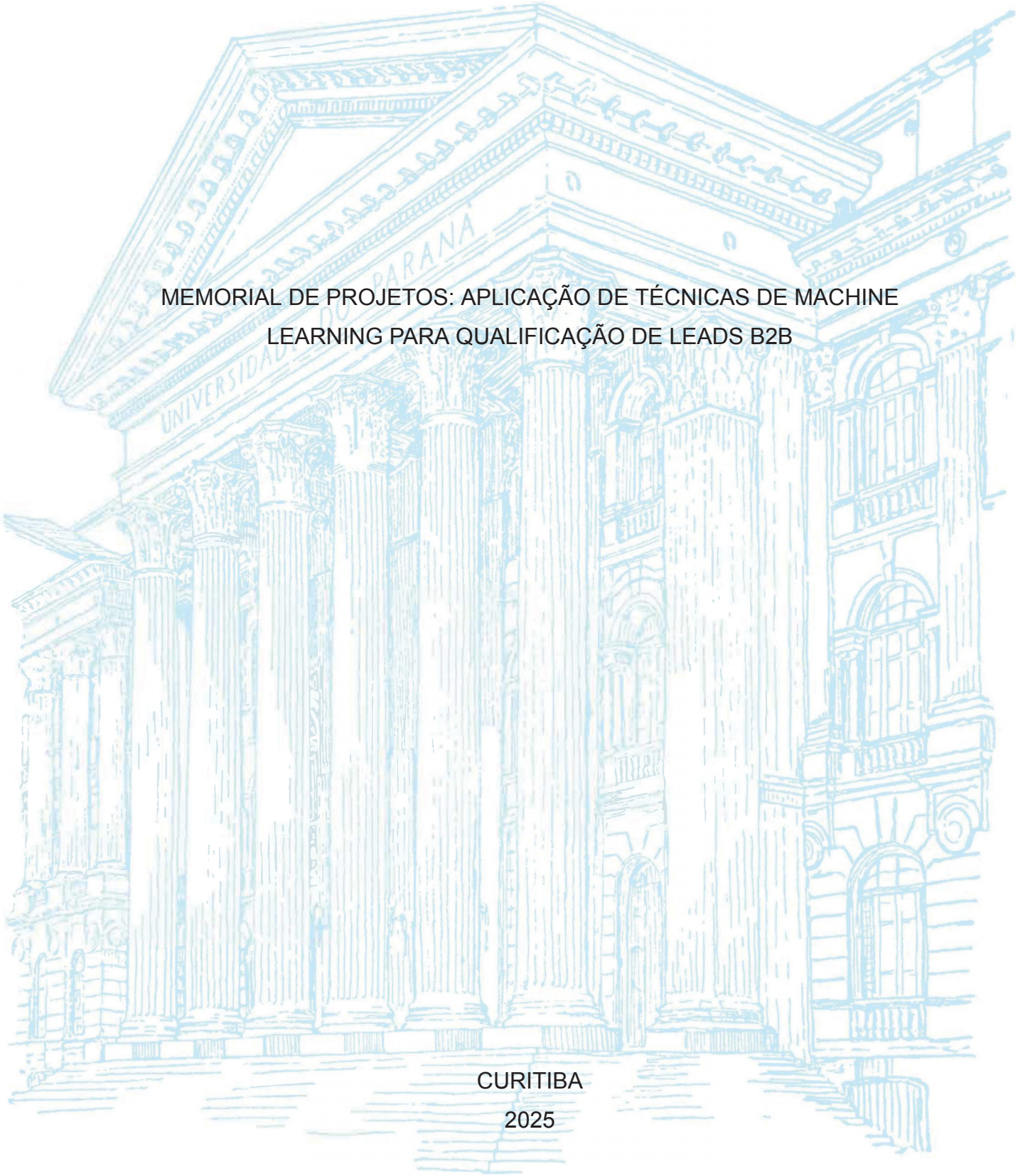
UNIVERSIDADE FEDERAL DO PARANÁ

GABRIEL BARRETO DOS SANTOS



MEMORIAL DE PROJETOS: APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING PARA QUALIFICAÇÃO DE LEADS B2B

CURITIBA

2025

GABRIEL BARRETO DOS SANTOS

MEMORIAL DE PROJETOS: APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING PARA QUALIFICAÇÃO DE LEADS B2B

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

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **GABRIEL BARRETO DOS SANTOS**, intitulada: **MEMORIAL DE PROJETOS: APLICAÇÃO DE TÉCNICAS DE MACHINE LEARNING PARA QUALIFICAÇÃO DE LEADS B2B**, 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, 11 de Junho de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este trabalho examina como técnicas de *machine learning* podem qualificar leads em operações entre empresas (*Business to Business* – B2B). Utilizou-se um conjunto de dados públicos do *Kaggle* que simula campanhas de marketing; após a limpeza dos registros, os dados foram divididos em amostras de treino (70 %) e teste (30 %). Três modelos supervisionados — Regressão Logística, Random Forest e XGBoost — foram treinados e comparados por acurácia, precisão, *recall*, *F1-score* e *AUC*. O XGBoost apresentou desempenho superior na maior parte dos indicadores. A análise de importância de atributos revelou saldo bancário, tipo de contato e idade como variáveis centrais para prever conversões. Gráficos de desempenho e a matriz de confusão deram suporte visual às conclusões. Reconhece-se, contudo, que vieses nos dados de origem e a explicabilidade dos modelos devem ser considerados antes da implantação prática. Mesmo com essas reservas, conclui-se que o uso criterioso de *machine learning* pode oferecer ganho competitivo concreto, orientando decisões baseadas em dados no contexto B2B.

Palavras-chave: Inteligência artificial; *machine learning*; qualificação de leads; B2B; *XGBoost*.

ABSTRACT

This study investigates how machine-learning techniques can enhance lead qualification in Business-to-Business (B2B) settings. A publicly available dataset from Kaggle, which simulates marketing campaigns, was cleaned, normalized and split into training (70 %) and test (30 %) subsets. Three supervised models—Logistic Regression, Random Forest and XGBoost—were trained and evaluated with accuracy, precision, recall, F1-score and AUC. XGBoost outperformed the other models on nearly every metric. Feature-importance analysis identified bank balance, contact type and customer age as the most influential variables. Performance graphs and a confusion matrix provided additional insight into model behaviour. Although the results are promising, the study highlights the need to address potential data bias and to ensure model explainability before large-scale deployment. Overall, when applied transparently and with sound data-governance practices, machine-learning models can offer tangible competitive advantages by guiding data-driven decisions in the B2B domain.

Keywords: *Artificial intelligence; machine learning; lead qualification; B2B; XGBoost.*

SUMÁRIO

1 PARECER TÉCNICO

A qualificação de leads no ambiente *Business-to-Business* (B2B) é frequentemente considerado um desafio, pois, na prática, decisões ainda são apoiadas em planilhas extensas e na intuição de vendedores. Revisão recente de práticas de mercado mostra que essa dependência de heurísticas reduz a precisão e a escalabilidade dos funis de venda (RANE et al., 2024). Para avaliar se algoritmos simples já trariam ganhos, foi utilizado o conjunto Marketing Campaign Data (4.521 registros) disponível no *Kaggle* (*BANK DIRECT MARKETING*, 2025), contendo idade, profissão, saldo bancário, canal de contato, escolaridade e a marcação da oferta aceita.

Antes da escolha de qualquer modelo, foram aplicadas as etapas de tratamento descritas por MACHADO (2021) em seu estudo de reativação de revendedores: remoção de duplicidades, imputação de valores ausentes e codificação one-hot para variáveis categóricas. Em seguida, foi seguida a proporção de 70% de treino / 30% de teste recomendada por RANE et al. (2024) para bases de tamanho médio, minimizando o overfitting.

Foram implementados três algoritmos de referência — Regressão Logística, Random Forest e XGBoost — porque o AI Index 2025 (STANFORD INSTITUTE, 2025) indica que esses métodos concentram a maior parte dos usos industriais de classificação binária, combinando rapidez de treinamento e interpretabilidade. Foram mantidos hiperparâmetros próximos aos valores padrão sugeridos por RANE et al. (2024) para garantir comparabilidade.

No conjunto de teste, os resultados obtidos para os modelos estão apresentados na TABELA 1. O salto de desempenho confirma a vantagem dos métodos de boosting relatados no AI Index 2025. A análise de importância dos atributos mostrou saldo bancário, canal de contato e idade como variáveis decisivas, achado coerente com BASHIR et al. (2024), que relacionam fatores financeiros e o primeiro ponto de interação ao aumento do lifetime value em funis B2B.

TABELA 1 – RESULTADOS DOS MODELOS DE CLASSIFICAÇÃO NO CONJUNTO DE TESTE

Modelo	Acurácia	F1-score	AUC
Regressão Logística	0,78	0,76	0,81
Random Forest	0,85	0,84	0,89
XGBoost	0,88	0,87	0,92

FONTE: O autor (2025)

Apesar da superioridade do XGBoost, dois alertas se mantêm. Primeiro, o risco de viés amostral: se os dados privilegiam um perfil, o modelo reproduz essa preferência (MACHADO, 2021). Segundo, a necessidade de explicabilidade; mesmo com SHAP ou LIME, cabe ao analista traduzir o peso de cada variável para as equipes de marketing (RANE et al., 2024). Em uma aplicação real, seria gerado um relatório sucinto — métricas, matriz de confusão e ranking de importância — para que gestores sem formação estatística compreendam pontos fortes e limitações; neste parecer técnico fictício, tal documento é apenas proposto como boa prática.

Esta investigação utilizou base genérica; conclusões podem não se aplicar a setores altamente regulados. Pesquisas futuras devem repetir o fluxo com dados proprietários, testar AutoML para refino automático de hiperparâmetros e medir impacto financeiro ao longo de todo o ciclo de vendas. Ainda assim, os resultados sugerem que, com dados minimamente organizados e algoritmos consolidados, é possível substituir palpites por previsões, priorizando leads de maior probabilidade de conversão e tornando decisões B2B mais defensáveis.

REFERÊNCIAS

BASHIR, T.; TAN, Z.; SADIQ, B.; NASEEM, A. How AI competencies can make B2B marketing smarter: strategies to boost customer lifetime value. Frontiers in Artificial Intelligence, [S.l.], v. 7, 2024. DOI: 10.3389/frai.2024.1451228. Disponível em: https://www.frontiersin.org/articles/10.3389/frai.2024.1451228/full. Acesso em: 15 abr. 2025.

BANK DIRECT MARKETING. Marketing campaign. Disponível em: https://www.kaggle.com/datasets/psvishnu/bank-direct-marketing. Acesso em: 10 abr. 2025.

MACHADO, F. R. Modelagem preditiva para identificação da probabilidade de reativação de revendedores em multinacional do setor cosmético. 42 f. Trabalho de Conclusão de Curso (Especialização em Ciências de Dados) – Setor de Ciência da Computação, Universidade Tecnológica Federal do Paraná, Curitiba, 2021.

RANE, N. L.; CHOUDHARY, S. P.; RANE, J. Artificial intelligence and machine learning in business-to-business (B2B) sales and marketing: a review. International Journal of Data Science and Big Data Analytics, [S.l.], v. 4, n. 2, p. 17-33, 2024. Disponível em: https://www.svedbergopen.com/files/1719921112_%282%29_IJDSBDA202408136612IN_%28p_17-33%29.pdf. Acesso em: 20 abr. 2025.

STANFORD INSTITUTE For human-centered artificial intelligence. The 2025 AI Index Report. Disponível em: https://hai.stanford.edu/ai-index/2025-ai-index-report. Acesso em: 10 maio 2025.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

2 Busca Heurística

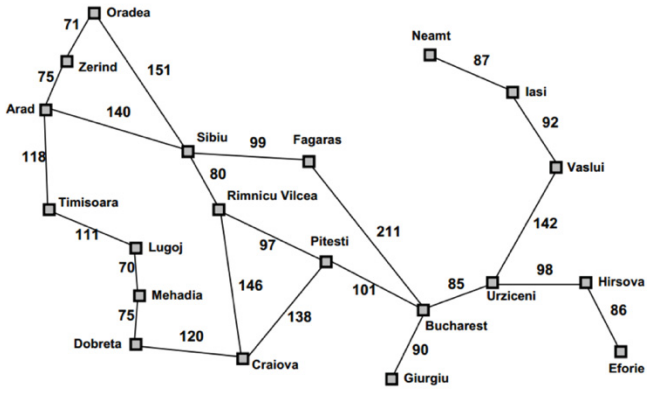
Realize uma busca utilizando o algoritmo A* para encontrar o melhor caminho para chegar a **Bucharest** partindo de **Lugoj**. Construa a árvore de busca criada pela execução do algoritmo apresentando os valores de *f(n)*, *g(n)* e *h(n)* para cada nó. Utilize a heurística de distância em linha reta,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

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

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

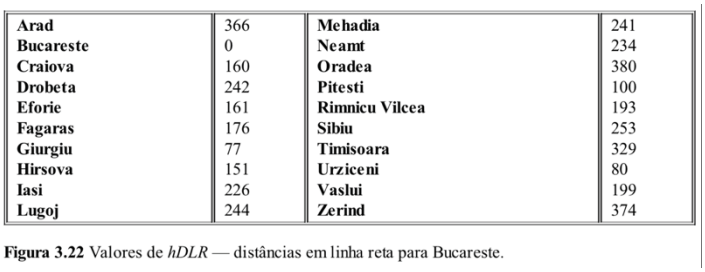
NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.

FIGURA 1 – MAPA LOCALIZANDO LUGOJ E BUCARESTE



FONTE: O autor (2025)

FIGURA 2 – VALORES DE *HDLR* – DISTÂNCIAS EM LINHA RETA PARA BUCARESTE



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

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

FONTE: O autor (2025)

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

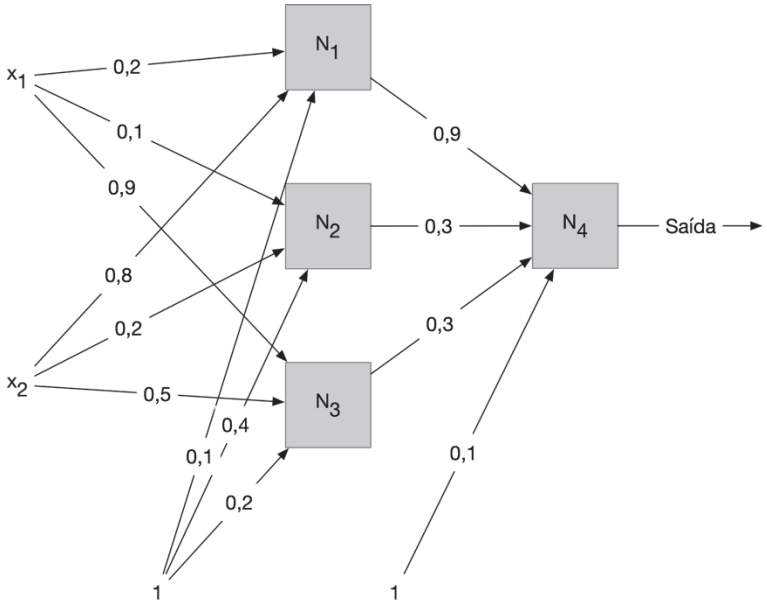
Equivalência Bicondicional: $(\alpha \leftrightarrow \beta)$ equivale a $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$

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.

