliente que esses descontos são condicionados ao uso e comercialização de seus dados pessoais e de consumo. Essa rede de farmácias possui até mesmo uma empresa própria para a coleta, transformação e comercialização desses dados, dado o valor comercial dessas informações. Com a utilização de modelos de IA que codificam de uma forma opaca as relações entre os dados que alimentam o treinamento desses modelos, fica até difícil estabelecer essa relação entre a saída e o dado que alimentou e originou essa saída, evidenciando que talvez seja necessário algum tipo de regulamentação sobre a guarda dos dados originais utilizados para o treinamento para seja possível auditar eventuais violações de privacidade e até mesmo de direitos autorais, como muito tem se discutido atualmente, onde autores e editoras de livros têm denunciado o uso não autorizado de suas obras para o treinamento desses modelos.

O uso de dados pessoais para treinar o ChatGPT traz reflexões importantes sobre privacidade, propriedade e consentimento de uso de dados. É essencial que os cidadãos tenham controle sobre seus próprios dados e que fique claro o tipo de uso que será feito daqueles dados que, eventualmente, haja consentimento de uso. Além de algum mecanismo que garanta que os dados serão utilizados exclusivamente para os fins acordados.

6. CHATGPT NA TOMADA DE DECISÕES

A tomada de decisões automatizada com o apoio do ChatGPT também é um tema que requer cuidado e atenção no que se refere à responsabilidade e transparência. À medida que a sociedade deposita muita confiança nessas ferramentas, elas se tornam grandes oráculos modernos, o que leva muitos a acatarem decisões como as melhores possíveis e como verdades absolutas, enquanto sabe-se que essas ferramentas não são perfeitas e estão suscetíveis a falhas e a perpetuar vieses já existentes e indesejáveis. Nesse aspecto, é importante buscar garantir que essas falhas e vieses sejam minimizados e monitorados para evitar prejuízo para pessoas e grupos. A intervenção humana deverá estar disponível a qualquer momento para evitar que os prejuízos sejam amplificados. Esse tipo de debate remete também à discussão sobre a manipulação na criação de conteúdo, nesse caso especificamente, pessoas com influência no projeto dessas ferramentas de IA, podem criar vieses intencionais para obterem algum tipo de benefício. Assim, é importante entender como essas ferramentas são projetadas e como funcionam para que não haja mistificações sobre elas e para seja sempre possível redirecionar as saídas à medida em que se comportam de forma antagônica à ética e a lei vigente naquele momento e local.

7. CONSIDERAÇÕES FINAIS

Para lidar com os desafios éticos do ChatGPT, é necessário adotar uma abordagem responsável em seu desenvolvimento e uso. Algumas soluções incluem:

- 1 Transparência:** Os usuários devem ser informados sobre como o modelo funciona e quais dados são utilizados para treiná-lo.
- 2 Responsabilidade:** Os desenvolvedores devem ser responsabilizados pelos resultados do modelo e tomar medidas para mitigar os riscos.

3 Diversidade e Inclusão: É importante garantir que o modelo seja treinado com dados representativos de diferentes grupos sociais para evitar vieses e discriminação.

4 Educação e Conscientização: É fundamental educar os usuários sobre os benefícios e riscos do ChatGPT para que possam tomar decisões informadas sobre seu uso.

O ChatGPT tem o potencial de revolucionar diversas áreas, mas seu uso também apresenta desafios éticos significativos. Ao adotar uma abordagem responsável, podemos garantir que essa tecnologia seja utilizada de forma ética e benéfica para a sociedade. É crucial que pesquisadores, desenvolvedores, legisladores e a sociedade em geral trabalhem juntos para encontrar soluções que promovam o uso responsável e ético do ChatGPT.

■ REFERÊNCIAS

- CHEN, B.; LIU, H.; ZHANG, J. Integrating artificial intelligence into educational technology research and development. New Jersey: Educational Technology Research and Development, 2019.
- Floridi, L. (2020). Inteligência artificial: Uma breve introdução. Rio de Janeiro: Zahar.
- MCKINSEY & COMPANY. What is the future of work? McKinsey Explainers. Disponível em: <https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-the-future-of-work>. Acesso em: 30 jun. 2024.
- O'Neil, C. (2016). Weapons of math destruction: How big data increases inequality and threatens democracy. New York: Crown.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. OpenAI Blog.
- Rossi, Amanda. Vitamina usada para tentar engravidar pode direcionar até anúncio de carro. UOL, São Paulo, 1 set. 2023. Disponível em: <https://noticias.uol.com.br/saude/ultimas-noticias/redacao/2023/09/01/remedio-para-engravidar-pode-direcionar-ate-propaganda-de-carro.htm?cmpid=copiaecola>. Acesso em: 30 jun. 2024.
- Zuboff, S. (2019). The age of surveillance capitalism: The fight for a human future at the new frontier of power. New York: PublicAffairs.

APÊNDICE 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

Qual o objetivo do estudo descrito pelo artigo?	Qual o problema/oportuni- dade/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 principal do artigo é estudar a diferença (gap) entre as práticas de "RE" (engenharia de requisitos) na questão do desenvolvimento de Inteligências Artificiais centradas no ser humano. Mostra principalmente a diferença entre as práticas aplicadas pela indústria comparado com as práticas na parte de pesquisa. Essa lacuna existente apenas	Conforme comentado no objetivo, a causa principal que levou à elaboração desse estudo foi a necessidade de priorizar ganhos de desempenho e performance ao longo dos anos nas I.A 's, levando sempre a altíssimos patamares de poder e processamento computacional, para termos modelos mais robustos, com maior quantidade de datasets (dados) para	Os autores realizaram uma pesquisa com profissionais que trabalham com desenvolvimento de sistemas de I.A. O principal objetivo de realizar a pesquisa foi compreender se e como esses profissionais aplicam práticas centradas no ser humano na fase de engenharia de requisitos dos projetos. A pesquisa também ajudou a identificar e	Após análise quantitativa das respostas realizadas pelos participantes da pesquisa, somando também às análises e estudos prévios realizados pelos autores na parte de entender como a indústria se comporta na fase de levantamento de requisitos, ficou confirmado a necessidade de considerar o uso e aplicação dos aspectos humanos nos

comprova como é preocupante o uso desenfreado tanto de modelos de inteligência artificial como a pesquisa, criação e desenvolvimento de novos modelos apenas focados no cerne técnico dessa tecnologia, pois nos últimos anos a preocupação sempre foi em criar modelos mais performáticos, rápidos e otimizados. Contudo nesse mesmo período não foi dado tanta ênfase para um dos assuntos mais importantes quando falamos de prováveis entidades autônomas que estão surgindo graças a evolução constante de técnicas de machine learning, i.a generativa dentre outras que, como demonstra o artigo, mostra que falta um foco maior nessas tecnologias para serem centradas no ser humano. Por ser uma área	treinamentos e processamento de respostas (para I.A. 's generativas por exemplo) mais rápidos, coerentes e assertivos. Devido a essa falta de foco nos aspectos centrados em humanos durante o desenvolvimento de I.A, surgiram modelos e sistemas com problemas de vieses, pouca capacidade explicativa de como chegou a determinados resultados e conclusões, não atender as necessidades reais do usuário e falhas grandes no quesito de ignorar questões humanas causando consequências negativas, por exemplo, chatbots racistas. Devido a essa lacuna entre os objetivos técnicos e a questão humana, o estudo destaca a oportunidade de como práticas de Engenharia de Requisitos podem evoluir	entender quais ferramentas e métodos são utilizados por cada um. A estruturação da pesquisa foi realizada elaborando um questionário a partir dos resultados de uma revisão sistemática da literatura (RSL) sobre RE4AI e de diretrizes da indústria (através de documentos e metodologias criadas e disponibilizadas por grandes players do mercado: Google, Microsoft e Apple). Através das diretrizes industriais, foram categorizados em seis áreas-chaves os aspectos principais de I.A. centrada em humanos, sendo elas: - Necessidade do Usuário - Necessidade do Modelo - Necessidade de Dados - Feedback e Controle de Usuário - Explicabilidade e Confiança	projetos de I.A. Além disso ficou comprovado um grande nível de desafio e limitações dentre os profissionais da área, muitas delas causadas pela natureza complexa e incerta da I.A. Conforme já citado também, a lacuna entre a prática aplicada no mercado comparado com as práticas recomendadas na literatura são bastante divergentes, onde profissionais, por exemplo, utilizam plataformas e ferramentas genéricas para estruturar e criar os projetos de I.A., e não ferramentas específicas recomendadas pela literatura. Outro aspecto importante resultante da pesquisa foi a grande maioria dos participantes concordar e entender a importância da aplicar as seis áreas-chaves categorizadas na fase de pesquisa. Mesmo
--	---	--	--

relativamente nova e, comparado com qualquer outra tecnologia ou inovação lançada no mercado sem uma regulamentação ou padrão, cada interessado acabou seguindo suas próprias crenças e normas de como realizar a pesquisa e desenvolvimento, causando esses problemas de falta de foco no ser humano. Grandes empresas do mercado, referências na área, já estão criando documentos e manuais de melhores práticas a serem seguidas, dessa forma a sociedade possui, mesmo ainda não sendo tão objetivo e claro, um caminho a ser seguido para criarmos cada vez mais modelos de inteligência artificial com preocupações reais de ética, empatia, justiça e imparcialidade, mesmo o foco atualmente dessas grandes empresas	ajudar a melhorar esses aspectos, porém aborda também a dificuldade e até mesmo a falta de ferramentas e metodologias adequadas para definir, capturar, analisar e gerenciar os requisitos necessários para a elaboração de modelos de I.A.	- Erros e Falha. Após essa categorização, foram mapeados as relações dessa lista com o estudo acadêmico RSL, buscando encontrar as convergências e divergências entre a pesquisa e as práticas recomendada pela indústria. A principal diferença, como já detalhado anteriormente, é a lacuna entre a preocupação técnica da indústria e a literatura acadêmica RE4AI. A pesquisa foi respondida, em sua totalidade por 29 profissionais, onde obtiverem o acesso a ela através de um formulário online enviado para pessoas próximas dos autores e, em seguida, solicitado para compartilharem com outras pessoas da área. Os principais tópicos abordados foram referentes aos dados demográficos dos participantes, as ferramentas e métodos utilizados por	assim, muitos admitiram a falta de ênfase e foco em áreas como a coleta de feedback e o controle de erros. Também ficou definido como a falta de ferramentas específicas para projetos de I.A. podem acarretar na falta de incorporação dos princípios centrados no ser humano. Uma plataforma ou metodologia que auxiliasse na aplicação, revisão e garantia que, por exemplo, as seis áreas-chaves fossem corretamente aplicadas já garantiria uma consistência, assertividade e até uma segurança maior nos próximos lançamentos de projetos de I.A. Além disso novas linguagens de modelagem de dados, mais específicas para a complexidade dessa área, também ajudaria em toda essa definição de requisitos.
--	--	---	--

serem mais voltados à parte técnica do desenvolvimento. Dessa forma o artigo traz toda uma contextualização do motivo da importância desse assunto, além de, através de pesquisas, conseguir determinar as lacunas que existem entre as práticas aplicadas no mercado de trabalho comparada com o praticado na área acadêmica/pesquisa, junto também com uma forma de determinar quais diretrizes de IA devemos seguir para o levantamento de requisitos de um projeto		eles, os desafios e limitações enfrentados durante a fase de levantamento de requisitos para os projetos de I.A. e, por último, a importância de diretrizes centradas no ser humano.	
---	--	---	--

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/wine-quality-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)

1. importação de bibliotecas

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
```

2. importação e separação da base de dados

```
from tensorflow.keras.datasets import fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", y_test.shape)
print("y_test.shape: ", y_test.shape)
```

3. Pré-processamento

```
# Normalizar os valores dos pixels para o intervalo [0, 1]
x_train, x_test = x_train/255.0, x_test/255.0

print(x_train.shape, x_test.shape) # Mantém o formato original
```

4. Criação do Modelo

```
i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
```

```
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)
```

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

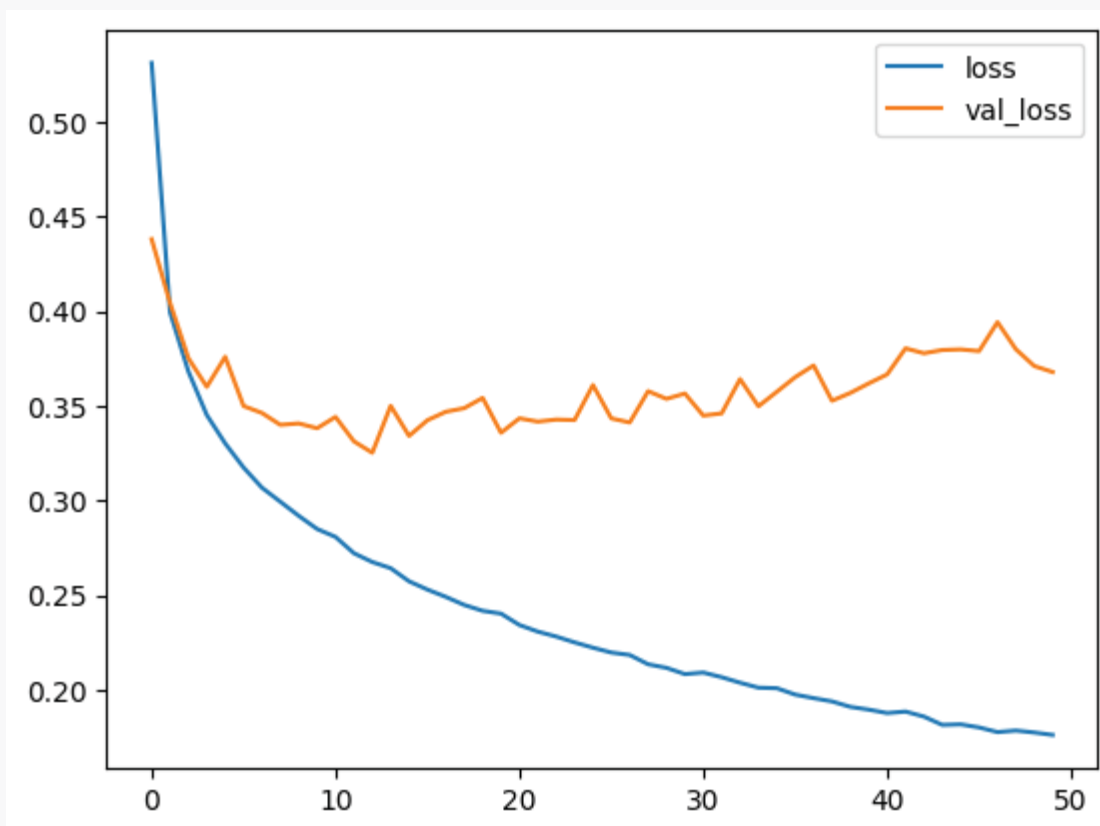
5. Compilação e Treinamento do Modelo

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

```
r = model.fit(x_train,
             y_train,
             validation_data=(x_test, y_test),
             epochs=50)
```

6. Avaliação do Modelo

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



Fonte: O Autor (2025)

A função de perda no conjunto de treinamento (*loss*) está diminuindo de forma consistente, indicando que o modelo está aprendendo e ajustando seus pesos corretamente durante o treinamento.

Porem, após algumas épocas (cerca de 10), a perda no conjunto de validação (`val_loss`) começa a se estabilizar ou até mesmo a aumentar, enquanto a perda no conjunto de treinamento continua diminuindo.

Isso sugere *overfitting*, ou seja, o modelo está memorizando os dados de treinamento e perdendo a capacidade de generalizar para novos dados.

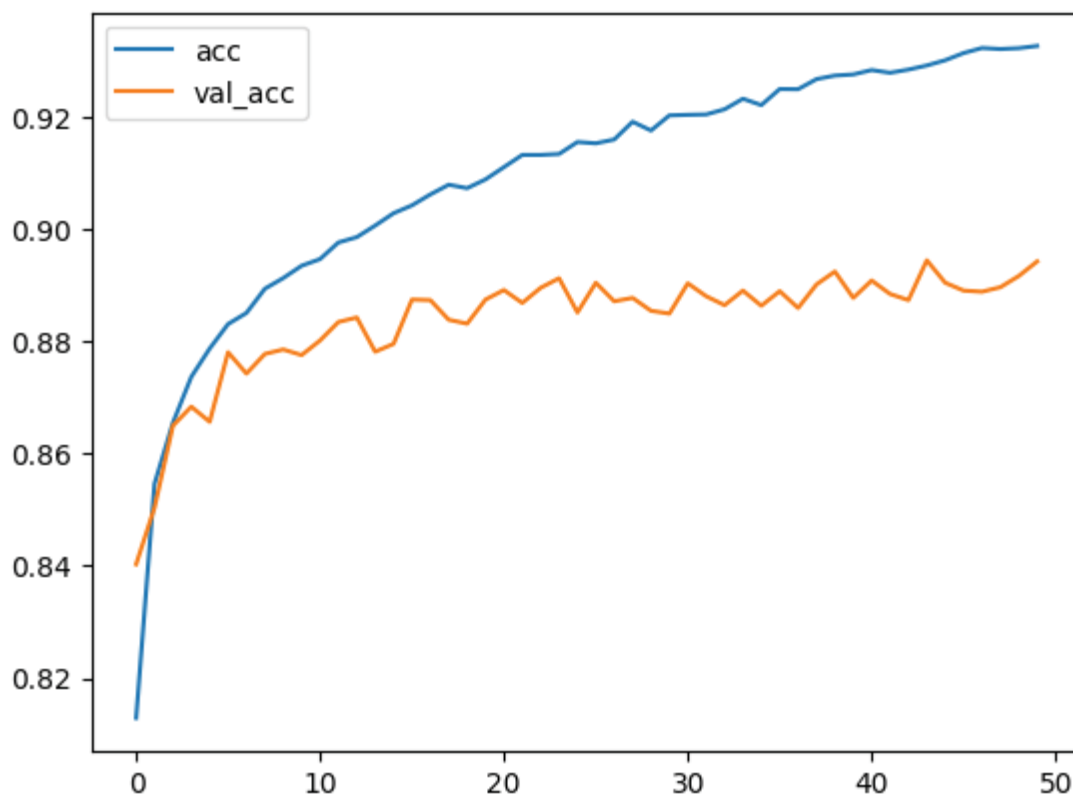
Para tentar contornar a situação, podemos fazer as seguintes técnicas:

****1)**** melhorar o desempenho do modelo interrompendo o treinamento antes que o `val_loss` comece a aumentar. Isso pode ser feito utilizando uma técnica chamada *Early Stopping* no Keras.

****2)**** regularização para evitar *overfitting*, adicionando dropout no modelo ou regularização L2.

****3)**** Ajustando os hiperparâmetros, para que reduza a taxa de aprendizado do otimizador adam, o que pode ajudar a encontrar um ponto de convergência melhor.

```
# Plotar a acurácia
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
```



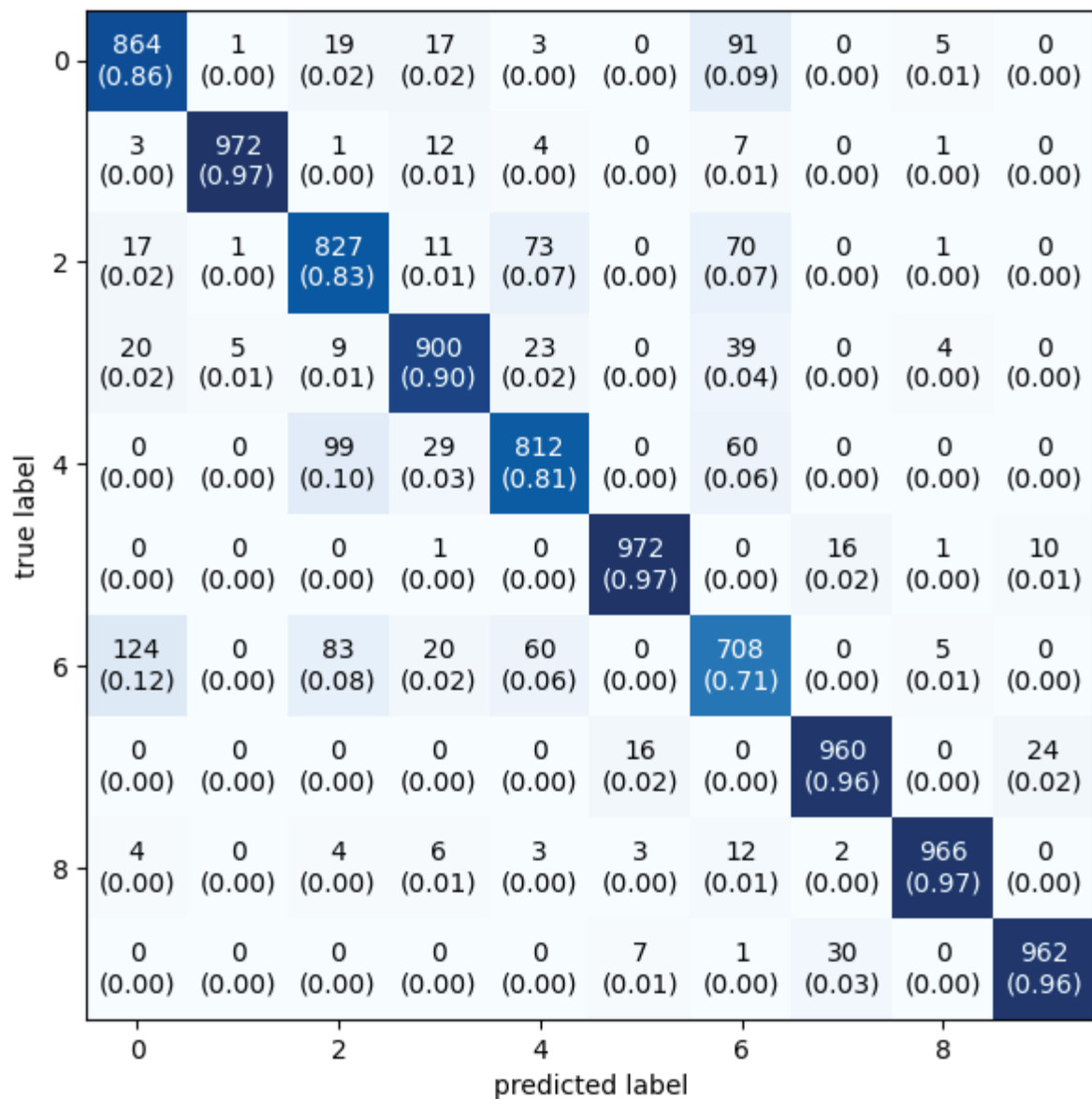
O mesmo comentário se aplica quando olhamos para a acurácia do modelo após as 10 épocas.

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

7. Predições