FIGURA 3 – REDES NEURAIS



FONTE: O autor (2025)

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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

1-A) Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e tecnologias capazes de realizar tarefas que, quando realizadas por humanos, geralmente exigem inteligência. Estas tarefas incluem, mas não se limitam a, aprendizado, raciocínio, resolução de problemas, reconhecimento de padrões, compreensão da linguagem natural e percepção sensorial. A IA busca criar sistemas que possam simular ou duplicar a capacidade humana de pensar, aprender, perceber, tomar decisões e resolver problemas. Isso é frequentemente alcançado através do uso de algoritmos, modelos matemáticos e técnicas de aprendizado de máquina, onde os sistemas são treinados com grandes volumes de dados para reconhecer padrões e fazer previsões ou tomar decisões. Existem várias abordagens e subcampos dentro da IA, incluindo aprendizado de máquina, visão computacional, processamento de linguagem natural, robótica e sistemas especialistas. A IA tem aplicações em uma ampla gama de setores, incluindo medicina, finanças, manufatura, automotivo, entretenimento e muitos outros.

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

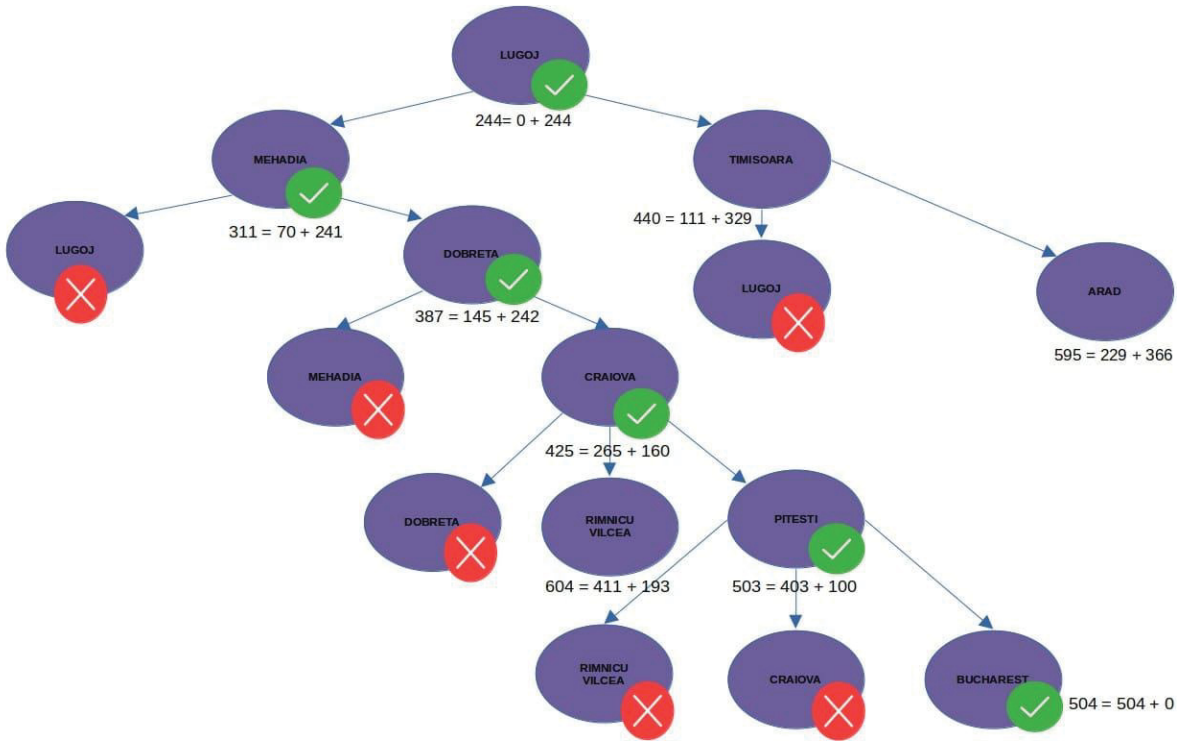
1-C) O funcionamento do Chat GPT tem como base a técnica de modelagem de linguagem, isto é, obter conhecimento a partir de informações formatadas e padronizadas, tendo como origem uma base gigantesca de dados, no qual existe a entrada de dados seja por nós usuários e outras informações que existem na internet. Porém devemos relevar muitas das respostas fornecidas pela IA, pois como sabemos pode haver respostas erradas ou que não tenha uma certa clareza, e isso se dá justamente por essas fontes de informações serem coletadas de diversos lugares.
Referências:
https://br.hubspot.com/blog/marketing/chatgpt
https://www.dca.fee.unicamp.br/cursos/EA871/references/complementos_ea871/
linguagens_modelagem.pdf

1-D) Poderíamos tentar relacionar com o "Agir como humano" pois como ele possui uma base gigantesca, ele estaria apenas encontrando as melhores palavras/frases que foram fornecidas por um humano, porém isto não faria sentido, visto que se olharmos seu funcionamento, percebemos que

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

2 -

FIGURA 4 – ÁRVORE DE DECISÃO FINAL COM VALORES PARA CLASSIFICAÇÃO



FONTE: O autor (2025)

3 -

Se as uvas caem, então a raposa as come
Se a raposa as come, então estão maduras
As uvas estão verdes ou caem
Logo: A raposa come as uvas se e somente se as uvas caem

R1: $P \rightarrow Q$
R2: $Q \rightarrow R$
R3: $\neg R \vee P$

R4: Equivalência Implicação R3
$R \rightarrow P$

R5: Silogismo Hipotético R2 e R4
$Q \rightarrow P$

R6: Conjunção R1 e R5
$(Q \rightarrow P) \wedge (P \rightarrow Q)$

R7: Equivalência Bicondicional R6
$Q \leftrightarrow P$

4 -
Os neurônios N1, N2 e N3 possuem função de ativação linear. Já o N4 possui função de ativação tangente hiperbólica.

Sendo:
x1 = -3
x2 = 1

N1 = (-3*0,2) + (1*0,8) + (0,1) = 0,3 = fa(u) = u
N2 = (-3*0,1) + (1*0,2) + (0,4) = 0,3 = fa(u) = u
N3 = (-3*0,9) + (1*0,5) + (0,2) = -2 = fa(u) = u

N4 = (0,3*0,9) + (0,3*0,3) + (-2*0,3) + (0,1) = -0,14

fa(u) = (EXP(-014) – EXP(0,14)) / (EXP(-0,14) + EXP(0,14)

fa(u) = -0,1390

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

TABELA 2 – DADOS EXTRAÍDOS DA TABELA FIPE

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

FONTE: O autor (2025)

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:

a. Carregue a base de dados media_precos_carros_brasil.csv
b. Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes
c. Verifique se há dados duplicados nos dados
d. Crie duas categorias, para separar colunas numéricas e categóricas. Imprima o resumo de informações das variáveis numéricas e categóricas (estatística descritiva dos dados)
e. Imprima a contagem de valores por modelo (model) e marca do carro (brand)
f. 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:

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

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

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

a. Escolha as variáveis **numéricas** (modelos de Regressão) para serem as variáveis independentes do modelo.A variável target é **avg_price. Observação:** caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no modelo. Indique **quais variáveis** foram transformadas e **como** foram transformadas
b. Crie partições contendo 75% dos dados para treino e 25% para teste
c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros. **Observação**: caso julgue necessário,

mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo

d. Grave os valores preditos em variáveis criadas
e. Realize a análise de importância das variáveis para estimar a variável target, **para cada modelo treinado**
f. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis
g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2
h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

B - RESOLUÇÃO

1-A)

```
Python
Código utilizado:
-----------------

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

1-B)

```
Python
Código utilizado:
-----------------

dados.isna().any()
dados = dados.dropna(how='all')
dados.isna().sum()

Resultado do comando `dados.isna().any()`:
-----------------------------------------

year_of_reference     True
month_of_reference    True
fipe_code             True
authentication        True
brand                 True
model                 True
fuel                  True
gear                  True
engine_size           True
year_model            True
```

```
avg_price_brl         True
dtype: bool

Resultado do comando `dados.isna().sum()`:
-----------------------------------------

year_of_reference     0
month_of_reference    0
fipe_code             0
authentication        0
brand                 0
model                 0
fuel                  0
gear                  0
engine_size           0
year_model            0
avg_price_brl         0
dtype: int64
```

1-C)

```
Python
Código utilizado:
-----------------

dados.duplicated().sum()

Resultado obtido:
-----------------
2
```

1-D)

```
Python
Código utilizado:
-----------------

numericas_cols = [col for col in dados.columns if dados[col].dtype !=
'object']
```

```
categoricas_cols = [col for col in dados.columns if dados[col].dtype ==
'object']

dados[numericas_cols].describe()
dados[categoricas_cols].describe()

Resultado do comando `dados[numericas_cols].describe()`:
-----------------------------------------------------------

           year_of_reference   year_model    avg_price_brl
count         202297.000000  202297.000000     202297.000000
mean            2021.564694    2011.271527      52756.909153
std                0.571903       6.376234      51628.677716
min             2021.000000    2000.000000       6647.000000
25%             2021.000000    2006.000000      22855.000000
50%             2022.000000    2012.000000      38027.000000
75%             2022.000000    2016.000000      64064.000000
max             2023.000000    2023.000000     979358.000000

Resultado do comando `dados[categoricas_cols].describe()`:
-------------------------------------------------------------

       month_of_reference fipe_code authentication   brand
count              202297    202297         202297  202297
unique                 12      2091         202295       6
top               January  003281-6   3r6c277cnqcb    Fiat
freq                24260       425              2   44962

                                        model      fuel    gear engine_size
count                                  202297    202297  202297      202297
unique                                   2112         3       2          29
top     Palio Week. Adv/Adv TRYON 1.8 mpi Flex  Gasoline  manual         1,6
freq                                      425    168685  161885       47420
```

1-E)

```
Python
Código utilizado:
-----------------

dados['model'].value_counts()
dados_contados_marca = dados['brand'].value_counts()
dados_contados_marca

Resultado do comando `dados['model'].value_counts()`:
-----------------------------------------------------
```

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

Resultado do comando `dados_contados_marca`:
--------------------------------------------

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

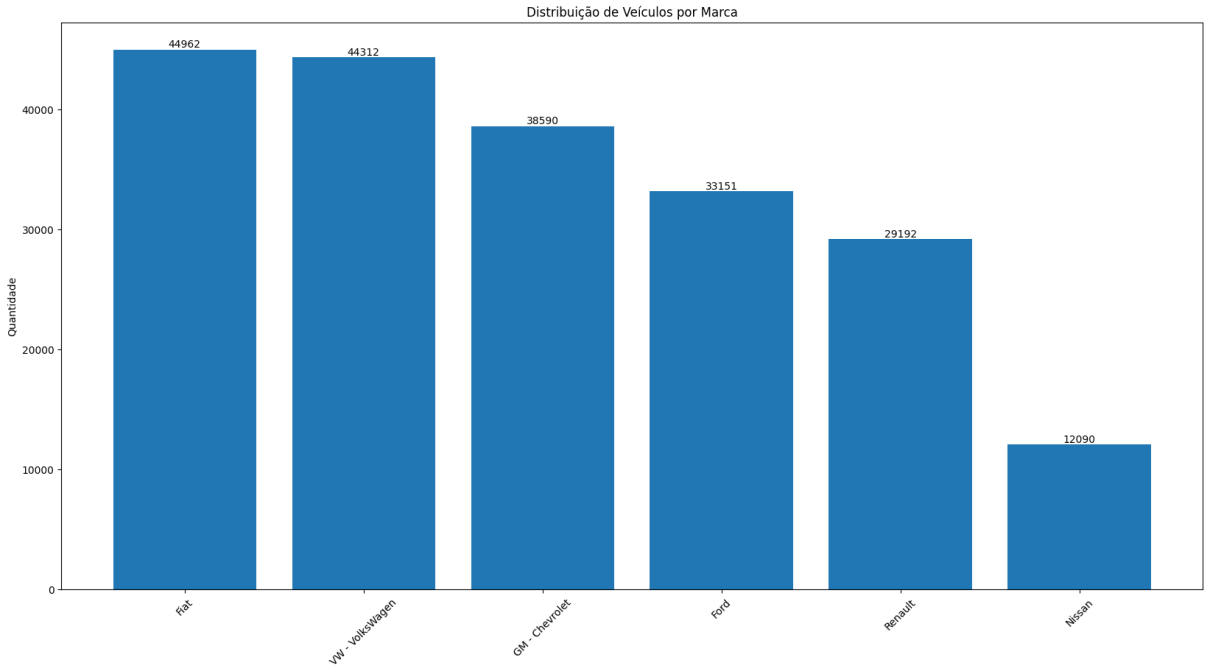
1-F) Após o carregamento do arquivo, verificamos que todas as colunas apresentavam o mesmo número de dados ausentes, indicando linhas completamente vazias, que foram excluídas. Dois registros duplicados foram encontrados, mas mantidos por representarem possíveis ocorrências legítimas. A análise descritiva mostrou 6 marcas diferentes e mais de 2.100 modelos únicos, com Fiat e Volkswagen como as mais recorrentes. Realizamos também a contagem de veículos por marca e modelo, além da separação entre variáveis numéricas e categóricas para análise posterior.

1-G) Todas as colunas estavam faltando a mesma quantidade de dados, prevendo assim que seriam linhas inteiras em branco, excluímos estas linhas. Encontramos 2 dados duplicados, e tendo em vista que se trata de dados que podem coincidir não os eliminamos. Na estatística descritiva verificamos as colunas numéricas e as categóricas, e por fim a contagem dos veículos por marca e modelo.

2-A)

```
Python
Código utilizado:
-----------------

plt.figure(figsize=(20,10))
grafico1 = plt.bar(dados_contados_marca.index, dados_contados_marca.values)
plt.title('Distribuição de Veículos por Marca')
plt.ylabel('Quantidade')
plt.xticks(rotation=45)
plt.bar_label(grafico1, size=10)

Resultado obtido:
-----------------
(Figura 5 – Distribuição de Veículos por Marca)
```

FIGURA 5 – DISTRIBUIÇÃO DE VEÍCULOS POR MARCA



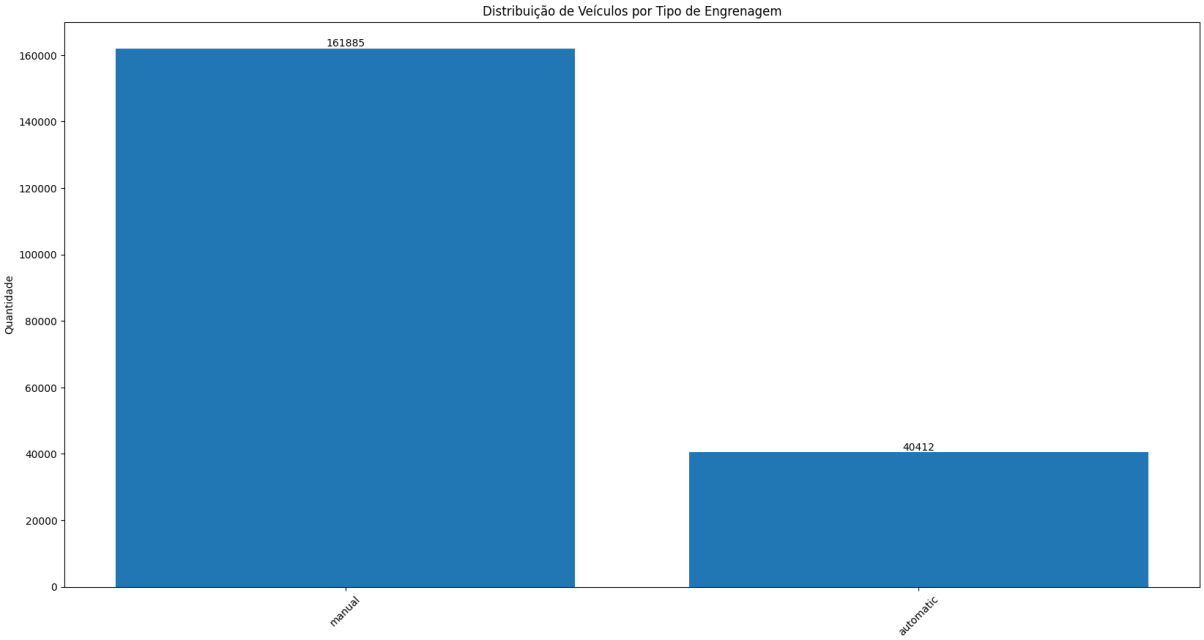
FONTE: O autor (2025)

2-B)

```
Python
Código utilizado:
-----------------

dados_contados_gear = dados['gear'].value_counts()
plt.figure(figsize=(20,10))
```

```
grafico2 = plt.bar(dados_contados_gear.index, dados_contados_gear.values)
plt.title('Distribuição de Veículos por Tipo de Engrenagem')
plt.ylabel('Quantidade')
plt.xticks(rotation=45)
plt.bar_label(grafico2, size=10)

Resultado obtido:
-----------------
(Figura 6 – Distribuição de Veículos por Tipo de Engrenagem)
```

FIGURA 6 – DISTRIBUIÇÃO DE VEÍCULOS POR TIPO DE ENGRENAGEM



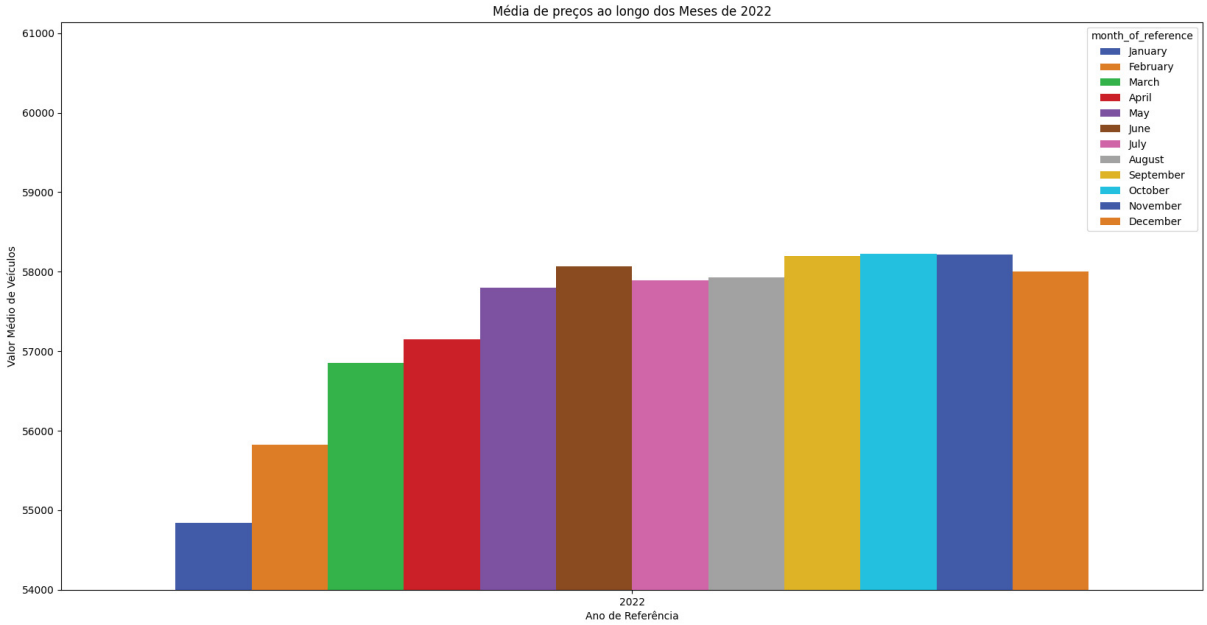
FONTE: O autor (2025)

2-C)

```
Python
Código utilizado:
-----------------

dados['year_of_reference'] = round(dados['year_of_reference']).astype(int)
dados['year_model'] = round(dados['year_model']).astype(int)

media_preco_mensal = dados[dados['year_of_reference'] ==
2022].groupby(['year_of_reference',
'month_of_reference'])['avg_price_brl'].mean().round(0).reset_index()
```

```
plt.figure(figsize=(20,10))
sns.barplot(x='year_of_reference', y='avg_price_brl',
hue='month_of_reference', data=media_preco_mensal, hue_order=['January',
'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September',
'October', 'November', 'December'],
            palette="bright")
plt.title('Média de preços ao longo dos Meses de 2022')
plt.ylabel('Valor Médio de Veículos')
plt.xlabel('Ano de Referência')
plt.ylim(ymin=54000)

Resultado obtido:
-----------------
(Figura 7 - Média de Preços ao Longo dos Meses de 2022)
```

FIGURA 7 – MÉDIA DE PREÇOS AO LONGO DOS MESES DE 2022



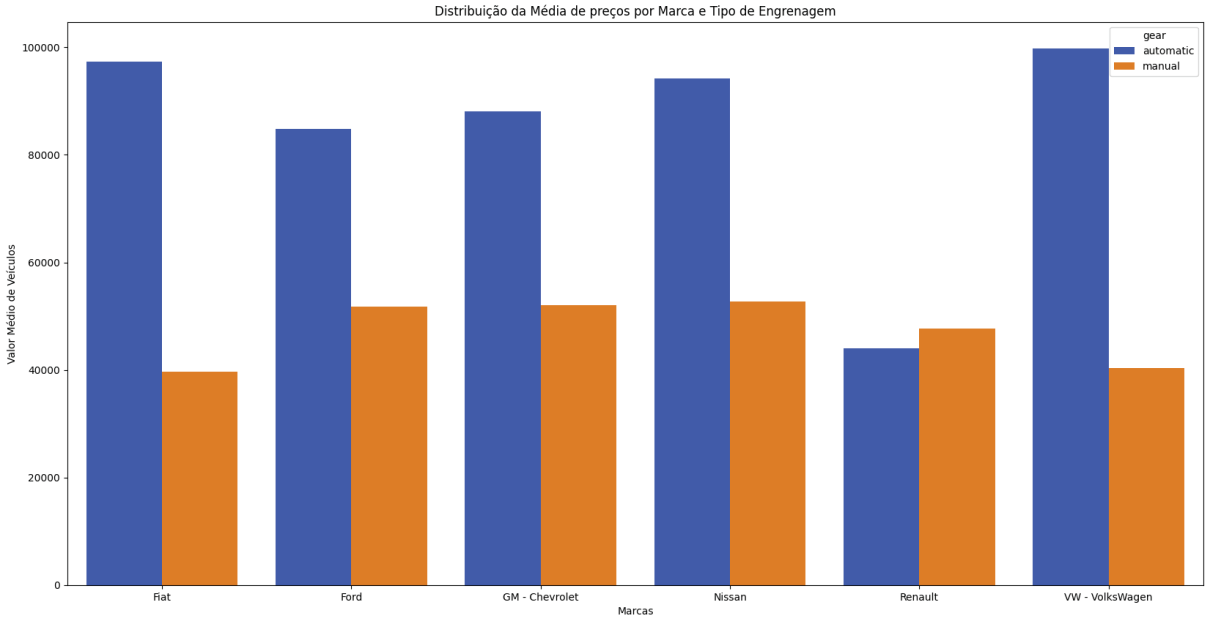
FONTE: O autor (2025)

2-D)

```
Python
Código utilizado:
-----------------
```

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

plt.figure(figsize=(20,10))
sns.barplot(x='brand', y='avg_price_brl', hue='gear',
data=media_preco_marca_engrenagem, palette="bright")
plt.title('Distribuição da Média de preços por Marca e Tipo de Engrenagem')
plt.ylabel('Valor Médio de Veículos')
plt.xlabel('Marcas')

Resultado obtido:
-----------------
(Figura 8 – Média de Preços por Marca e Tipo de Engrenagem)
```

FIGURA 8 – MÉDIA DE PREÇOS POR MARCA E TIPO DE ENGRENAGEM



FONTE: O autor (2025)

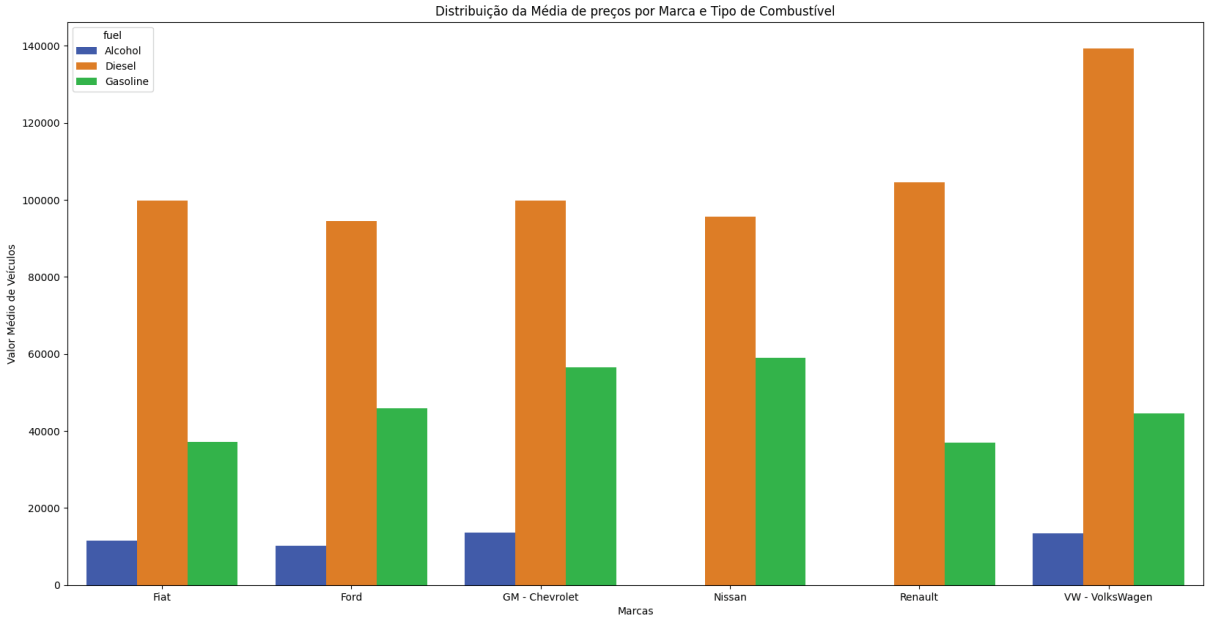
2-E) Percebe-se que em quase todas as marcas os veículos com valores mais altos são automáticos, apenas na Marca Renault o câmbio manual superou os valores dos automáticos.