```
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)
```

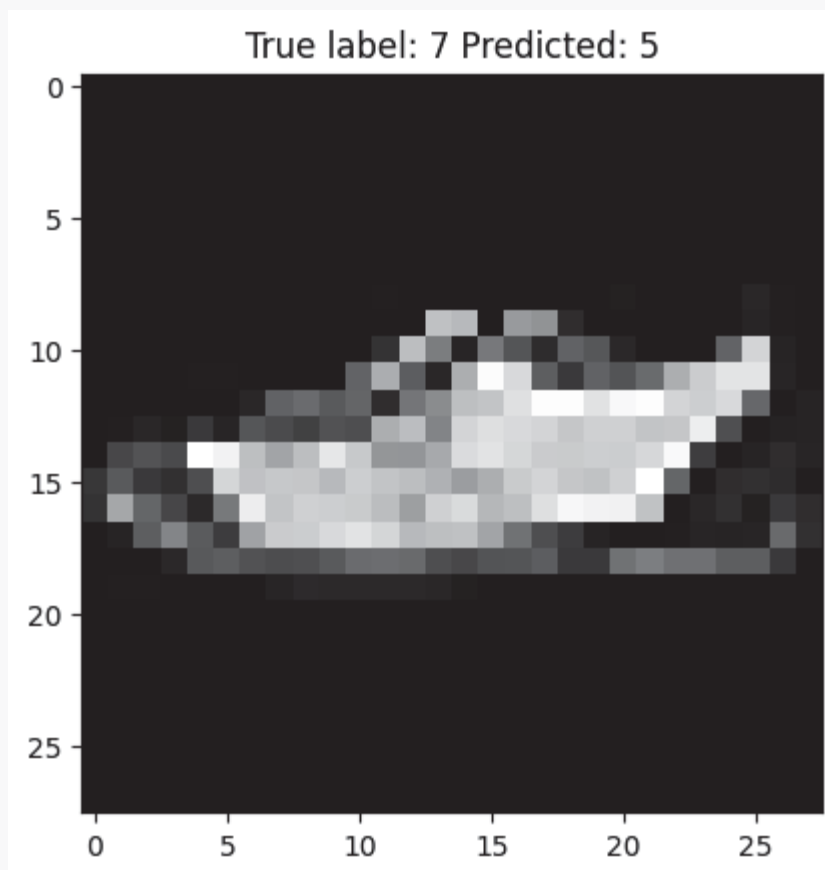


8. Mostrando algumas predições erradas

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

i = np.random.choice(misclassified)

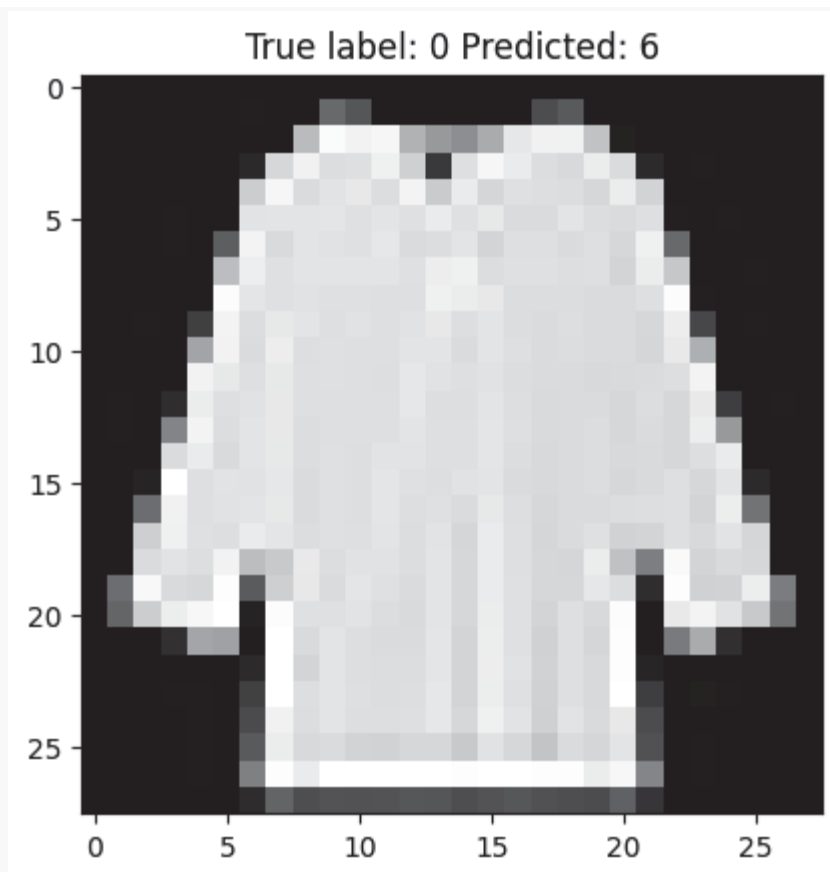
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))
```



Fonte: O Autor (2025)

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

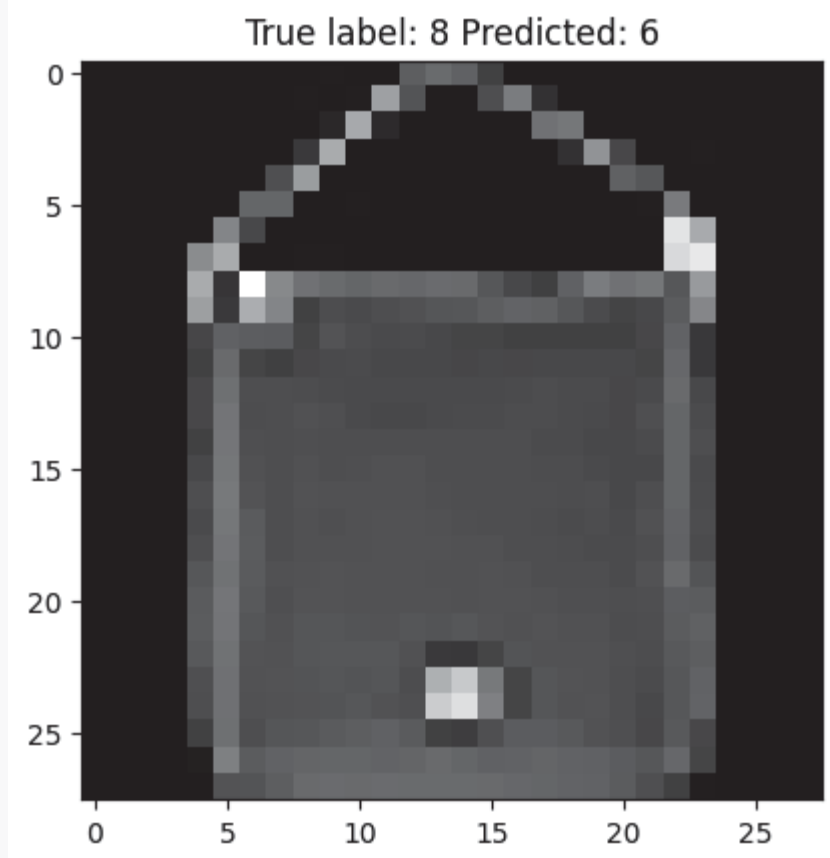
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))
```



Fonte: O Autor (2025)

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

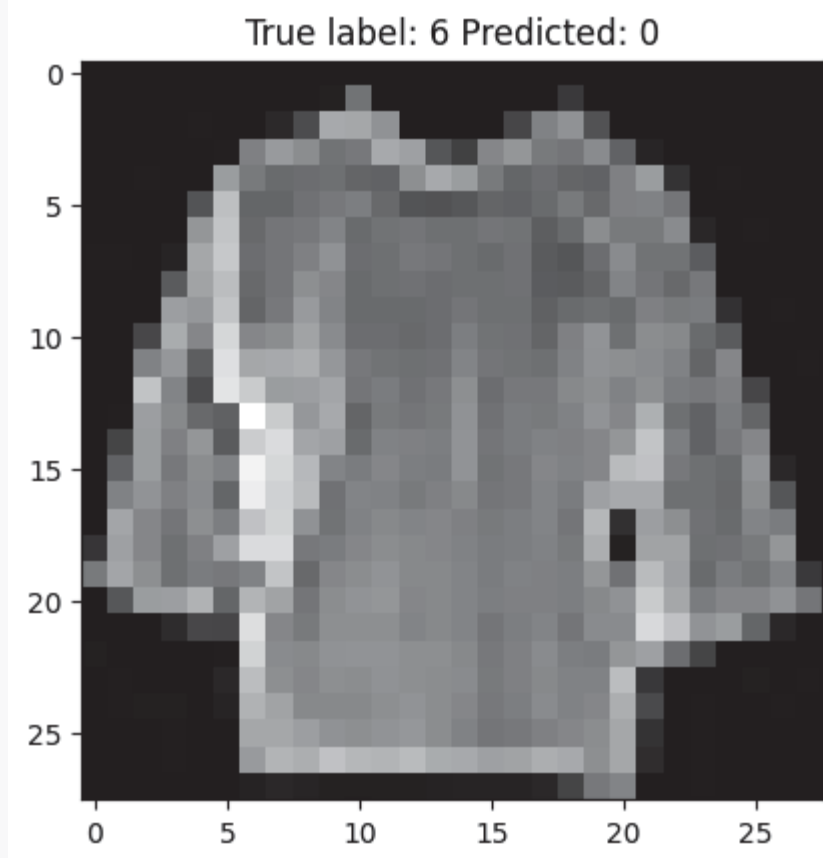
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))
```



Fonte: O Autor (2025)

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

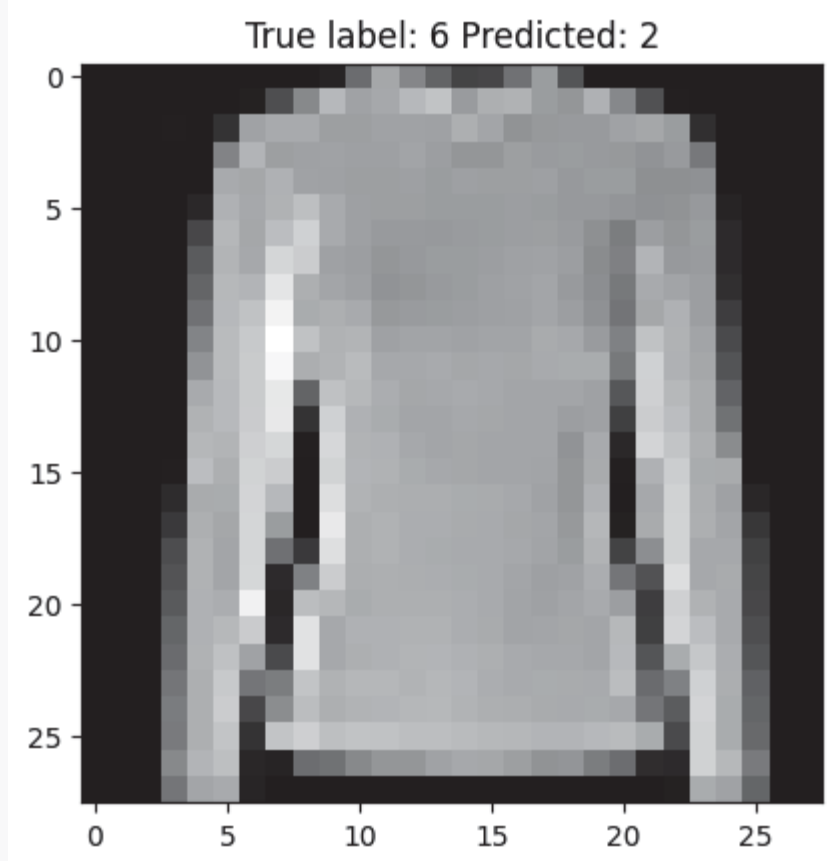
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))
```



Fonte: O Autor (2025)

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

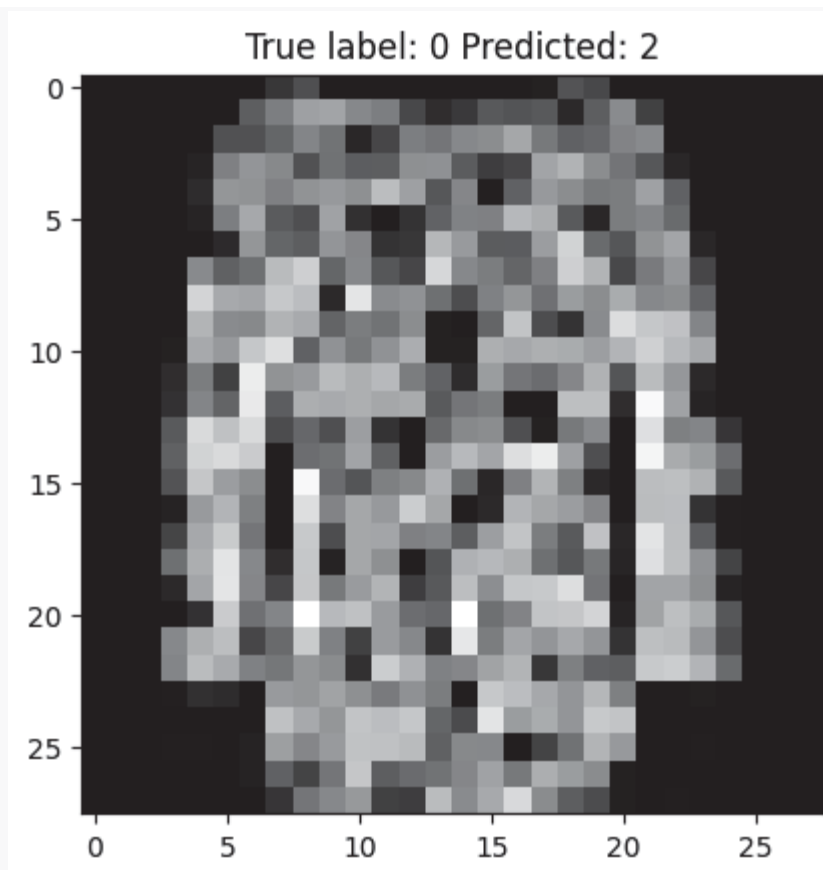
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))
```



Fonte: O Autor (2025)

```
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]))
```



Fonte: O Autor (2025)

Analisar exemplos mal classificados é crucial para entender onde o modelo está falhando e como ele pode ser melhorado. Pode indicar a necessidade de mais dados de treinamento, ajustes nos hiperparâmetros ou melhorias na arquitetura da rede.