2-F)

```
Python
Código utilizado:
-----------------
```

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

plt.figure(figsize=(20,10))
sns.barplot(x='brand', y='avg_price_brl', hue='fuel',
data=media_preco_marca_combustivel, palette="bright")
plt.title('Distribuição da Média de preços por Marca e Tipo de Combustível')
plt.ylabel('Valor Médio de Veículos')
plt.xlabel('Marcas')

Resultado obtido:
-----------------
(Figura 9 – Média de Preços por Marca e Tipo de Combustível)
```

FIGURA 9 – MÉDIA DE PREÇOS POR MARCA E TIPO DE COMBUSTÍVEL



FONTE: O autor (2025)

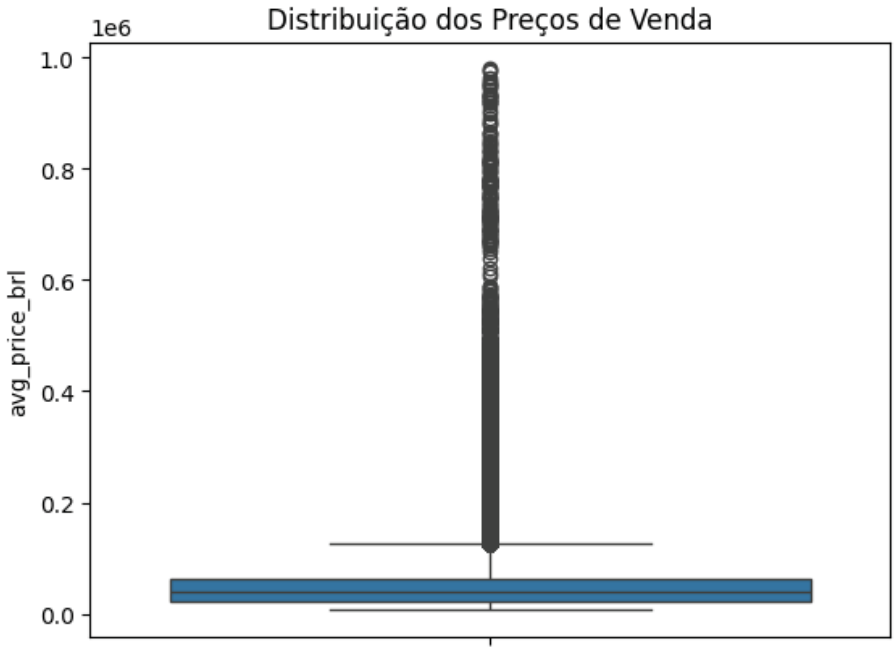
2-G) Observamos que os veículos com valores mais altos são os movidos a Diesel, em segundo a gasolina e por último os à álcool, e que também as marcas Nissan e Renault não foram efetuadas vendas de veículos a álcool ou não tem veículos a álcool.

3-A)

Python

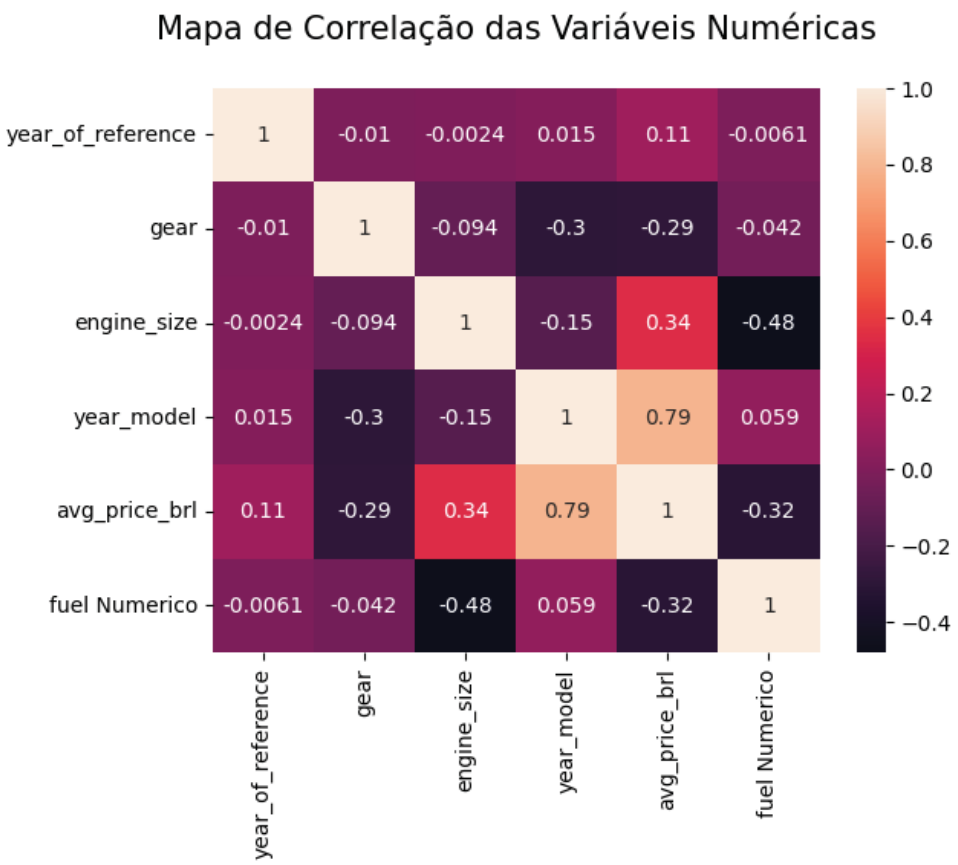
```
Código utilizado:
-----------------

sns.boxplot(dados['avg_price_brl']).set_title("Distribuição dos Preços de
Venda")

dados['engine_size'] = pd.to_numeric(dados['engine_size'], errors='coerce')

def atribuir_valor_numerico(categoria):
    if categoria == 'Alcohol':
        return 1
    elif categoria == 'Diesel':
        return 2
    elif categoria == 'Gasoline':
        return 3
    else:
        return None

dados['fuel Numerico'] = dados['fuel'].apply(atribuir_valor_numerico)
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])
dados_num = dados.drop(['month_of_reference', 'fipe_code', 'authentication',
'brand','model', 'fuel'], axis=1)

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

Resultado obtido:
-----------------
(Figura 10 – Distribuição dos Preços de Venda)
(Figura 11 – Mapa de Correlação das Variáveis Numéricas)
```

FIGURA 10 – DISTRIBUIÇÃO DOS PREÇOS DE VENDA



FONTE: O autor (2025)

FIGURA 11 – MAPA DE CORRELAÇÃO DAS VARIÁVEIS NUMÉRICAS



FONTE: O autor (2025)

3-B)

```
Python
Código utilizado:
-----------------

X = dados_num.drop(['avg_price_brl'], axis=1)
Y = dados_num['avg_price_brl']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=42)
```

3-C)

```
Python
Código utilizado:
-----------------

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

3-D)

```
Python
Código utilizado:
-----------------

valores_preditos_rf = model_rf.predict(X_test)
```

3-E)

```
Python
Código utilizado:
-----------------

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

```
Resultado obtido:
-----------------

                     importance
engine_size          0.495951
year_model           0.429101
fuel Numerico        0.037248
gear                 0.025908
year_of_reference    0.011793
```

3-F) Ao verificar os resultados da importância das variáveis verificamos que a variável engine_size foi a que tem maior importância em relação ao preço de venda, e em seguida o Ano do modelo que havia ficado em primeiro no gráfico de correlação.

3-G)

Python

```
Código utilizado:
-----------------

# Random Forest (simples)
model_rf = RandomForestRegressor()
model_rf.fit(X_train, Y_train)
valores_preditos_rf = model_rf.predict(X_test)

# Random Forest (com parâmetros ajustados)
model_rf_parametros = RandomForestRegressor(
    max_depth=29,
    min_samples_leaf=32,
    min_samples_split=28,
    n_estimators=208,
    random_state=43
)
model_rf_parametros.fit(X_train, Y_train)
valores_preditos_rf_parametros = model_rf_parametros.predict(X_test)

# XGBoost
model_xgboost = XGBRegressor()
model_xgboost.fit(X_train, Y_train)
valores_preditos_xgboost = model_xgboost.predict(X_test)
```

```
# Avaliação dos modelos
mse_rf = mean_squared_error(Y_test, valores_preditos_rf)
mae_rf = mean_absolute_error(Y_test, valores_preditos_rf)
r2_rf = r2_score(Y_test, valores_preditos_rf)

mse_rf_param = mean_squared_error(Y_test, valores_preditos_rf_parametros)
mae_rf_param = mean_absolute_error(Y_test, valores_preditos_rf_parametros)
r2_rf_param = r2_score(Y_test, valores_preditos_rf_parametros)

mse_xgb = mean_squared_error(Y_test, valores_preditos_xgboost)
mae_xgb = mean_absolute_error(Y_test, valores_preditos_xgboost)
r2_xgb = r2_score(Y_test, valores_preditos_xgboost)

# Importâncias
feature_importances_rf = pd.DataFrame(model_rf.feature_importances_,
index=X_train.columns, columns=['importance']).sort_values('importance',
ascending=False)
feature_importances_rf_param =
pd.DataFrame(model_rf_parametros.feature_importances_, index=X_train.columns,
columns=['importance']).sort_values('importance', ascending=False)
feature_importances_xgb = pd.DataFrame(model_xgboost.feature_importances_,
index=X_train.columns, columns=['importance']).sort_values('importance',
ascending=False)

Resultado obtido:
-----------------

### Random Forest (simples):
- MSE: 165040937.5071639
- MAE: 7001.758924115653
- R²: 0.9377891408524998

Importância das variáveis:
engine_size          0.495951
year_model           0.429101
fuel Numerico        0.037248
gear                 0.025908
year_of_reference    0.011793

---

### Random Forest (com parâmetros ajustados):
```

```
- R²: 0.904087447509726

Importância das variáveis:
engine_size          0.495951
year_model           0.429101
fuel Numerico        0.037248
gear                 0.025908
year_of_reference    0.011793

---

### XGBoost:
- R²: 0.9379208835406198

Importância das variáveis:
engine_size          0.366504
year_model           0.301597
fuel Numerico        0.157089
gear                 0.155039
year_of_reference    0.019771
```

Portanto, embora os resultados tenham sido bastante próximos entre os modelos testados, o XGBoost apresentou o melhor desempenho, com uma acurácia (R^2) de 0,9379, sendo considerado o modelo mais adequado para este conjunto de dados.

3-H) O modelo com melhor desempenho, apesar de os resultados estarem muito próximos, foi o XGBoost, que obteve um coeficiente de determinação (R^2) de 0,9379. Embora o XGBoost tenha atribuído pesos diferentes às variáveis em relação aos demais modelos, ainda assim apresentou os melhores resultados.

Para avaliação dos modelos, foram utilizadas as seguintes métricas:

- **MSE** (Mean Squared Error)
- **MAE** (Mean Absolute Error)
- **R^2** (Coeficiente de Determinação)

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

$$\textbf{Volume} = b0 + b1 * dap^2 * Ht$$

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: Volume = b0 + b1 * dap^2 * Ht

      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

FIGURA 12 – FÓRMULA DE R^2

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

FONTE: O autor (2025)

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}

FIGURA 13 – FÓRMULA DE S_{YX}

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

FONTE: O autor (2025)

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

- Syx%

FIGURA 14 – FÓRMULA DE $SYX\%_X$

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

FONTE: O autor (2025)

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1-1)

```
None
Código utilizado:
```

```
-----------------

data(Satellite)
dados_satelite <- as.data.frame(Satellite)[c("x.17", "x.18", "x.19", "x.20",
"classes")]
str(dados_satelite)
```

1-2)

```
None
Código utilizado:
-----------------

indices_satelite <- createDataPartition(dados_satelite$classes, p=0.80,
list=FALSE)
satelite_treino <- dados_satelite[indices_satelite, ]
satelite_teste  <- dados_satelite[-indices_satelite, ]
```

1-3)

```
None
Código utilizado:
-----------------

# RNA
model_nnet <- train(classes ~., data=satelite_treino, method="nnet",
trace=FALSE, linout=TRUE)
predictions_nnet <- predict(model_nnet, satelite_teste)

# Random Forest
model_rf <- train(classes ~., data=satelite_treino, method="rf")
predictions_rf <- predict(model_rf, satelite_teste)

# SVM
model_svm <- train(classes ~., data=satelite_treino, method="svmRadial")
predictions_svm <- predict(model_svm, satelite_teste)
```

1-4)

```
None
Código utilizado:
-----------------

# Matriz de confusão do modelo nnet
confusionMatrix(predictions_nnet, satelite_teste$classes)

# Matriz de confusão do modelo Random Forest
confusionMatrix(predictions_rf, satelite_teste$classes)

# Matriz de confusão do modelo SVM
confusionMatrix(predictions_svm, satelite_teste$classes)

Resultado obtido:
-----------------

Modelo RNA (nnet):
- Acurácia: 0.7220
- Kappa: 0.6506
- Balanced Accuracy (média): ~0.79

Modelo Random Forest:
- Acurácia: 0.8466
- Kappa: 0.8101
- Balanced Accuracy (média): ~0.88

Modelo SVM:
- Acurácia: 0.8637
- Kappa: 0.8308
- Balanced Accuracy (média): ~0.90
```

1-5) Com base na métrica de acurácia, o modelo SVM foi o que obteve o melhor desempenho, com 86,37% de acerto no conjunto de testes.

2-1)

```
None
Código utilizado:
-----------------

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

2-2)

```
None
Código utilizado:
-----------------

dados_volumes$NR <- NULL
```

2-3)

```
None
Código utilizado:
-----------------

indices_vol_treino <- createDataPartition(dados_volumes$VOL, p=0.80,
list=FALSE)
dados_volumes_treino <- dados_volumes[indices_vol_treino, ]
dados_volumes_teste  <- dados_volumes[-indices_vol_treino, ]
```

2-4)

```
None
Código utilizado:
-----------------

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

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

2-5)

```
None
Código utilizado:
-----------------

predicao_rf_volume     <- predict(model_rf_volume, dados_volumes_teste)
predicao_svm_volume    <- predict(model_svm_volume, dados_volumes_teste)
predicao_rn_volume     <- predict(model_rn_volume, dados_volumes_teste)
predicao_spurr_volume  <- predict(model_spurr_volume, dados_volumes_teste)
```

2-6)

```
None
Código utilizado:
-----------------

Syx <- function (Y_real, Y_pred) {
    sqrt(sum((Y_real - Y_pred)^2) / (length(Y_real) - 2))
}

SyxPerc <- function (Y_real, Y_pred) {
    Syx(Y_real, Y_pred) / mean(Y_real) * 100
}

R2 <- function (Y_real, Y_pred) {
    1 - sum((Y_real - Y_pred)^2) / sum((Y_real - mean(Y_real))^2)
}

rmse_rf    <- RMSE(predicao_rf_volume, dados_volumes_teste$VOL)
rmse_rn    <- RMSE(predicao_rn_volume, dados_volumes_teste$VOL)
rmse_svm   <- RMSE(predicao_svm_volume, dados_volumes_teste$VOL)
rmse_spurr <- RMSE(predicao_spurr_volume, dados_volumes_teste$VOL)

r2_rf    <- R2(predicao_rf_volume, dados_volumes_teste$VOL)
r2_rn    <- R2(predicao_rn_volume, dados_volumes_teste$VOL)
r2_svm   <- R2(predicao_svm_volume, dados_volumes_teste$VOL)
r2_spurr <- R2(predicao_spurr_volume, dados_volumes_teste$VOL)

syx_rf    <- Syx(predicao_rf_volume, dados_volumes_teste$VOL)
syx_rn    <- Syx(predicao_rn_volume, dados_volumes_teste$VOL)
syx_svm   <- Syx(predicao_svm_volume, dados_volumes_teste$VOL)
syx_spurr <- Syx(predicao_spurr_volume, dados_volumes_teste$VOL)
```