Este gráfico é uma ferramenta útil para diagnosticar e melhorar o desempenho do modelo de classificação. Ele destaca as áreas onde o modelo precisa de ajustes e ajuda a direcionar os esforços de melhoria.

2 Regressão (RNA)

```
url
="https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/wine
quality-red.csv"
data = pd.read_csv(url, delimiter=';')
# renomeado as colunas
data.columns = [
'acidez_fixa', # fixed acidity
'acidez_volatil', # volatile acidity
'acido_citrico', # citric acid
'acucar_residual', # residual sugar
'cloretos', # chlorides
'dioxido_de_enxofre_livre', # free sulfur dioxide
'dioxido_de_enxofre_total', # total sulfur dioxide
```

```

'densidade', # density
'pH', # pH
'sulfatos', # sulphates
'alcool', # alcohol
'score_qualidade_vinho' # quality
]

# Se parando em variavel resposta e explicativa
X = data.drop('score_qualidade_vinho',axis = 1)
Y = data['score_qualidade_vinho'].astype(np.float32)
# separando a base em treino e teste
x_train, x_test, y_train, y_test = train_test_split(X, Y,
                                                    test_size=0.3)

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

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

5. Compilação e treinamento do modelo

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

def r2(y_true, y_pred):
    media = backend.mean(y_true)
    num    = backend.sum (backend.square(y_true - y_pred))
    den    = backend.sum (backend.square(y_true - media))
    return (1.0 - num/den)

# Compilação
optimizer=tf.keras.optimizers.Adam(learning_rate=0.05)
# optimizer=tf.keras.optimizers.SGD(learning_rate=0.2, momentum=0.5)
# optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
              loss="mse",
              metrics=[rmse, r2])

# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
                    monitor='val_loss',
                    patience=20,
                    restore_best_weights=True)
r = model.fit(x_train, y_train,
              epochs=1500,
              validation_data=(x_test, y_test),
              callbacks=[early_stop])

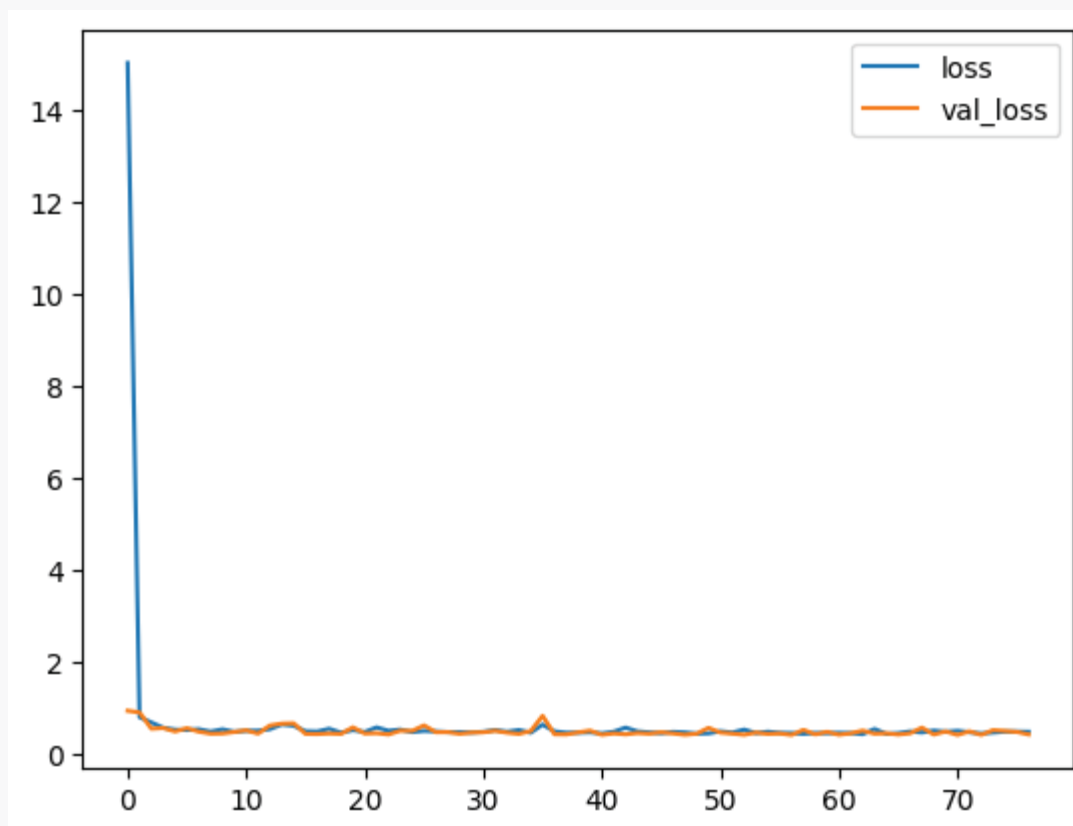
```

```
Epoch 1/1500 35/35 ----- 1s 7ms/step - loss:
35.8539 - r2: -61.3804 - rmse: 3.7833 - val_loss: 0.9386 - val_r2: -0.5609
- val_rmse: 0.7534 Epoch 2/1500 35/35 ----- 0s
2ms/step - loss: 0.8366 - r2: -0.4094 - rmse: 0.7183 - val_loss: 0.9023 -
val_r2: -0.5099 - val_rmse: 0.7278 Epoch 3/1500 35/35
----- 0s 2ms/step - loss: 0.7323 - r2: -0.1903 -
rmse: 0.6677 - val_loss: 0.5477 - val_r2: 0.1039 - val_rmse: 0.5731 Epoch
4/1500 35/35 ----- 0s 2ms/step - loss: 0.5709 -
r2: 0.0972 - rmse: 0.5824 - val_loss: 0.5635 - val_r2: 0.0716 - val_rmse:
0.5754 Epoch 5/1500
Epoch 76/1500 35/35 ----- 0s 2ms/step - loss:
0.4849 - r2: 0.1756 - rmse: 0.5546 - val_loss: 0.4786 - val_r2: 0.1894 -
val_rmse: 0.5599 Epoch 77/1500 35/35 ----- 0s
3ms/step - loss: 0.4718 - r2: 0.2675 - rmse: 0.5414 - val_loss: 0.4227 -
val_r2: 0.2979 - val_rmse: 0.4978
```

6. Avaliação do modelo

```
# função da perda
```

```
plt.plot( r.history["loss"], label="loss" )
plt.plot( r.history["val_loss"], label="val_loss" )
plt.legend()
```



Fonte: O Autor (2025)

Este gráfico é crucial para diagnosticar o comportamento do modelo durante o treinamento e a validação. Ele permite ajustes no modelo ou nos hiperparâmetros, se necessário, para melhorar o desempenho.

O grafico de perda do modelo se ajustei muito rapido aos dados, tanto de validação quanto de treno, não tendo tanta necessidade de chegar ate as 70 epocas.

##Observações Específicas

####Início do Treinamento:

A perda é alta, mas rapidamente diminui, indicando que o modelo está aprendendo.

###Estabilização:

Após algumas épocas, tanto a perda quanto a perda de validação se estabilizam em valores baixos, sugerindo que o modelo está convergindo.

###Proximidade das Linhas:

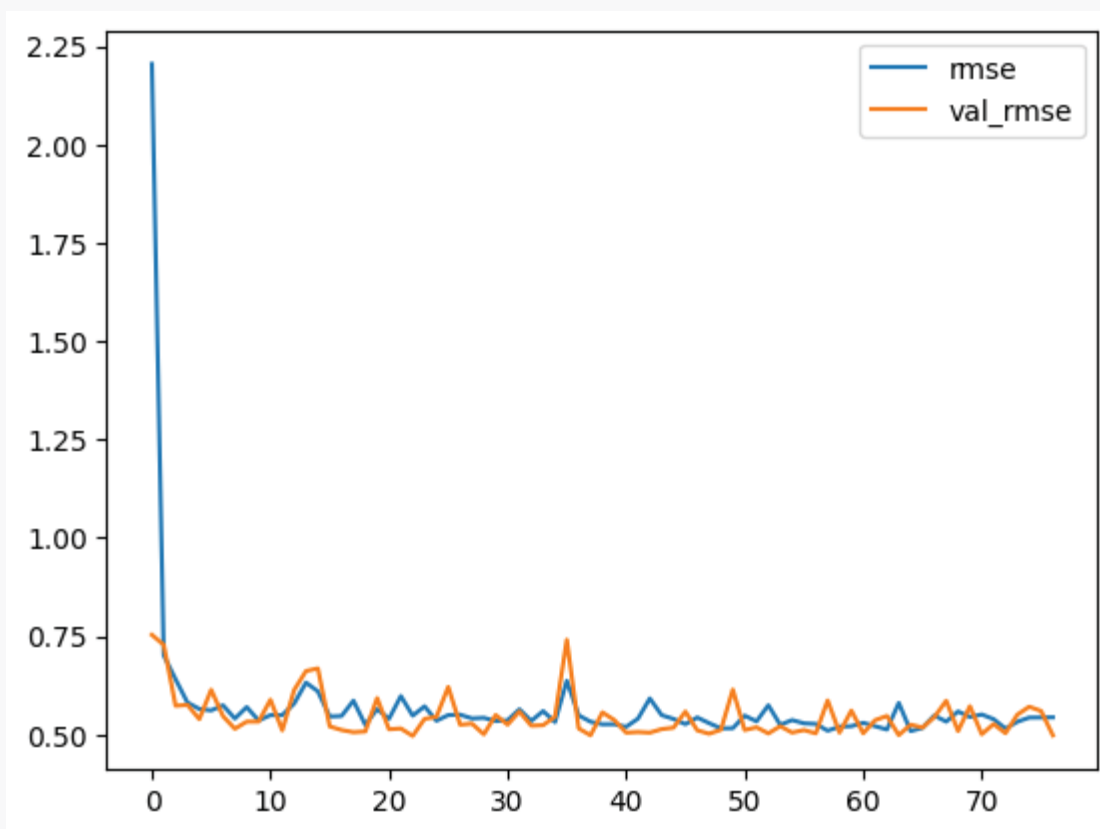
A proximidade entre as linhas de perda e perda de validação sugere que o modelo não está sofrendo de overfitting significativo, pois a perda de validação não está aumentando enquanto a perda de treinamento continua a diminuir.

Valor de RMSE

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

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

```
plt.legend()
```

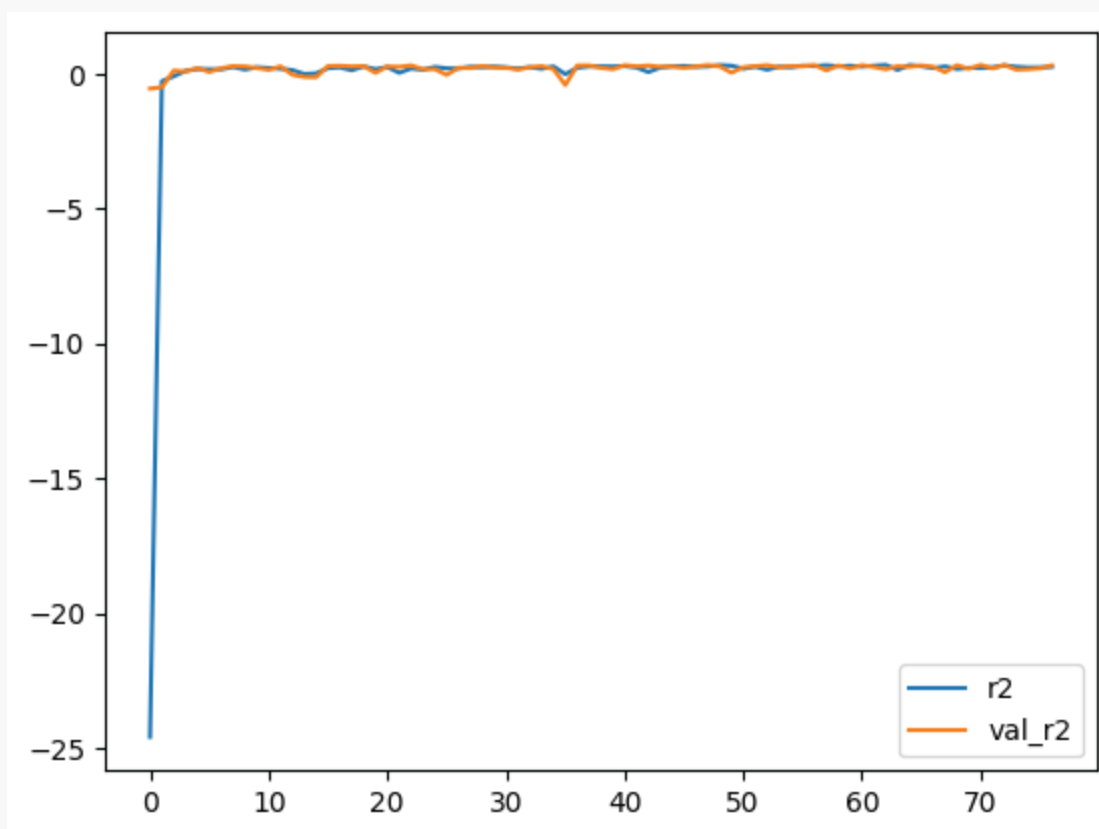


Fonte: O Autor(2025)

Valor do R2

```
plt.plot( r.history["r2"], label="r2" )
```

```
plt.plot( r.history["val_r2"], label="val_r2" )
plt.legend()
```



Fonte: O Autor(2025)

7. Predições

```
# Predição
y_pred = model.predict(x_test).flatten()

# Cálculo das métricas de acurácia: mse, r2 e rmse
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred)
```

1) MSE e RMSE: O modelo tem um erro de previsão razoável, mas o erro quadrático (RMSE) pode ser alto, dependendo do contexto e das unidades dos dados.