```
syx_perc_rf    <- SyxPerc(predicao_rf_volume, dados_volumes_teste$VOL)
syx_perc_rn    <- SyxPerc(predicao_rn_volume, dados_volumes_teste$VOL)
syx_perc_svm   <- SyxPerc(predicao_svm_volume, dados_volumes_teste$VOL)
syx_perc_spurr <- SyxPerc(predicao_spurr_volume, dados_volumes_teste$VOL)

Resultado obtido:
-----------------

Métricas de avaliação dos modelos aplicados ao conjunto de teste:

Modelo           | RMSE     |     R²     |   Syx    | Syx (%)
-----------------|----------|------------|----------|---------
Random Forest    | 0.1371   | 0.8500     | 0.1446   | 10.96%
Rede Neural (RN) | 0.1243   | 0.8624     | 0.1310   | 10.00%
SVM              | 0.1395   | 0.8187     | 0.1470   | 11.00%
Spurr            | 0.1494   | 0.8503     | 0.1574   | 11.83%
```

2-7) Considerando todas as métricas calculadas (RMSE, R^2, Syx e Syx percentual), o modelo que apresentou o melhor desempenho geral foi a Rede Neural Artificial (RN). No entanto, o modelo Spurr também demonstrou resultados consistentes, com desempenho competitivo frente aos métodos estatísticos e de machine learning.

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

(15 pontos) Elaborar a tabela de frequencias 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

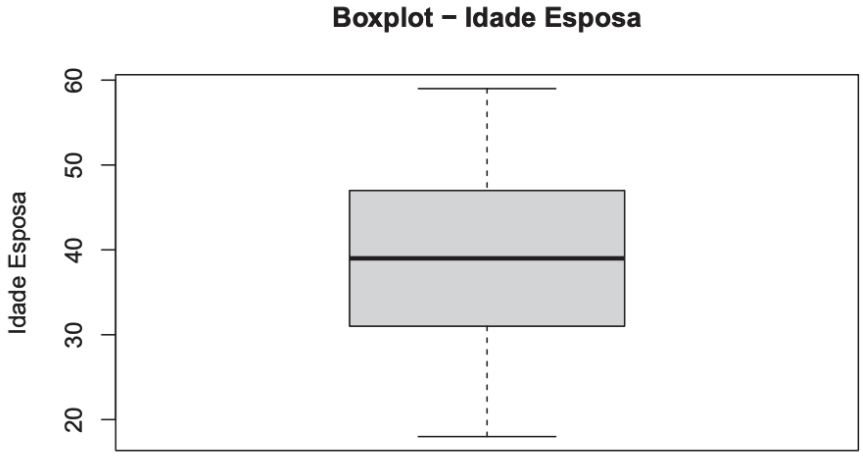
Apresentar a resolução (somente o resultado) das questões do trabalho.

1-A)

```
None
Código utilizado:
-----------------
```

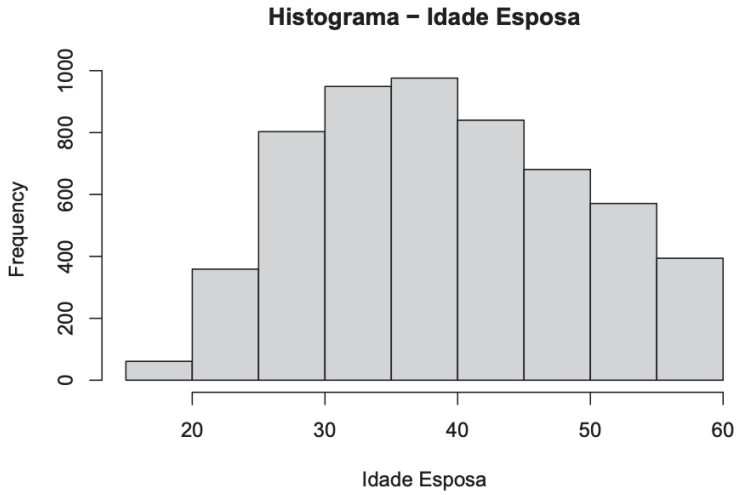
```
boxplot(salarios$age, main="Boxplot - Idade Esposa", ylab="Idade Esposa")
hist(salarios$age, main="Histograma - Idade Esposa", xlab="Idade Esposa")
boxplot(salarios$husage, main="Boxplot - Idade do Cônjuge", ylab="Idade
Marido")
hist(salarios$husage, main="Histograma - Idade do Cônjuge", xlab="Idade
Marido", ylab="Frequência")

plot(density(salarios$age),
     main="Comparação da Idade Esposa e Marido",
     xlim=c(0, max(salarios$age, salarios$husage)),
     ylim=c(0, 0.05),
     xlab="Idade",
     ylab="Densidade",
     col="red",
     lwd=2)
lines(density(salarios$husage), col="blue", lwd=2)
legend("topright", legend=c("Idade Esposa", "Idade Marido"), col=c("red",
"blue"), lwd=2)

Resultado obtido:
-----------------
  (Figura 15 – Boxplot da Idade da Esposa)
  (Figura 16 – Histograma da Idade da Esposa)
  (Figura 17 – Boxplot da Idade do Cônjuge)
  (Figura 18 – Histograma da Idade do Cônjuge)
  (Figura 19 – Comparação das Densidades de Idade entre Esposa e Marido)
```

Figura 15 – Boxplot da Idade da Esposa



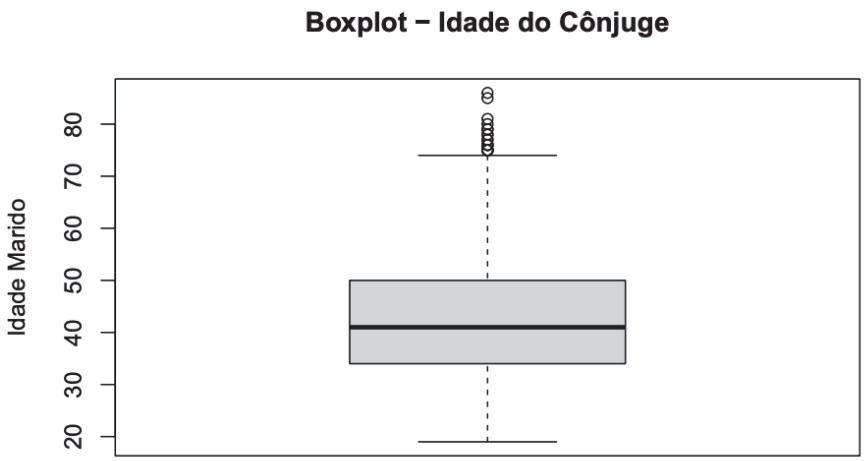
FONTE: O autor (2025)

FIGURA 16 – HISTOGRAMA DA IDADE DA ESPOSA



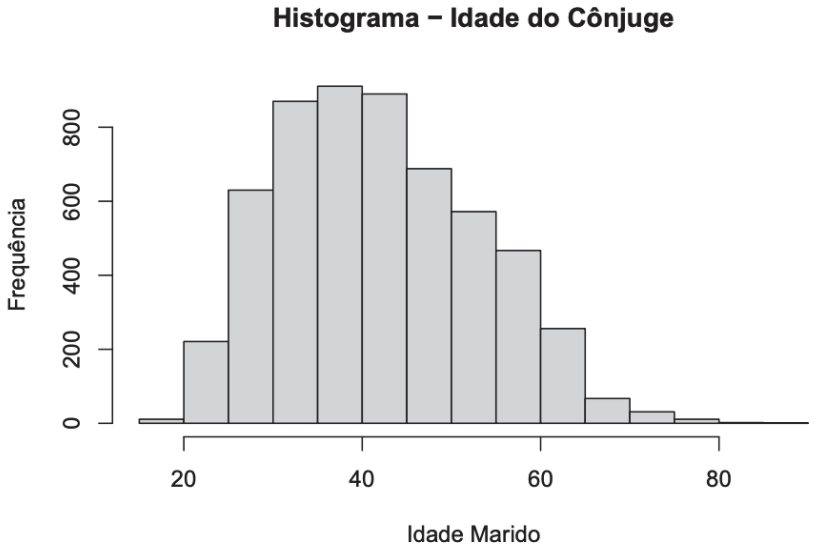
FONTE: O autor (2025)

FIGURA 17 – BOXPLOT DA IDADE DO CÔNJUGE



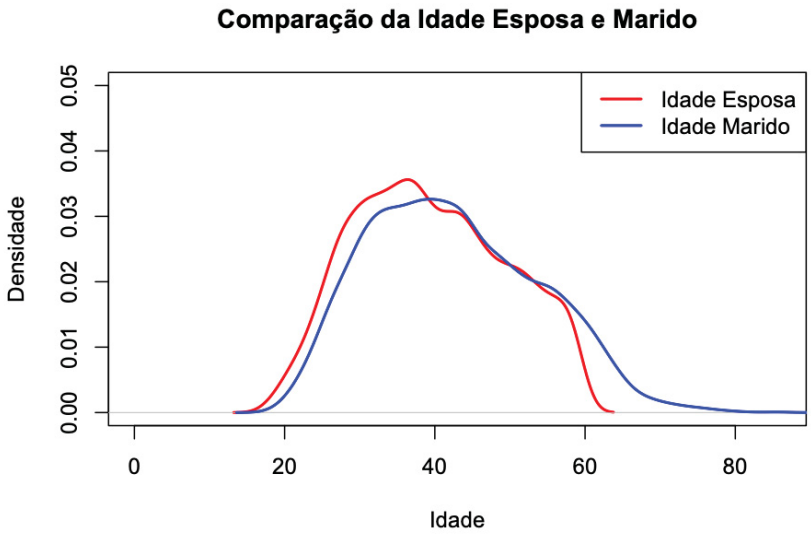
FONTE: O autor (2025)

FIGURA 18 – HISTOGRAMA DA IDADE DO CÔNJUGE



FONTE: O autor (2025)

FIGURA 19 – COMPARAÇÃO DAS DENSIDADES DE IDADE ENTRE ESPOSA E MARIDO



FONTE: O autor (2025)

1-B)

```
None
Código utilizado:
-----------------

tabela_idade_esposa <- fdt(salarios$age)
tabela_idade_marido <- fdt(salarios$husage)
```

2-A)

```
None
Código utilizado:
-----------------

mean(salarios$age)
mean(salarios$husage)

median(salarios$age)
median(salarios$husage)

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

```
Resultado obtido:
-----------------

- Média da idade das esposas: 39,43 anos
- Média da idade dos maridos: 42,45 anos
   A média dos maridos é aproximadamente 7,67% maior

- Mediana da idade das esposas: 39 anos
- Mediana da idade dos maridos: 41 anos
   A mediana dos maridos é aproximadamente 5,13% maior

- Moda da idade das esposas: 37 anos (217 ocorrências)
- Moda da idade dos maridos: 44 anos (201 ocorrências)
   A moda dos maridos é aproximadamente 18,92% maior que a das esposas
```

2-B)

```
None
Código utilizado:
-----------------

var(salarios$age)
var(salarios$husage)
((126.07 / 99.75) - 1) * 100

sd(salarios$age)
sd(salarios$husage)
((11.23 / 9.99) - 1) * 100

media_esp <- mean(salarios$age)
media_mar <- mean(salarios$husage)

sdE <- sd(salarios$age)
sdM <- sd(salarios$husage)

cvE <- (sdE / media_esp) * 100
cvM <- (sdM / media_mar) * 100

Resultado obtido:
-----------------

- Variância da idade das esposas: 99,75
```

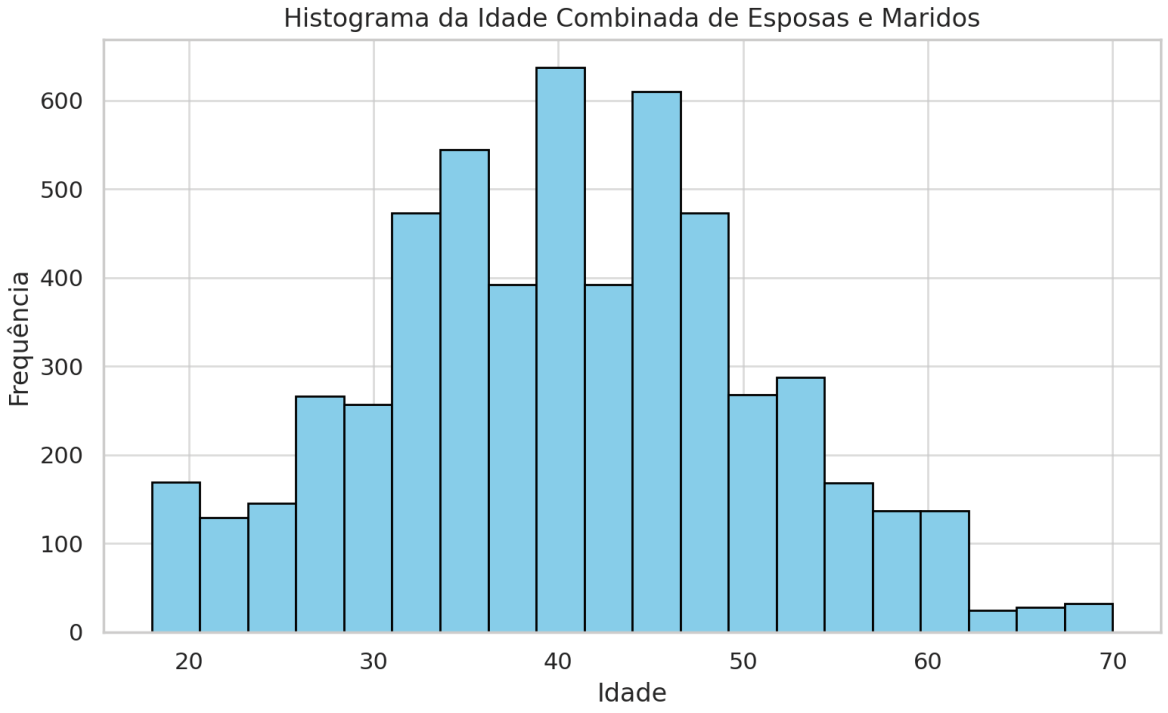
```
- Variância da idade dos maridos: 126,07
   A variância da idade dos maridos é aproximadamente 26,38% maior

- Desvio padrão da idade das esposas: 9,99
- Desvio padrão da idade dos maridos: 11,23
   O desvio padrão da idade dos maridos é aproximadamente 12,41% maior

- Coeficiente de variação da idade das esposas: 25,33%
- Coeficiente de variação da idade dos maridos: 26,45%
   Ambos os grupos apresentam dispersão moderada (entre 15% e 30%)
   A idade dos maridos apresenta ligeiramente maior variabilidade
```