2) R^2 : Com 35.47%, o modelo está explicando uma parte moderada da variabilidade dos dados, mas ainda há bastante variabilidade que o modelo não está capturando. Isso pode indicar que o modelo pode ser aprimorado, ou que outras variáveis explicativas devem ser consideradas.

Proximos passos para tentar melhorar as métricas:

- Ajuste do modelo: mudar a arquitetura da rede neural ou adicionar novas variáveis para melhorar a explicação da variabilidade dos dados.

- *Análise de Residuals: Verifique os resíduos (diferença entre valores previstos e reais) para avaliar se há padrões que o modelo não está capturando.*

- *Ajustes de Hiperparâmetros: Teste diferentes hiperparâmetros, como número de camadas, neurônios, taxas de aprendizado, etc.*

fweffgew

3 Sistemas de Recomendação

```
df = pd.read_excel('Base_livros.xlsx')
df.head(3)

# removendo as linhas em que nao tivemos nota
print(df.shape)
df = df.dropna(subset=['Notas', 'Titulo'])
print(df.shape)

3. Conversão de userId e movieId para categoria

df['ID_usuario'].isna().sum(), df['Titulo'].isna().sum()

# userId e movieId não estão no formato certo para usar
# Embeddings > devem ser categóricos

df.ID_usuario = pd.Categorical(df.ID_usuario)
df['new_user_id'] = df.ID_usuario.cat.codes

df.Titulo = pd.Categorical(df.Titulo)
df['new_book_id'] = df.Titulo.cat.codes

# Dimensões
N = len(set(df.new_user_id))
M = len(set(df.new_book_id))

# dimensão do embedding (tentar outros)
K = 10

4. Criar o modelo

# Definindo o modelo com Dropout
# usuário
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída: (num_samples, 1, K)
u_emb = Flatten()(u_emb) # saída: (num_samples, K)

# usuário
u = Input(shape=(1,))
```

```

u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

# filme
m = Input(shape=(1,))
m_emb = Embedding(M, K)(m) # saída : num_samples, 1, K
m_emb = Flatten()(m_emb)   # saída : num_samples, K

x = Concatenate()([u_emb, m_emb])

# Camadas densas com Dropout
x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)

5. Compilação do modelo

model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)

6. Separação dos dados e pré-processamento

user_ids, book_ids, ratings = shuffle(df.new_user_id, df.new_book_id,
df.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

train_user = user_ids[:Ntrain]
train_book = book_ids[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_book = book_ids[Ntrain:]
test_ratings = ratings[Ntrain:]

# centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating

import numpy as np

# Verifica NaNs ou Inf nos dados
print(np.any(np.isnan(train_user)), np.any(np.isnan(train_book)),
np.any(np.isnan(train_ratings)))
print(np.any(np.isinf(train_user)), np.any(np.isinf(train_book)),
np.any(np.isinf(train_ratings)))

7. Treinamento do modelo

```

```

epochs = 50
r = model.fit(
    x=[train_user, train_book],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso
    validation_data=([test_user, test_book], test_ratings)
)

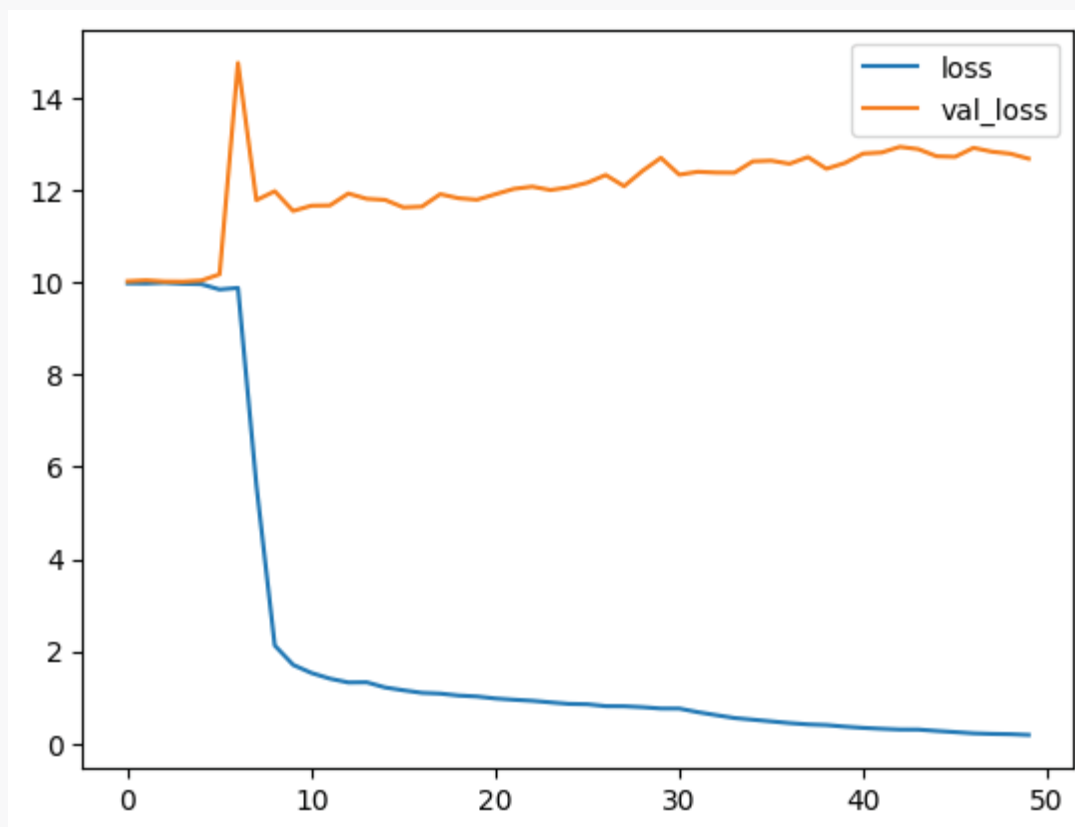
```

8. Plotar a função de perda

```

plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

```



Fonte: O Autor (2025)

Para o conjunto de treinamento a função perda diminui rapidamente nas primeiras épocas, o que indica que o modelo está aprendendo a ajustar seus pesos para minimizar o erro.

Por outro lado nos dados de treinamento diminui no início, mas estabiliza e começa a aumentar levemente ao longo das épocas levando indícios de possível overfitting, ou seja, o modelo está se ajustando bem aos dados de treinamento, mas está perdendo sua capacidade de generalização para novos dados (os dados de validação).

Este gráfico é crucial para diagnosticar o comportamento do modelo durante o treinamento e a validação. Ele permite ajustes no modelo ou nos hiperparâmetros, se necessário, para melhorar o desempenho.

9. Recomendações para o usuário 276726

```
# Gerar o array com o usuário único
# repete a quantidade de filmes
input_usuario = np.repeat(a=276726.0, repeats=M)
film = np.array(list(set(book_ids)))

preds = model.predict( [input_usuario, film] )

# descentraliza as predições
rat = preds.flatten() + avg_rating

# índice da maior nota
idx = np.argmax(rat)

print("Recomendação: Filme - ", film[idx], " / ", rat[idx] , "")

df[df['new_book_id'] == 7801 ].Titulo.unique()
```

O modelo recomenda o filme "Rage" para o usuário com ID 276726, baseado nas predições do modelo e ajustado pela média das avaliações.

4 Deepdream

2. importacao da imagem

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

# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
    name = url.split('/')[-1]
    image_path = tf.keras.utils.get_file(name, origin=url)
    img = PIL.Image.open(image_path)
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)

# Normalização da imagem
def deprocess(img):
    img = 255*(img + 1.0)/2.0
    return tf.cast(img, tf.uint8)

# Mostra a imagem
```

```
def show(img):
    display.display(PIL.Image.fromarray(np.array(img)))
```



Fonte: O Autor(2025)

```
# 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">Vo
n.grzanka</a>'))
```

3. Preparar o modelo de extração de recursos

```
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)
```

4. Cálculo da perda (*loss*)

```
def calc_loss(img, model):
    # Passe a imagem pelo modelo para recuperar as ativações.
    # Converte a imagem em um batch de tamanho 1.
    img_batch = tf.expand_dims(img, axis=0)
    layer_activations = model(img_batch)
    if len(layer_activations) == 1:
        layer_activations = [layer_activations]
```

```

losses = []
for act in layer_activations:
    loss = tf.math.reduce_mean(act)
    losses.append(loss)

return tf.reduce_sum(losses)

```

5. Subida de gradiente (*Gradient ascent*)

```

class DeepDream(tf.Module):
    def __init__(self, model):
        self.model = model

    @tf.function(
        input_signature=(
            tf.TensorSpec(shape=[None, None, 3], dtype=tf.float32),
            tf.TensorSpec(shape=[], dtype=tf.int32),
            tf.TensorSpec(shape=[], dtype=tf.float32),)
    )
    def __call__(self, img, steps, step_size):
        print("Tracing")
        loss = tf.constant(0.0)

        for n in tf.range(steps):
            with tf.GradientTape() as tape:
                # Gradientes relativos a img
                tape.watch(img)
                loss = calc_loss(img, self.model)

            # Calculo do gradiente da perda em relação aos pixels da imagem de
            entrada.
            gradients = tape.gradient(loss, img)

            # Normalizacao dos gradintes
            gradients /= tf.math.reduce_std(gradients) + 1e-8

            # Na subida gradiente, a "perda" é maximizada.
            # Você pode atualizar a imagem adicionando diretamente os gradientes
            (porque eles têm o mesmo formato!)
            img = img + gradients*step_size
            img = tf.clip_by_value(img, -1, 1)

        return loss, img

```

```

deepdream = DeepDream(dream_model)

```

6. Circuito principal (*Main Loop*)

```

def run_deep_dream_simple(img, steps=100, step_size=0.01):

    img = tf.keras.applications.inception_v3.preprocess_input(img)
    img = tf.convert_to_tensor(img)
    step_size = tf.convert_to_tensor(step_size)

```

```

steps_remaining = steps
step = 0
while steps_remaining:
    if steps_remaining>100:
        run_steps = tf.constant(100)
    else:
        run_steps = tf.constant(steps_remaining)
    steps_remaining -= run_steps
    step += run_steps

    loss, img = deepdream(img, run_steps, tf.constant(step_size))

    display.clear_output(wait=True)
    show(deprocess(img))
    print ("Step {}, loss {}".format(step, loss))

result = deprocess(img)
display.clear_output(wait=True)
show(result)

return result

dream_img = run_deep_dream_simple(img=original_img,
                                  steps=100, step_size=0.01)

```



Fonte: O Autor(2025)

O resultado é uma imagem modificada que realça os padrões vistos pela rede neural, criando um efeito visual diferente da imagem original e interessante, porém com baixa granularidade.

7. Levando o modelo até um oitava

```

import time
start = time.time()

OCTAVE_SCALE = 1.30

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[: -1]
float_base_shape = tf.cast(base_shape, tf.float32)

for n in range(-2, 3):
    new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

    img = tf.image.resize(img, new_shape).numpy()

    img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start

```



Fonte: O Autor(2025)

O resultado é uma imagem modificada, mais próxima do objetivo de criar uma imagem que remete ao 'sonho', criando um efeito visual diferente da imagem original e surreal, com aumento da granularidade.

Ao comparar as duas imagens, percebemos que a primeira imagem gerada é mais semelhante à imagem original, com pouca granularidade. Ao elevar o modelo à oitava, temos uma imagem gerada com maior granularidade e mais surreal, remetendo melhor a imagens oníricas.

APÊNDICE 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

Desmatamento na Floresta Amazônica

Introdução

A Amazônia possui um papel crucial e importantíssimo no equilíbrio do ecossistema global natural, pois abriga a maior floresta tropical do mundo, consequentemente uma enorme biodiversidade (fauna e flora) e inúmeros povos indígenas. Entretanto, nessas últimas décadas a região vem sofrendo com o aumento do desmatamento. Estima-se que cerca de 17% da floresta amazônica já foi totalmente destruída, com outros 17% em estado degradado, aproximando-se de um limite perigoso.

Cientistas alertam que se perdermos entre 20% a 25% da área original da floresta poderia levar ao chamado “ponto de não retorno”, já que essas grandes áreas desmatadas levariam a liberação de toneladas de carbono agr avando não só o aquecimento global mas também a transformação dessas partes de floresta em bioma savana. Diante dessa situação exposta, realizarei uma análise das tendências de desmatamento nos últimos 30 anos, bem como impactos diretos e indiretos no clima, fauna, flora e populações locais.

É de extrema importância compreender esse histórico para fundamentar não só políticas públicas, mas também o despertar da consciência da população, evitando assim danos irreversíveis para o planeta terra.

Dados e Metodologia

Os dados reunidos para realizar essa análise foram coletados de fontes confiáveis, sendo a principal referência o Projeto PRODES, o qual fornece anualmente informações precisas sobre o andamento do desmatamento na Amazônia. Esse projeto possui registrado dados e taxas desde o ano de 1988, permitindo assim uma análise detalhada do percentual de crescimento e/ou decréscimo ao longo dos anos.

A metodologia abrangeu desde a análise e padronização/normalização dos dados extraídos, até a criação de visualizações de dados mais precisas, intuitivas e, não menos importante, visualizações mais bonitas e chamativas.

Foi utilizada a ferramenta datawrapper (<https://datawrapper.de>) para elaboração dos gráficos aqui apresentados. Cada gráfico utilizado foi definido baseado nas variáveis e valores presentes para a melhor distribuição e concepção visual de como ficaria disposta as informações da forma mais direta possível.

O trabalho se destina principalmente a tomadores de decisão, organizações ambientais, comunidades locais e setores produtivos que buscam entender o desmatamento da Amazônia e orientar medidas de conservação e conscientização do povo. Nesses grupos, incluem-se governos em âmbitos nacional (Brasil) como internacionais também (países da América do Sul que também possuem parte da Floresta Amazônica), ONGs que promovem campanhas de proteção, empresas que precisam alinhar atividades à sustentabilidade e comunidades que vivenciam diretamente os efeitos da devastação.

Além disso, pesquisadores e estudantes/interessados em geral também se beneficiam do conteúdo, pois encontram dados e análises para desenvolver estudos acadêmicos, projetos de pesquisa ou reportagens sobre a importância ecológica e social da floresta. Assim podemos ajudar a contribuir cada vez mais com a disseminação da importância de entendermos a realidade do nosso “quintal de casa”

Fonte de dados utilizada: <http://www.obt.inpe.br/OBT/assuntos/programas/amazonia/prodes> (acessada em fev/2025). Segue abaixo uma prévia do dataset base utilizado, junto com outros datasets utilizados e extraídos das fontes listadas no final deste trabalho.

Ano/Estados AC AM AP MA MT PA RO RR TO AMZ LEGAL

1988 620 1510 60 2450 5140 6990 2340 290 1650 21050

1989 540 1180 130 1420 5960 5750 1430 630 730 17770

1990 550 520 250 1100 4020 4890 1670 150 580 13730

1991 380 980 410 670 2840 3780 1110 420 440 11030

1992 400 799 36 1135 4674 3787 2265 281 409 13786

1993 482 370 0 372 6220 4284 2595 240 333 14896

1994 482 370 0 372 6220 4284 2595 240 333 14896

1995 1208 2114 9 1745 10391 7845 4730 220 797 29059

Visualização e Análise dos Resultados

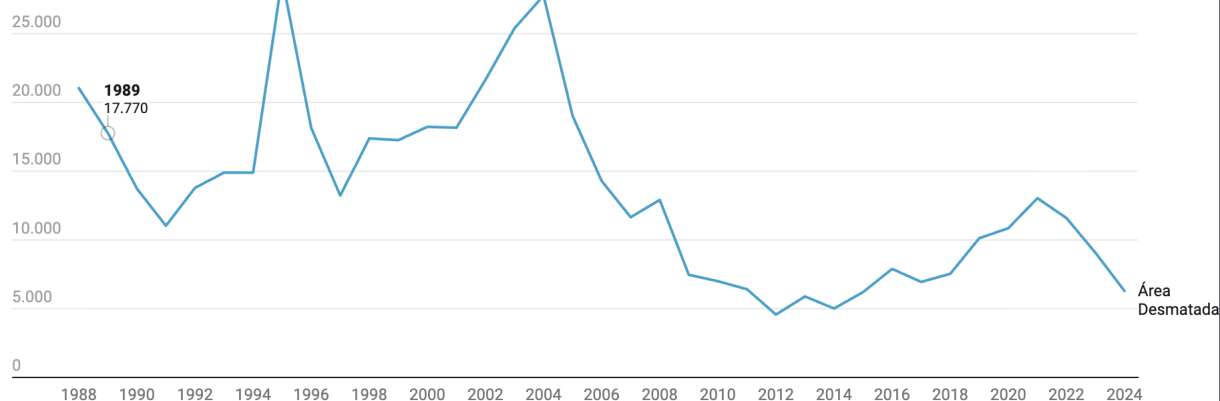
Evolução Histórica do Desmatamento (1988–2025)

Conforme gráfico abaixo, podemos notar o pico no ano de 1995 e 2004, seguido por uma forte redução de 2005 em diante, com uma tendência de alta no início dos anos 2020 em

diante.

Desmatamento Floresta Amazônica (área desmatada em km²)

Criado por Matheus Kemer - 2025



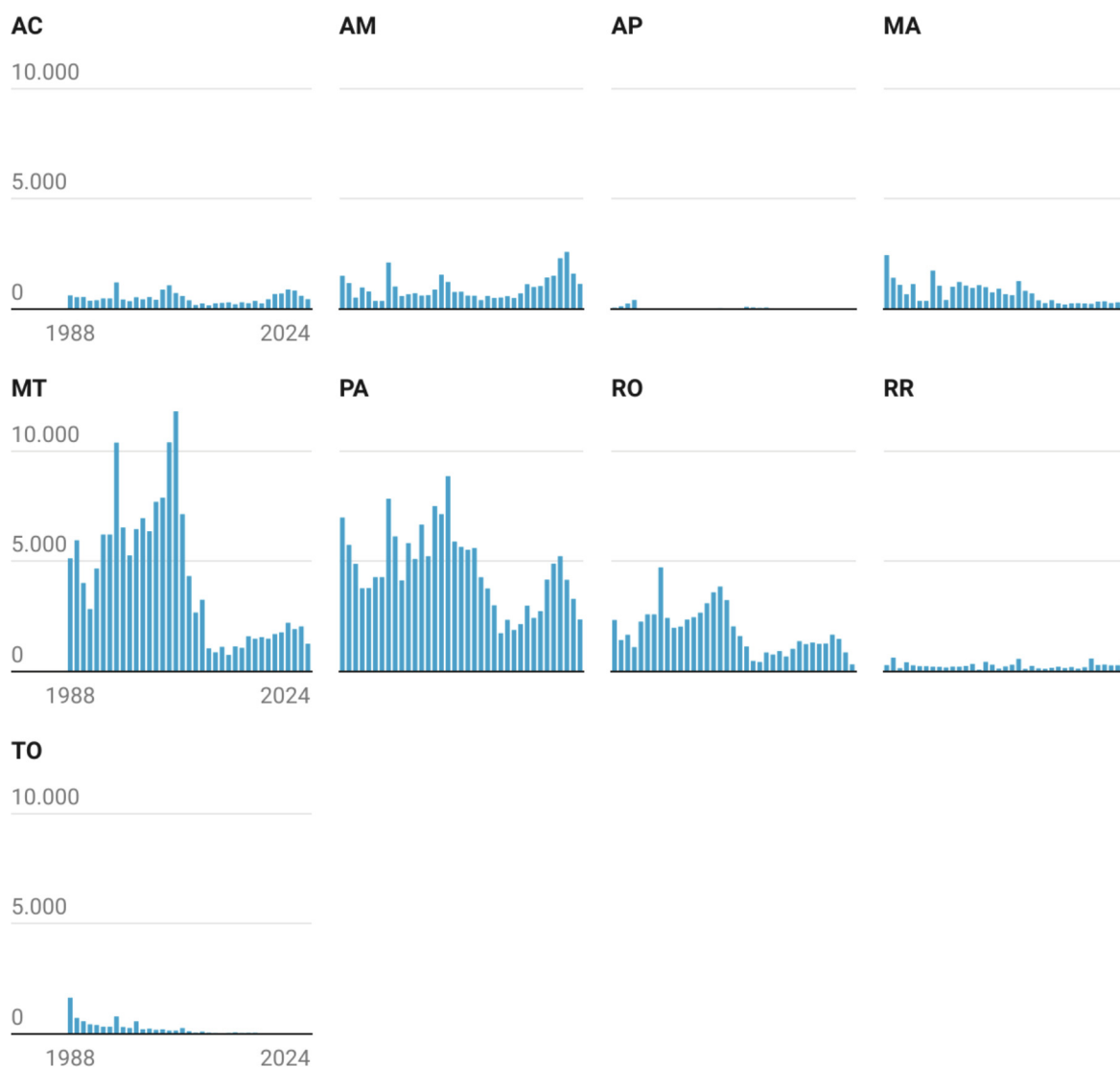
Fonte: PRODES · Obter dados · Criado com Datawrapper

(para acessar o gráfico interativo, acesse: <https://datawrapper.dwcdn.net/Ql6EF/2>)

Nos anos 1990 e início dos anos 2000, o desmatamento atingiu níveis extremamente elevados, chegando a aproximadamente 27 mil km² em 2004, praticamente o mesmo valor encontrado no topo histórico atingido em 1995.

Isso se deu porque na época o desmatamento atingiu números equivalentes a mais de 20 mil km² em alguns períodos, devido a expansão de fazendas de gado, grileiros/garimpeiros, plantações de soja, madeiras, etc. Sempre visando o lucro e o comércio. Felizmente houve pressão interna e externa (internacional) para, em meados dos anos 2000, forçar o governo a implementar um Plano de Ação para Prevenção e Controle do Desmatamento na Amazônia Legal (PPCDAm), adotando medidas para um maior controle das áreas protegidas, demarcando terras indígenas e fortalecendo a fiscalização através de satélites e do próprio Ibama. Essas políticas tiveram êxito notável: em menos de uma década o desmatamento caiu cerca de 84%, atingindo em 2012 o menor valor histórico de 4.571 km², ante os 27.772 km² de 2004.

Essa redução representou um caso de sucesso ambiental mundial, frequentemente citado como prova de que é possível conciliar proteção florestal com desenvolvimento. Como a Floresta Amazônica possui uma extensão interestadual e internacional, abaixo segue uma representação gráfica do desmatamento por estado no mesmo período de tempo (1988-2024).



Fonte: PRODES • Criado com Datawrapper

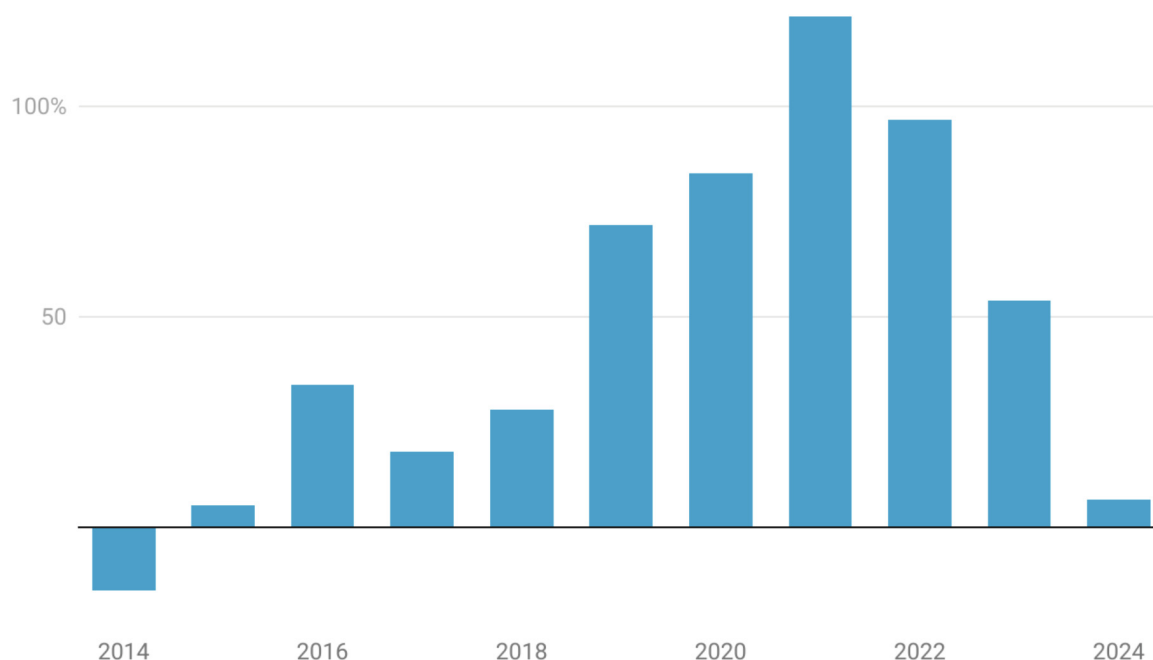
(para acessar o gráfico interativo, acesse: <https://datawrapper.dwcdn.net/5mRZZ/2/>)

Como prova de que a única forma de diminuir o desmatamento é manter padrões rígidos e elevados de fiscalização, monitoramento e conscientização humana, a partir do ano de 2013 lamentavelmente houve uma inversão na tendência, após quase uma década de decaimento do desmatamento, de 2013 em diante voltou a ocorrer aumento nessa taxa. Entre 2013 e 2020 houve uma retomada contínua, culminando em 10.129 km² desmatados em 2019 e aproximadamente 11 mil km² em 2020. Isso significa que o Brasil, tanto comogoverno quanto como população, falhou em cumprir a meta climática estipulada em limitar o desmatamento anual a no máximo 3.925 km² em 2020 (80% de redução em relação à média 1996-2005) .