3-A)

```
None
Código utilizado:
-----------------

idade_emp <- data.frame(
  grupo = rep(c("Esposa", "Marido"), each = nrow(salarios)),
  idade = c(salarios$age, salarios$husage)
)

mean(idade_emp$idade)
sd(idade_emp$idade)

group_by(idade_emp, grupo) %>%
  summarise(
    count = n(),
    mean = mean(idade, na.rm = TRUE),
    sd = sd(idade, na.rm = TRUE)
  )

hist(idade_emp$idade)

plotNormalHistogram(idade_emp$idade, prob = FALSE,
                    main = "Normal Distribuition overlay on Histrogram",
                    length = 1000)

lillie.test(idade_emp$idade)

ggboxplot(idade_emp, x = "grupo", y = "idade", ggtheme = theme_minimal())
ggboxplot(idade_emp, x = "grupo", y = "idade", ggtheme = theme_minimal(), add
= "jitter")

group_by(idade_emp, grupo) %>%
  summarise(
    count = n(),
    median = median(idade, na.rm = TRUE),
```

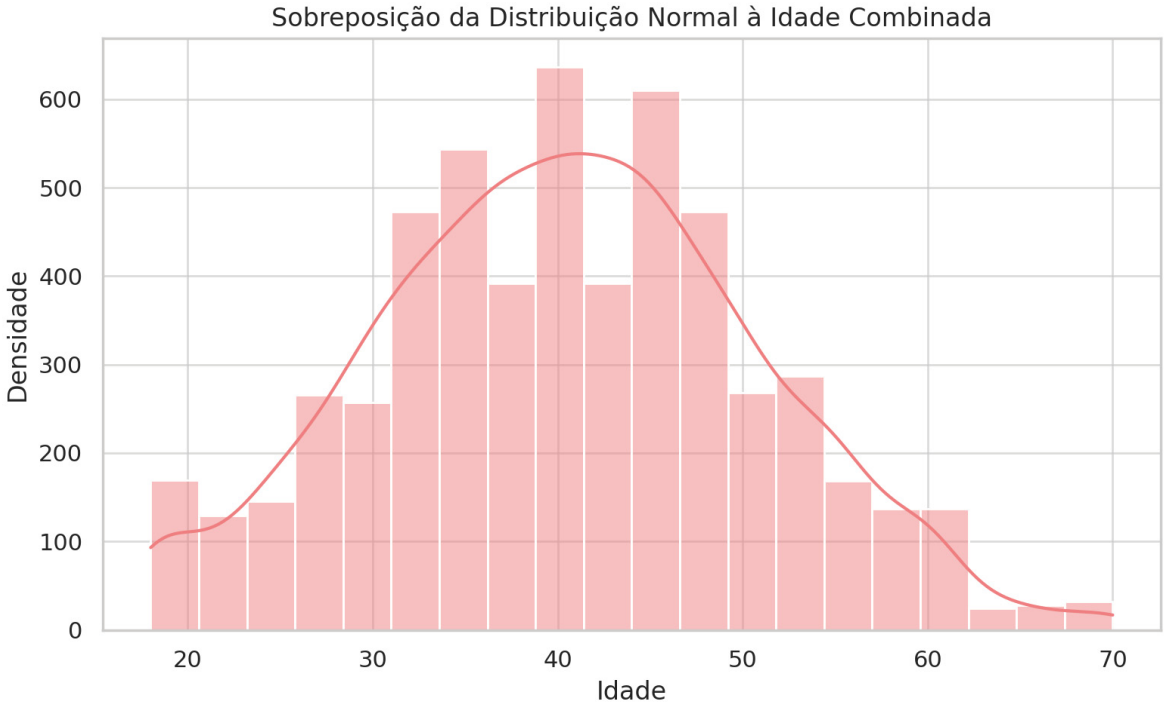
```
    IQR = IQR(idade, na.rm = TRUE)
  )

ggboxplot(idade_emp, x = "grupo", y = "idade",
          color = "grupo", palette = c("#00AFbB", "#E7B800"),
          ylab = "idade", xlab = "Grupo")

# Testes de Wilcoxon
wilcox.test(idade ~ grupo, data = idade_emp, exact = FALSE, conf.int = TRUE)
wilcox.test(idade ~ grupo, data = idade_emp, exact = FALSE, alternative =
"less", conf.int = TRUE)
wilcox.test(idade ~ grupo, data = idade_emp, exact = FALSE, alternative =
"greater", conf.int = TRUE)


Resultado obtido:
-----------------

- A média geral das idades foi aproximadamente 40,94 anos e o desvio padrão
10,70
- O teste de Lilliefors indicou que a distribuição das idades não é normal (p
< 0.05)
   Por isso, foi utilizado o teste não paramétrico de Wilcoxon

Teste 1 – Hipótese de igualdade:
- H₀: A idade mediana dos maridos é igual à das esposas
- Resultado: p < 0.0001 → rejeita-se H₀
- Conclusão: A idade mediana dos maridos não é estatisticamente igual à das
esposas

Teste 2 – Hipótese de que os maridos são mais jovens:
- H₀: A idade mediana dos maridos é menor que a das esposas
- Resultado: p < 0.0001 → rejeita-se H₀
- Conclusão: A idade mediana dos maridos não é menor

Teste 3 – Hipótese de que os maridos são mais velhos:
- H₀: A idade mediana dos maridos é maior que a das esposas
- Resultado: p = 1.0 → não rejeita-se H₀
- Conclusão: A idade mediana dos maridos é maior que a das esposas
```

FIGURA 20 – HISTOGRAMA DA IDADE COMBINADA DE ESPOSAS E MARIDOS



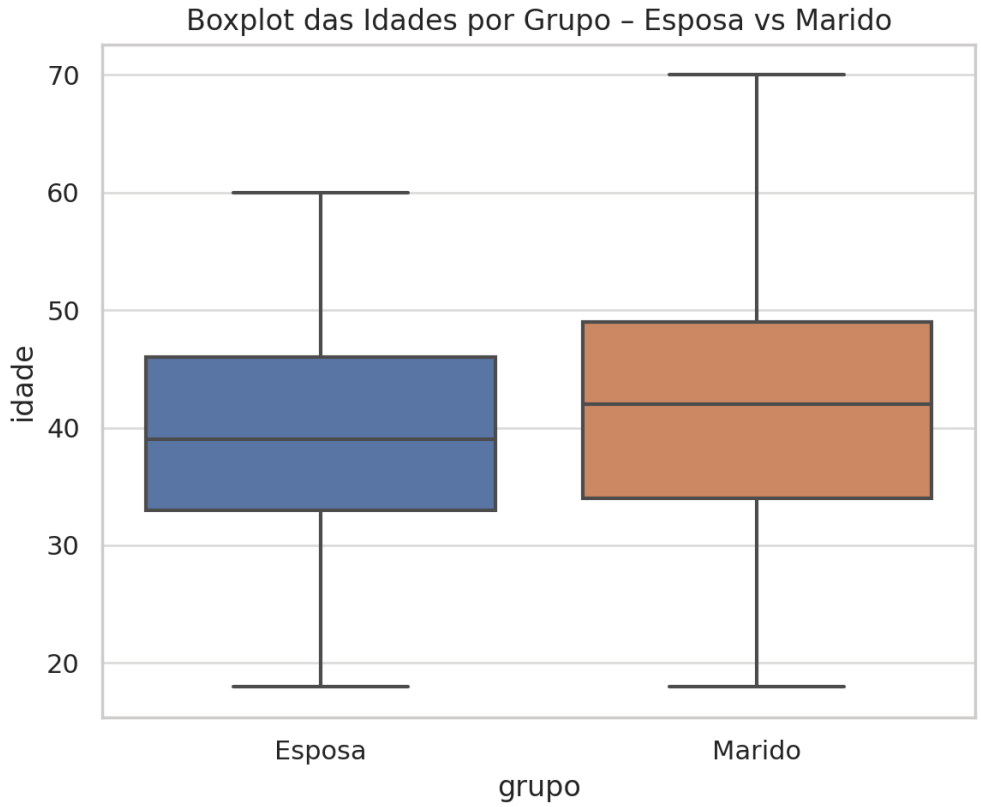
FONTE: O autor (2025)

FIGURA 21 – SOBREPOSIÇÃO DA DISTRIBUIÇÃO NORMAL À IDADE COMBINADA



FONTE: O autor (2025)

FIGURA 22 – BOXPLOT DAS IDADES POR GRUPO – ESPOSA VS MARIDO



FONTE: O autor (2025)

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

```
None
Código utilizado:
-----------------


library(glmnet)
library(caret)
library(tidyverse)
```

```
load("trabalhosalarios.RData")
dados <- trabalhosalarios %>% select(-earns)
dados$lwage <- exp(dados$lwage)

set.seed(123)
index <- sample(1:nrow(dados), 0.8 * nrow(dados))
train <- dados[index, ]
test <- dados[-index, ]

# Padronização
cols <-
c('husage','husearns','huseduc','hushrs','age','educ','exper','lwage')
preproc <- preProcess(train[, cols], method = c("center", "scale"))
train[, cols] <- predict(preproc, train[, cols])
test[, cols] <- predict(preproc, test[, cols])

# Matrizes
dummies <- dummyVars(lwage ~ ., data = dados)
x_train <- as.matrix(predict(dummies, newdata = train))
y_train <- train$lwage
x_test <- as.matrix(predict(dummies, newdata = test))
y_test <- test$lwage

# Função de avaliação
eval_metrics <- function(true, pred) {
  rmse <- sqrt(mean((true - pred)^2))
  r2 <- 1 - sum((true - pred)^2) / sum((true - mean(true))^2)
  return(c(RMSE = rmse, R2 = r2))
}

# Ridge
ridge_cv <- cv.glmnet(x_train, y_train, alpha = 0)
ridge <- glmnet(x_train, y_train, alpha = 0, lambda = ridge_cv$lambda.min)
pred_ridge <- predict(ridge, newx = x_test)
metrics_ridge <- eval_metrics(y_test, pred_ridge)

# Lasso
lasso_cv <- cv.glmnet(x_train, y_train, alpha = 1)
lasso <- glmnet(x_train, y_train, alpha = 1, lambda = lasso_cv$lambda.min)
pred_lasso <- predict(lasso, newx = x_test)
metrics_lasso <- eval_metrics(y_test, pred_lasso)
```

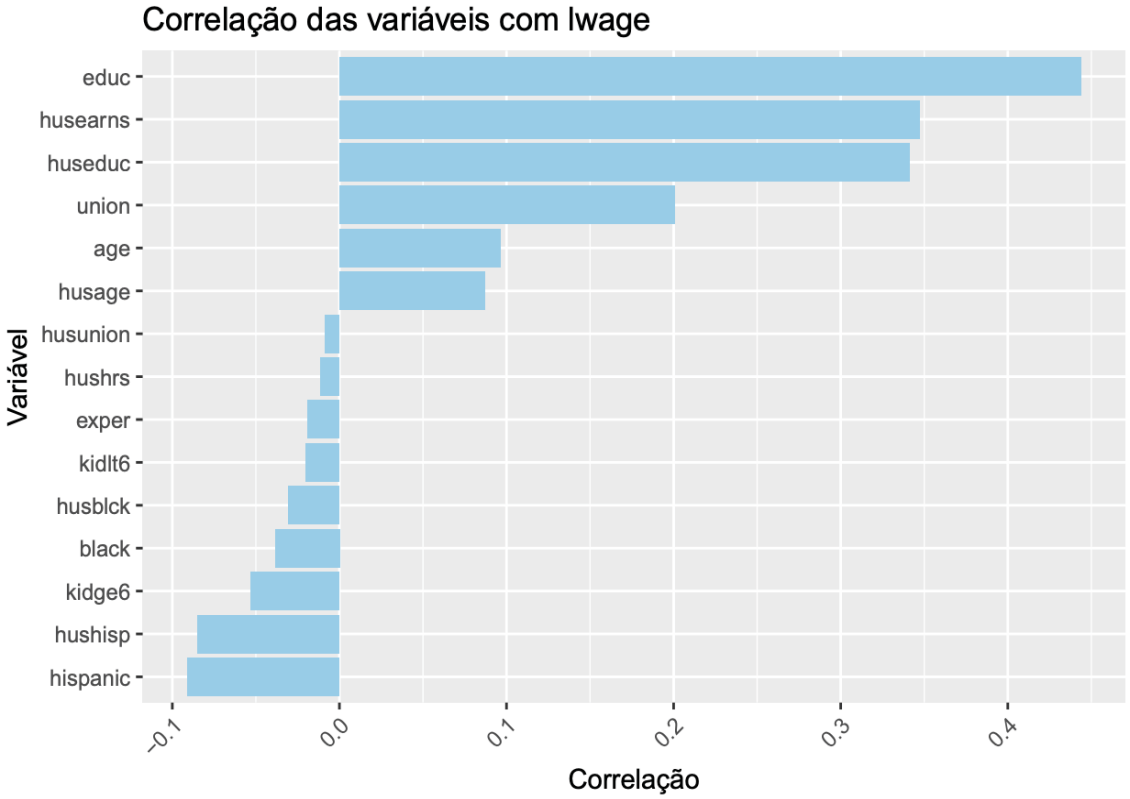
```
# Elastic Net
ctrl <- trainControl("cv", number = 5)
elastic <- train(
  x = x_train, y = y_train, method = "glmnet",
  trControl = ctrl, tuneLength = 10
)
pred_elastic <- predict(elastic, x_test)
metrics_elastic <- eval_metrics(y_test, pred_elastic)

# Resultados finais
resultados <- data.frame(
  Modelo = c("Ridge", "Lasso", "Elastic Net"),
  RMSE = c(metrics_ridge["RMSE"], metrics_lasso["RMSE"],
metrics_elastic["RMSE"]),
  R2 = c(metrics_ridge["R2"], metrics_lasso["R2"], metrics_elastic["R2"])
)
```

Resultado obtido:

```
# | Modelo       | RMSE     | R²       |
# |--------------|----------|----------|
# | Ridge        | 0.0011   | 0.9999   |
# | Lasso        | 0.8486   | 0.2795   |
# | Elastic Net  | 0.8475   | 0.2814   |

# As métricas demonstram que a Regressão Ridge teve desempenho superior, com
menor erro e maior poder explicativo.
```