Abaixo segue um gráfico mostrando, em porcentagem, a evolução do desmatamento dos anos subsequentes à 2013.

Porcentagem desmatamento comparado com 2013

Criado Por Matheus Kemer - 2025



Source: PRODES • Created with Datawrapper

(para acessar o gráfico interativo, acesse: <https://datawrapper.dwcdn.net/T0ApN/1>)

Para se ter uma ideia, o ano de 2020 fechou em torno de 182% acima da meta, resultando em enormes emissões de carbono. Levando em consideração apenas o desmatamento bruto do respectivo ano, foram liberadas cerca de 648 milhões de toneladas de CO₂ para a atmosfera, volume superior às emissões anuais de muitos países industrializados. Mudanças na legislação ambiental de 2012 culminou nesse aumento dos resultados negativos, quando houve uma revisão e mudança do Código Florestal. A partir de 2019 também houve uma perda grande nessa governança ambiental, causando diminuição de fiscalização e enfraquecimento das agências responsáveis por esse monitoramento, como o IBAMA E ICMBio.

Não podemos deixar de citar a pandemia do COVID-19 como outro fator que interrompeu planos, projetos e fiscalização que ocorriam de forma presencial e, que devido à quarentena, permitiu que muitas companhias e empresas com más intenções se aproveitassem da fragilidade mundial acarretada pelo momento. Em 2022, a taxa anual de desmatamento foi estimada em 11.568km², um leve decréscimo em relação a 2021 (13.235km²), ainda assim quase três vezes acima do limite estipulado para cumprir os compromissos climáticos do Brasil. Consequências para Comunidades Indígenas e Populações Locais

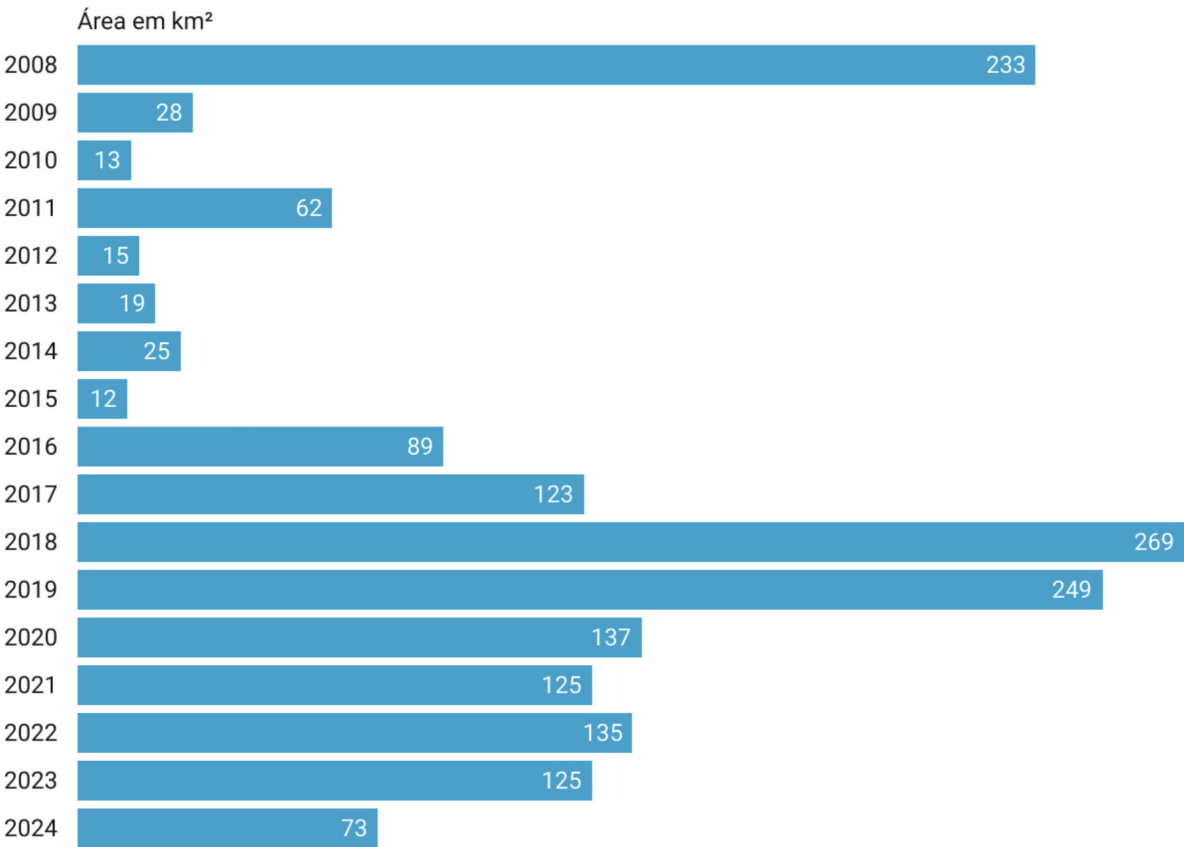
A Amazônia não é apenas uma região rica em biodiversidade, mas também o lar de aproximadamente 2,2 milhões de indígenas pertencentes a mais de 500 etnias e tribos diferentes. Essas comunidades tradicionais, que por direito são mais donas que nós de toda essa terra chamada de Brasil, dependem da floresta para sua subsistência física e cultural, obtendo alimentos, medicamentos através da flora, materiais de construção e mantendo práticas espirituais/religiosas centenárias ligadas à terra, à natureza e a essa conexão entre homem e espiritualidade.

Devido a essa intensa disputa e abuso do “homem branco”, conflitos fundiários, violência e insegurança em várias esferas, como alimentação (caça, pesca e plantações deixam de existir) ou também a própria segurança física das tribos e de sua história/cultura que correm riscos devido à inconstância e incerteza da diminuição do desmatamento e invasões às terras protegidas.

Abaixo segue um gráfico mostrando essa instabilidade relacionada ao desmatamento das terras indígenas. Mesmo com uma diminuição considerável no ano de 2024, ainda podemos dizer que é incerto o que podemos esperar nos próximos anos caso medidas mais rígidas de fiscalização, segurança e monitoramento não sejam tomadas pelas entidades responsáveis.

Desmatamento em Terras Indígenas na região Amazônica

Criado Por Matheus Kemer - 2025



Source: IMAZON • Created with Datawrapper

(para acessar o gráfico interativo, acesse: <https://datawrapper.dwcdn.net/iV9o1/1>)

Por outro lado, terras indígenas e unidades de conservação são barreiras eficazes contra o desmatamento: onde há gestão tradicional, a perda de floresta é menor e a proteção do carbono maior, beneficiando o clima. Além disso, essas áreas preservam saberes e plantas medicinais, reforçando o papel de guardiões ambientais que os povos indígenas exercem.

Proteger os direitos indígenas, portanto, não é apenas questão de justiça social ou uma estratégia de conservação florestal comprovadamente efetiva, é uma questão moral e humana que envolve direitos básicos à segurança, moradia e alimentação. Influência no Aquecimento Global e no Clima.

Como é de conhecimento geral, a Floresta Amazônica é crucial para o clima mundial, atuando como um enorme “pulmão” do mundo para filtrar o carbono através de suas árvores, que armazenam cerca de 48 bilhões de toneladas de carbono armazenadas neste momento graças às árvores e a vegetação, ajudando a diminuir o CO₂ atmosférico. Regionalmente, a floresta mantém um ciclo hídrico que é essencial para esse funcionamento completo, com suas árvores formando “rios voadores” que geram chuvas em toda a América do Sul. Com o desmatamento, queimadas e exploração florestal, há uma diminuição nas chuvas e consequentemente um aumento da frequência

de incêndios, criando um ciclo vicioso: menos floresta, clima mais seco, mais queimadas e mais emissões de carbono.

Cientistas alertam que se continuarmos com esse progressão do desmatamento e o aquecimento global ultrapassar/aumentar 2 a 3°C, a Amazônia pode atingir um ponto de inflexão, e, conforme dito anteriormente, podendo se transformar numa savana degradada e comprometendo a regulação climática global, o que dificultaria o cumprimento das metas do Acordo de Paris (com objetivo de justamente limitar esse aquecimento global para restringir o aumento a, no máximo 1,5°C).

Conclusões

Conforme visto nos gráficos e textos apresentados, ainda há uma oscilação muito grande referente ao desmatamento da Amazônia. Graças a pressão geral, as políticas impostas entre 2004 e 2012 reduziram essa perda florestal em quase 80%, mas, a partir de 2013, com uma ênfase maior a partir de 2019, houve uma retomada grande de práticas destrutivas.

De acordo com o cenário atual, é de suma importância a realização de medidas e ações imediatas, tanto em âmbitos de monitoramento e conscientização geral, como também na esfera fiscal/governamental para aplicações de multas e sanções às entidades e empresas que preferem degradar cada vez mais um patrimônio histórico, cultural e natural em troca de retornos e lucros puramente financeiro.

Para concluir, gostaria de deixar uma opinião minha: Espero que nós, como espécie dominante deste planeta, consigamos entender que o universo (estrelas, planetas, asteroides, etc) está cheio de matéria prima, pedras preciosas e quaisquer outros recursos que extraímos não só da Floresta Amazônica mas também de todo nosso planeta, simplesmente para fins financeiros. Precisamos entender que, até onde sabemos e descobrimos, o oxigênio é um recurso muito mais escasso quando começamos a olhar num âmbito além Planeta Terra.

Referências

<https://terrabrasilis.dpi.inpe.br/app/dashboard/deforestation/biomes/legal>
<https://www.greenpeace.org/brasil/protetendo-o-desconhecido/>
<https://www.greenpeace.org/brasil/protetendo-o-desconhecido/>
<https://www.science.org/doi/10.1126/science.1246752>
<https://www.fundobrasil.org.br/blog/incendios-e-os-povos-indigenas>
<http://riu.ufam.edu.br/handle/prefix/6135>
<https://www.globalforestwatch.org>
<https://revistapesquisa.fapesp.br/amazonia-agora-e-fonte-de-co2/>
<https://www.globalforestwatch.org/>
<https://www.nature.com/articles/s41586-021-03626-1>
<https://www.socioambiental.org/>
<https://www.iucn.org/amazon/rates>

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

Código e as imagens obtidas:

```
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import permutations
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import PCA

1) Algoritmo Genético - Problema do Caixeiro Viajante
# Parâmetros do Algoritmo
NUM_CIDADES = 100
TAMANHO_POPULACAO = 100
NUM_GERACOES = 1000
TAXA_MUTACAO = 0.01
TAXA_CROSSOVER = 0.90
# Gerar cidades aleatórias no espaço 100x100
cidades = np.random.rand(NUM_CIDADES, 2) * 100
# Função para calcular a distância euclidiana entre cidades
def distancia(caminho):
    return sum(np.linalg.norm(cidades[caminho[i]] - cidades[caminho[i + 1]]))
for i in range(len(caminho) - 1))
# Criar população inicial de caminhos aleatórios
populacao = [np.random.permutation(NUM_CIDADES).tolist() for _ in
range(TAMANHO_POPULACAO)]
# Preservar a cidade inicial e final na posição 0 e 100
for ind in populacao:
    primeira_cidade = ind[0]
    ind.append(primeira_cidade)
melhor_caminho = None
melhor_distancia = float('inf')
def selecao_torneio(populacao, k=5):
```

```

    selecionados = random.sample(populacao, k)
    return min(selecionados, key=distancia)
def crossover_ox(parent1, parent2):
    tam = len(parent1)
    start, end = sorted(random.sample(range(1, tam - 1), 2))
    filho = [-1] * tam
    filho[start:end] = parent1[start:end]
    pos_filho = set(filho)
    indice = end
    for gene in parent2:
        if gene not in pos_filho:
            if indice >= tam:
                indice = 1
            filho[indice] = gene
            indice += 1
    return filho
def mutacao(caminho):
    if random.random() < TAXA_MUTACAO:
        i, j = sorted(random.sample(range(1, len(caminho) - 1), 2))
        caminho[i], caminho[j] = caminho[j], caminho[i]
    return caminho
def plotar_caminho(caminho, titulo):
    plt.figure(figsize=(8, 8))
    caminho_coords = cidades[caminho]
    plt.plot(caminho_coords[:, 0], caminho_coords[:, 1], 'bo-', markersize=5)
    plt.plot(caminho_coords[0, 0], caminho_coords[0, 1], 'ro', markersize=10,
label='Início')
    plt.title(titulo)
    plt.legend()
    plt.show()
# Evolução do algoritmo
for geracao in range(NUM_GERACOES):
    nova_populacao = []
    populacao = sorted(populacao, key=distancia)
    melhor_da_geracao = populacao[0]
    if distancia(melhor_da_geracao) < melhor_distancia:
        melhor_caminho, melhor_distancia = melhor_da_geracao,
distancia(melhor_da_geracao)
    nova_populacao.append(melhor_da_geracao)

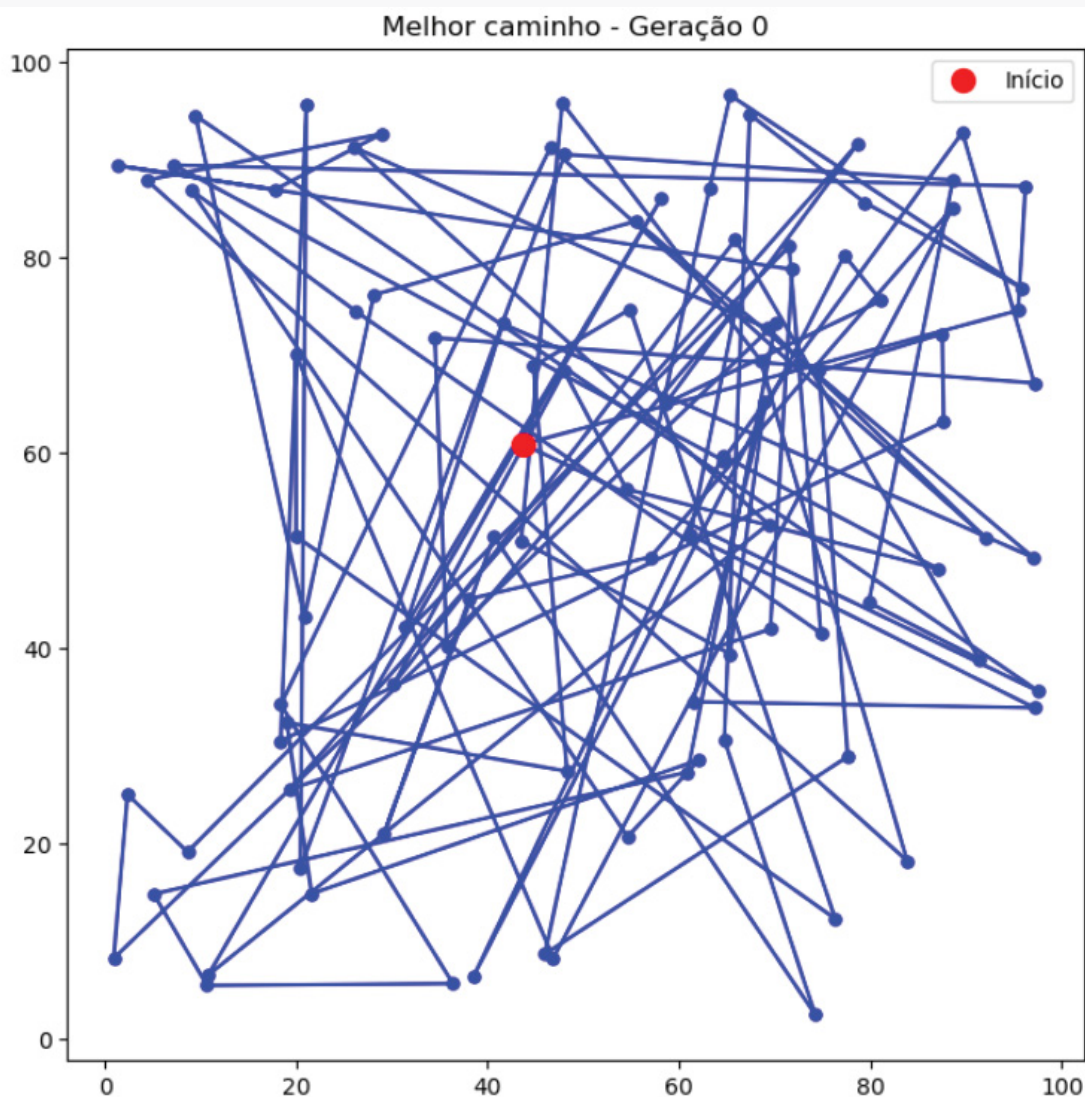
    while len(nova_populacao) < TAMANHO_POPULACAO:
        pai1, pai2 = selecao_torneio(populacao), selecao_torneio(populacao)
        if random.random() < TAXA_CROSSOVER:
            filho = crossover_ox(pai1, pai2)
        else:
            filho = pai1[:]
        filho = mutacao(filho)
        nova_populacao.append(filho)

    populacao = nova_populacao

if geracao in [0, NUM_GERACOES - 1]:
    plotar_caminho(melhor_caminho, f"Melhor caminho - Geração {geracao}")

```

```
print(f"Melhor distância encontrada: {melhor_distancia}")
```



Fonte: O Autor(2025)

2) Compare a representação de dois modelos vetoriais

Definição de algumas frases curtas

```
textos = [
    "O cachorro late à noite.",
    "O cão uiva para a lua.",
    "A gata dorme no sofá.",
    "O felino repousa tranquilamente.",
    "Os cães são animais fiéis.",
    "Os gatos são independentes e misteriosos."
]
```

Vetorização com TF-IDF

```
vectorizer_tfidf = TfidfVectorizer()
```

```
X_tfidf = vectorizer_tfidf.fit_transform(textos).toarray()
```

Vetorização com Bag-of-Words

```
vectorizer_bow = CountVectorizer()
```

```
X_bow = vectorizer_bow.fit_transform(textos).toarray()
```

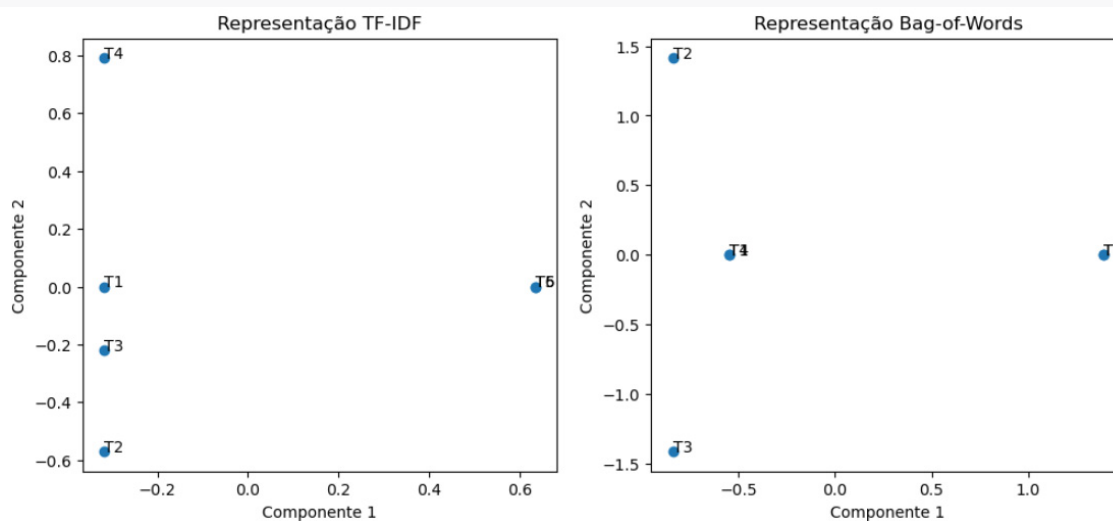
Redução de dimensionalidade com PCA para 2D

```
pca = PCA(n_components=2)
```

```

X_tfidf_pca = pca.fit_transform(X_tfidf)
X_bow_pca = pca.fit_transform(X_bow)
# Plot dos vetores projetados
fig, axs = plt.subplots(1, 2, figsize=(12, 5))
def plot_pca(ax, X_pca, title):
    ax.scatter(X_pca[:, 0], X_pca[:, 1])
    for i, txt in enumerate(range(len(textos))):
        ax.annotate(f'T{i+1}', (X_pca[i, 0], X_pca[i, 1]))
    ax.set_title(title)
    ax.set_xlabel("Componente 1")
    ax.set_ylabel("Componente 2")

```



Fonte: O Autor (2025)

```

plot_pca(axs[0], X_tfidf_pca, "Representação TF-IDF")
plot_pca(axs[1], X_bow_pca, "Representação Bag-of-Words")
plt.show()
# Exibir os vetores textualmente
print("Vetores TF-IDF:\n", X_tfidf)
print("\nVetores Bag-of-Words:\n", X_bow)

```

Vetores TF-IDF:

```
[[0.          0.57735027 0.          0.          0.          0.
  0.          0.          0.          0.          0.57735027 0.
  0.          0.          0.57735027 0.          0.          0.
  0.          0.          0.          0.          ]
 [0.          0.          0.          0.5          0.          0.
  0.          0.          0.          0.          0.          0.5
  0.          0.          0.          0.          0.5          0.
  0.          0.          0.          0.5          ]
 [0.          0.          0.          0.          0.5          0.
  0.          0.5          0.          0.          0.          0.
  0.          0.5          0.          0.          0.          0.
  0.5          0.          0.          0.          ]
 [0.          0.          0.          0.          0.          0.57735027
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.57735027
  0.          0.          0.57735027 0.          ]
 [0.47974754 0.          0.47974754 0.          0.          0.
  0.47974754 0.          0.          0.          0.          0.
  0.          0.          0.          0.39339985 0.          0.
  0.          0.39339985 0.          0.          ]
 [0.          0.          0.          0.          0.          0.
  0.          0.          0.47974754 0.47974754 0.          0.
  0.47974754 0.          0.          0.39339985 0.          0.
  0.          0.39339985 0.          0.          ]]
```

Vetores Bag-of-Words:

```
[[0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0]
 [0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1]
 [0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0]
 [1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1 0 0]]
```

Fonte: O Autor (2025)