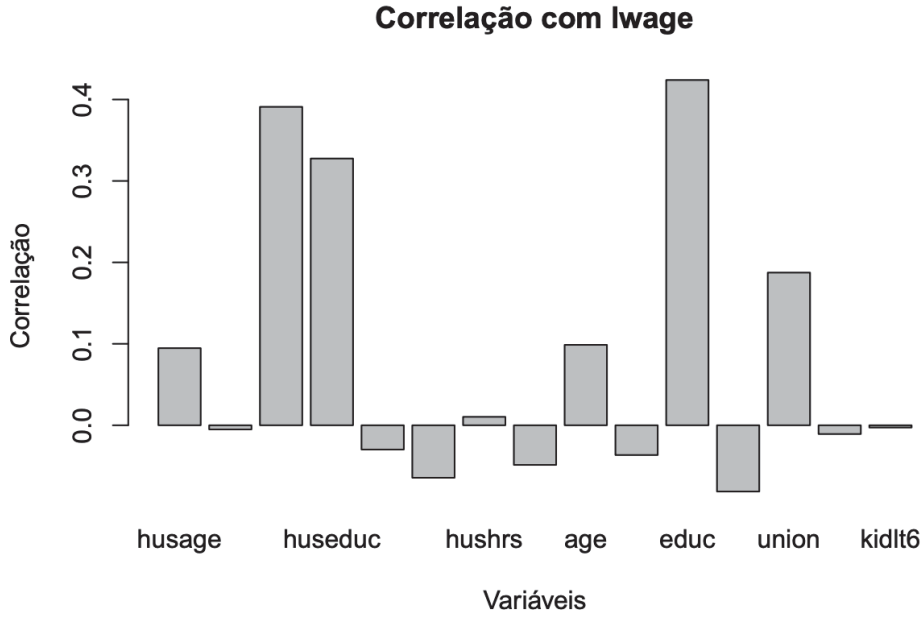
(Figura 23 – Correlação das variáveis com lwage)
(Figura 24 – Valores Reais x Preditos – Ridge)
(Figura 25 – Comparação dos RMSE entre modelos)

FIGURA 23 – CORRELAÇÃO DAS VARIÁVEIS COM LWAGE



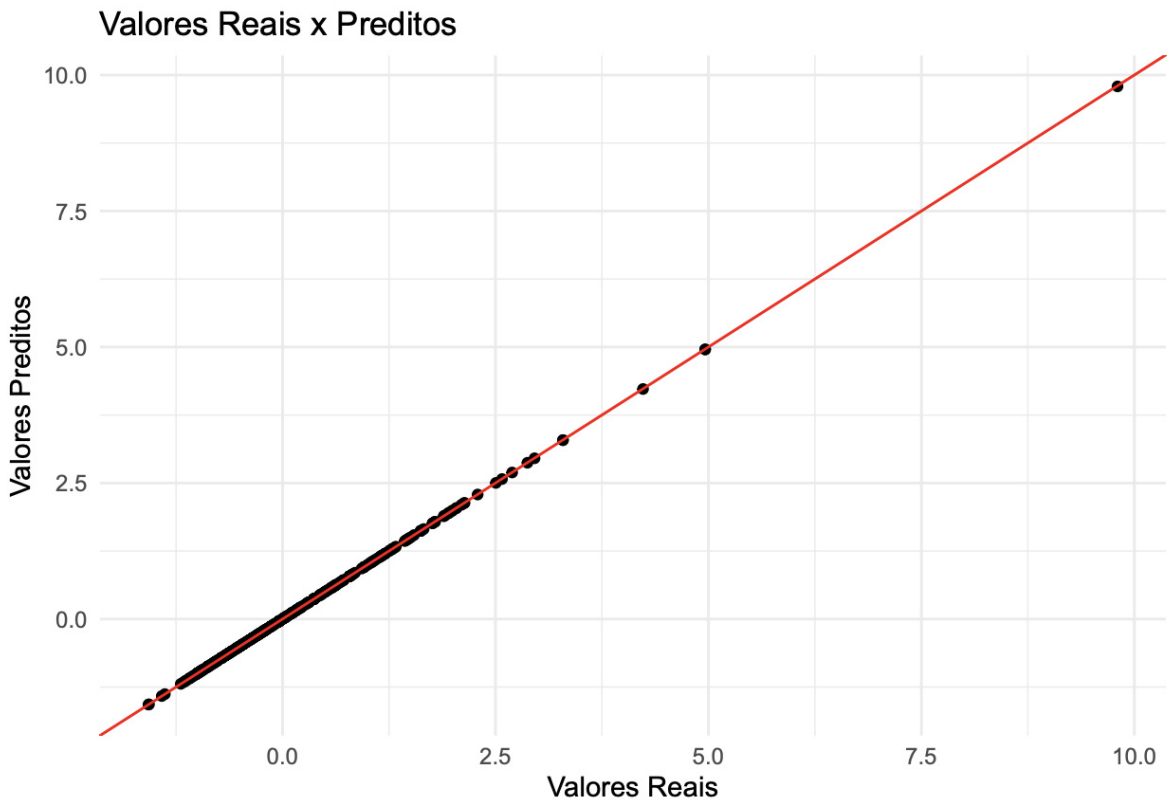
FONTE: O autor (2025)

FIGURA 24 – COMPARAÇÃO DOS RMSE ENTRE MODELOS



Fonte: O autor (2025)

FIGURA 25 – VALORES REAIS X PREDITOS – RIDGE



FONTE: O autor (2025)

APÊNDICE 6 – ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

1-

Python

```
Código utilizado:
-----------------

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."
]
```

```
pre_padroes = [(f, 'inglês') for f in ingles] + [(f, 'espanhol') for f in
espanhol] + [(f, 'português') for f in portugues]

import pandas as pd
import numpy as np
import re
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix, classification_report

def tamanhoMedioFrases(texto):
    palavras = re.findall(r'\b\w+\b', texto)
    tamanhos = [len(p) for p in palavras]
    return (sum(tamanhos) / len(tamanhos)) if tamanhos else 0

def contaCaracteristicas(texto):
    texto = texto.lower()
    vogais = "eiouáéíóúãõâêîôûàèìòùäëïöü"
    consoantes = "bcdfghjklmnpqrstvwxyz"
    letras_espanhol = '¿¡!iñü'
    letras_portugues = 'çáãêõû'
    letras_ingles = "wkyh'"
    num_vogais = sum(1 for c in texto if c in vogais)
    num_consoantes = sum(1 for c in texto if c in consoantes)
    num_letras_espanhol = sum(1 for c in texto if c in letras_espanhol)
    num_letras_portugues = sum(1 for c in texto if c in letras_portugues)
    num_letras_ingles = sum(1 for c in texto if c in letras_ingles)
    palavras = re.findall(r'\b\w+\b', texto)
    palavras_comuns_ingles = {"the", "be", "to", "of", "and", "a", "in"}
    palavras_comuns_espanhol = {"de", "la", "que", "el", "en", "y", "a"}
    palavras_comuns_portugues = {"de", "do", "em", "que", "um", "para", "é"}
    num_palavras_ingles = sum(1 for p in palavras if p in
palavras_comuns_ingles)
    num_palavras_espanhol = sum(1 for p in palavras if p in
palavras_comuns_espanhol)
    num_palavras_portugues = sum(1 for p in palavras if p in
palavras_comuns_portugues)
    return num_vogais, num_consoantes, num_letras_espanhol,
num_letras_portugues, num_letras_ingles, num_palavras_ingles,
num_palavras_espanhol, num_palavras_portugues

def extraiCaracteristicas(frase):
    texto = re.sub(r'[^\w+]', ' ', frase[0])
```

```
    return [tamanhoMedioFrases(texto), *contaCaracteristicas(texto),
frase[1]]

padroes = [extraiCaracteristicas(f) for f in pre_padroes]
dados = pd.DataFrame(padroes)
dados[0] = np.where(dados[9] == 'português', ((dados[0]) + 1) / 2, dados[0])

vet = np.array(dados)
X = vet[:, 0:-1]
y = vet[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
stratify=y, random_state=70)

modelo = svm.SVC()
modelo.fit(X_train, y_train)


acuracia_treino = modelo.score(X_train, y_train)
y_pred_train = modelo.predict(X_train)

acuracia_teste = modelo.score(X_test, y_test)
y_pred_test = modelo.predict(X_test)

texto = "celebrating the birthday of a person"
caracteristicas = extraiCaracteristicas([texto, '0'])[:-1]
idioma_predito = modelo.predict([caracteristicas])

 Resultado obtido:
 -----------------

Acurácia nos dados de treinamento: 86.96%

[[19  0  3]
 [ 2 21  0]
 [ 4  0 20]]
              precision    recall  f1-score   support

    espanhol       0.76      0.86      0.81        22
      inglês       1.00      0.91      0.95        23
   português       0.87      0.83      0.85        24

    accuracy                           0.87        69
   macro avg       0.88      0.87      0.87        69
```

```
weighted avg       0.88      0.87      0.87        69

Acurácia nos dados de teste: 86.96%

[[6 0 2]
 [0 6 1]
 [0 0 8]]
              precision    recall  f1-score   support

    espanhol       1.00      0.75      0.86         8
      inglês       1.00      0.86      0.92         7
   português       0.73      1.00      0.84         8

    accuracy                           0.87        23
   macro avg       0.91      0.87      0.87        23
weighted avg       0.91      0.87      0.87        23

Frase testada: celebrating the birthday of a person
O texto foi classificado como: inglês
```

2-

Python

```
Código utilizado:
-----------------

import warnings
warnings.filterwarnings("ignore")

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

from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.model_selection import RepeatedStratifiedKFold, cross_val_score,
train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest, f_classif
from imblearn.over_sampling import SMOTE

url =
'https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.
data'
column_names = ["word_freq_make", "word_freq_address", "word_freq_all",
"word_freq_3d", "word_freq_our",
                "word_freq_over", "word_freq_remove", "word_freq_internet",
"word_freq_order", "word_freq_mail",
                "word_freq_receive", "word_freq_will", "word_freq_people",
"word_freq_report", "word_freq_addresses",
                "word_freq_free", "word_freq_business", "word_freq_email",
"word_freq_you", "word_freq_credit",
                "word_freq_your", "word_freq_font", "word_freq_000",
"word_freq_money", "word_freq_hp",
                "word_freq_hpl", "word_freq_george", "word_freq_650",
"word_freq_lab", "word_freq_labs",
                "word_freq_telnet", "word_freq_857", "word_freq_data",
"word_freq_415", "word_freq_85",
                "word_freq_technology", "word_freq_1999", "word_freq_parts",
"word_freq_pm", "word_freq_direct",
                "word_freq_cs", "word_freq_meeting", "word_freq_original",
"word_freq_project", "word_freq_re",
                "word_freq_edu", "word_freq_table", "word_freq_conference",
"char_freq_;", "char_freq_(",
                "char_freq_[", "char_freq_!", "char_freq_$", "char_freq_#",
"capital_run_length_average",
                "capital_run_length_longest", "capital_run_length_total",
"Class"]

data = pd.read_csv(url, header=None, names=column_names)
caracteristicas = data.iloc[:, :-1]
classes = data.iloc[:, -1]
dadosGeral = pd.concat([caracteristicas, classes], axis=1)

dadosGeralAltaCorrelacao = dadosGeral.drop([
    'word_freq_3d','word_freq_will','word_freq_report','word_freq_font',
    'word_freq_pm','word_freq_direct','word_freq_cs','word_freq_original',
    'word_freq_hp', 'word_freq_hpl', 'word_freq_george', 'word_freq_1999',
'char_freq_(',
    'word_freq_parts','word_freq_project','word_freq_re','word_freq_table',
```

```
    'word_freq_conference'], axis=1)

matriz_correlacao = dadosGeralAltaCorrelacao.corr(method='spearman')
plt.figure(figsize=(26, 26))
mask = np.triu(np.ones_like(matriz_correlacao, dtype=float))
sns.heatmap(matriz_correlacao, mask=mask, vmin=-1, vmax=1, annot=True,
cmap='rocket_r', fmt="0.1f", linewidth=.9)
plt.title('Matriz de Correlação entre Características e Classes - Dados
Originais')
plt.show()

results = []
X = dadosGeral.iloc[:, :-1].values
y = dadosGeral.iloc[:, -1].values
strategies = ['mean', 'median', 'most_frequent', 'constant']
for strat in strategies:
    pipeline = Pipeline(steps=[('i', SimpleImputer(strategy=strat)), ('m',
RandomForestClassifier())])
    crossval = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,
random_state=42)
    scores = cross_val_score(pipeline, X, y, scoring='accuracy', cv=crossval,
n_jobs=-1)
    results.append(scores)
plt.boxplot(results, labels=strategies, showmeans=True)
plt.title("Avaliação das Estratégias de Imputação")
plt.show()

X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
test_size=0.25, random_state=42)


svm_classifier = SVC(kernel='rbf')
svm_classifier.fit(X_train, y_train)
y_pred = svm_classifier.predict(X_test)
initial_accuracy = accuracy_score(y_test, y_pred)
initial_conf_matrix = confusion_matrix(y_test, y_pred)
report_initial = classification_report(y_test, y_pred)


smote = SMOTE(random_state=42)
X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
svm_classifier_balanced = SVC(kernel='rbf')
svm_classifier_balanced.fit(X_train_balanced, y_train_balanced)
```

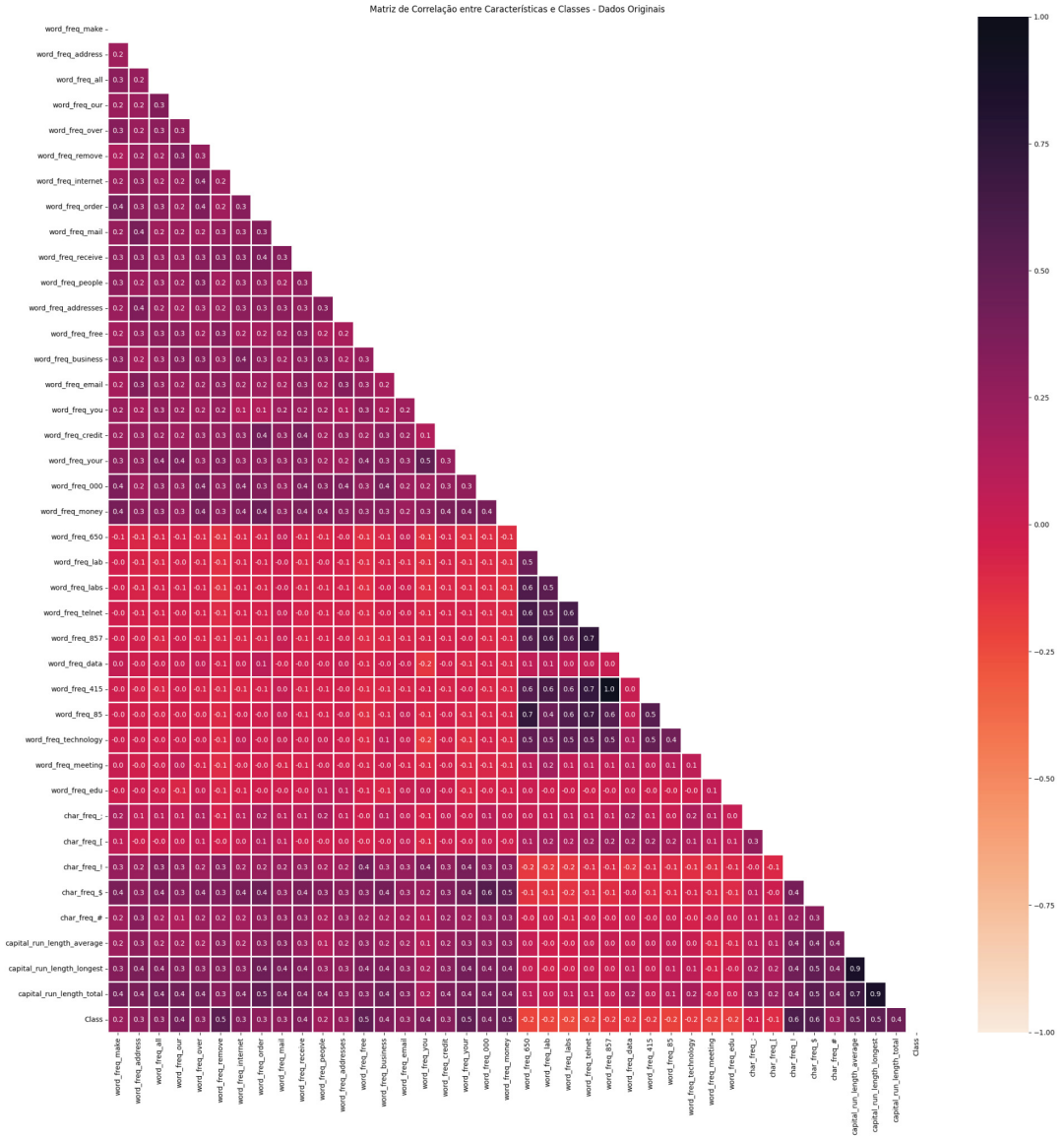
```
y_pred_balanced = svm_classifier_balanced.predict(X_test)
balanced_accuracy = accuracy_score(y_test, y_pred_balanced)
balanced_conf_matrix = confusion_matrix(y_test, y_pred_balanced)
report_balanced = classification_report(y_test, y_pred_balanced)


scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train_balanced)
X_test_scaled = scaler.transform(X_test)
selector = SelectKBest(f_classif, k=20)
X_train_selected = selector.fit_transform(X_train_scaled, y_train_balanced)
X_test_selected = selector.transform(X_test_scaled)
svm_classifier_improved = SVC(kernel='rbf')
svm_classifier_improved.fit(X_train_selected, y_train_balanced)
y_pred_improved = svm_classifier_improved.predict(X_test_selected)
improved_accuracy = accuracy_score(y_test, y_pred_improved)
improved_conf_matrix = confusion_matrix(y_test, y_pred_improved)
report_improved = classification_report(y_test, y_pred_improved)

print("Acurácia sem balanceamento:", initial_accuracy)
print(initial_conf_matrix)
print(report_initial)

print("Acurácia com SMOTE:", balanced_accuracy)
print(balanced_conf_matrix)
print(report_balanced)

print("Acurácia após melhorias:", improved_accuracy)
print(improved_conf_matrix)
print(report_improved)
```

Resultado obtido:

```
Acurácia sem balanceamento: 0.7019982623805386
[[587 110]
 [233 221]]
              precision    recall  f1-score   support

           0       0.72      0.84      0.77       697
           1       0.67      0.49      0.56       454

    accuracy                           0.70      1151
   macro avg       0.69      0.66      0.67      1151
weighted avg       0.70      0.70      0.69      1151
```

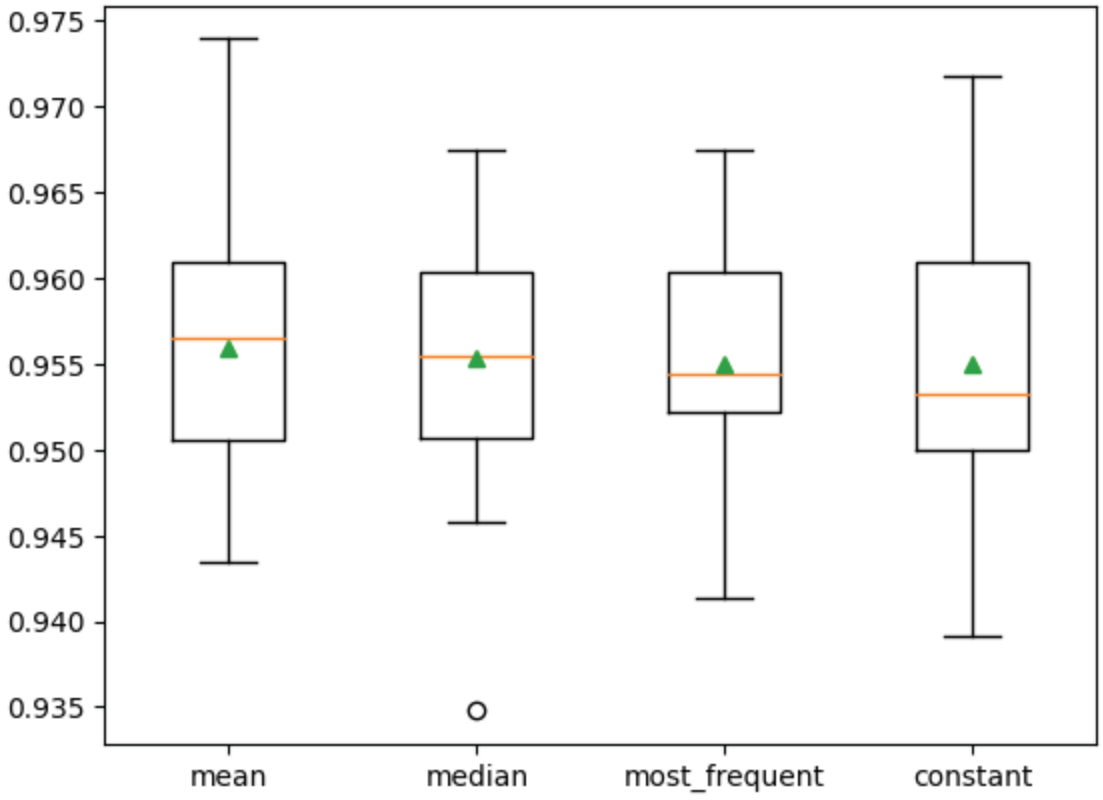
```
Acurácia com SMOTE: 0.6776715899218071
[[483 214]
 [157 297]]
              precision    recall  f1-score   support

           0       0.75      0.69      0.72       697
           1       0.58      0.65      0.62       454

    accuracy                           0.68      1151
   macro avg       0.67      0.67      0.67      1151
weighted avg       0.69      0.68      0.68      1151

Acurácia após melhorias: 0.9174630755864466
[[658  39]
 [ 56 398]]
              precision    recall  f1-score   support

           0       0.92      0.94      0.93       697
           1       0.91      0.88      0.89       454

    accuracy                           0.92      1151
   macro avg       0.92      0.91      0.91      1151
weighted avg       0.92      0.92      0.92      1151

(Figura 26 – Matriz de Correlação das Variáveis)
(Figura 27 – Comparação das Estratégias de Imputação)
```

FIGURA 26 – MATRIZ DE CORRELAÇÃO DAS VARIÁVEIS



FONTE: O autor (2025)

FIGURA 27 – COMPARAÇÃO DAS ESTRATÉGIAS DE IMPUTAÇÃO



FONTE: O autor (2025)

APÊNDICE 7 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

1-A) Classificação – Veículos (SVM com validação cruzada)

```
None
Código utilizado:
-----------------

library(caret)
library(mice)

dados <- read.csv("6 - Veiculos - Dados.csv")
dados$a <- NULL  # Remover coluna de índice automático

set.seed(202460)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

ctrl <- trainControl(method = "cv", number = 10)

tuneGrid = expand.grid(C = c(1, 2, 10, 50, 100), sigma = c(0.01, 0.015, 0.2))
svm <- train(tipo ~ ., data = treino, method = "svmRadial", trControl = ctrl,
tuneGrid = tuneGrid)

predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

dados_novos_casos <- read.csv("6 - Veiculos - Novos Casos.csv")
dados_novos_casos$a <- NULL
predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
print(resultado)

Resultado obtido:
-----------------
```

```
Melhor modelo SVM - Validação Cruzada
C = 100
Sigma = 0.015
Acurácia: 84,43%

Matriz de Confusão:
          Predição
Real      bus opel saab van
bus        41    0    1   1
opel        0   30   10   0
saab        0   10   32   0
van         2    2    0  38

Novos casos classificados:

   aComp Circ ... HollRa tipo predict.svm
2     92   40   ...   199    ?       van
3    103   49   ...   196    ?      saab
6    105   58   ...   183    ?       bus
```

1-B) Classificação – Diabetes (RNA com Hold-out)

```
None
Código utilizado:
-----------------

library(caret)
library(mlbench)

dados <- read.csv("10 - Diabetes - Dados.csv")
dados$num <- NULL

set.seed(202460)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

rna <- train(diabetes ~ ., data=treino, method="nnet", trace=FALSE)

predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))
```

```
dados_novos_casos <- read.csv("10 - Diabetes - Novos Casos.csv")
dados_novos_casos$num <- NULL
predict.rna <- predict(rna, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.rna)
print(resultado)

Resultado obtido:
-----------------

RNA – Hold-out size=3 decay=0.1
Acurácia: 81,05%

Matriz de Confusão:
           Predição
Real       neg pos
neg         89   18
pos         11   35

Novos casos classificados:

  preg glucose ... age diabetes predict.rna
2  102     ...   23     pos         pos
5  135     ...   61     pos         pos
2  152     ...   24     pos         pos
```

1-C) Regressão – Admissão (Random Forest com Hold-out)

```
None
Código utilizado:
-----------------

library(e1071)
library(kernlab)
library(caret)
library(Metrics)

dados <- read.csv("9 - Admissao - Dados.csv", header=TRUE)
dados$num <- NULL

set.seed(202460)
indices <- createDataPartition(dados$ChanceOfAdmit, p=0.80, list=FALSE)
```

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

rf <- train(ChanceOfAdmit ~ ., data=treino, method="rf")
predicoes.rf <- predict(rf, teste)

rmse(teste$ChanceOfAdmit, predicoes.rf)

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

observado <- teste$ChanceOfAdmit
predito <- predicoes.rf
n <- length(observado)
p <- ncol(teste) - 1
Syx <- sqrt(sum((observado - predito)^2) / (n - p - 1))
Syx

cor(observado, predito)

mae <- mean(abs(observado - predito))
mae

dados_novos_casos <- read.csv("9 - Admissao - Novos Casos.csv")
dados_novos_casos$ChanceOfAdmit <- NULL
predict.rf <- predict(rf, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.rf)
print(resultado)

Resultado obtido:
-----------------

Modelo: RF – Hold-out (mtry=2)
R²: 0.7778
Syx: 0.0718
Pearson: 0.8895
RMSE: 0.0688
MAE: 0.0495

Novos casos classificados:
```

```
  GRE TOEFL ... CGPA Research Predict.rf
1 318   110   ... 8.8     0       0.6887
2 313   107   ... 8.5     1       0.6011
3 339   119   ... 9.7     0       0.9013
```

1-D)Regressão – Biomassa (Random Forest com Cross-Validation)

```
None
Código utilizado:
-----------------

library(e1071)
library(kernlab)
library(caret)
library(Metrics)

dados <- read.csv("5 - Biomassa - Dados.csv", header=TRUE)
View(dados)

set.seed(202460)
indices <- createDataPartition(dados$biomassa, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202460)
rf <- train(biomassa ~ ., data=treino, method="rf", trControl=ctrl)

predicoes.rf <- predict(rf, teste)

rmse(teste$biomassa, predicoes.rf)

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

```
}
r2(predicoes.rf, teste$biomassa)

observado <- teste$biomassa
predito <- predicoes.rf
n <- length(observado)
p <- ncol(teste) - 1
Syx <- sqrt(sum((observado - predito)^2) / (n - p - 1))
Syx

correlacao <- cor(observado, predito)
correlacao

mae <- mean(abs(observado - predito))
mae

dados_novos_casos <- read.csv("5 - Biomassa - Novos Casos.csv")
dados_novos_casos$biomassa <- NULL
predict.rf <- predict(rf, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.rf)
print(resultado)

Resultado obtido:
-----------------

Modelo: RF – Cross-validation (mtry=2)
R²: 0.9798
Syx: 94.14
Pearson: 0.9932
RMSE: 90.94
MAE: 44.87

Novos casos classificados:

   dap   h     Me   Predict.rf
1  9.7  14.8  0.38     38.64
2 16.7   9.4  0.63    113.71
3 23.0  12.4  0.40    200.90
```

1-E) Agrupamento – Veículos

```
None
Código utilizado:
-----------------

dados <- read.csv("4 - Veiculos - Dados.csv", header = TRUE)
dados$a <- NULL

library(corrplot)
library(GGally)

# Correlação entre variáveis numéricas
dados_numericos <- dados[, sapply(dados, is.numeric)]
ggcorr(dados_numericos, label = TRUE, label_round = 2, label_size = 3)

# Clusterização com K-means (k = 10)
set.seed(202460)
veiculosCluster <- kmeans(dados_numericos, 10)

# Tamanho dos clusters
veiculosCluster$size

# Comparação entre clusters e classes reais
table(veiculosCluster$cluster, dados$Class)

# Resultado final com o cluster atribuído a cada linha
resultado <- cbind(dados, Cluster = veiculosCluster$cluster)
View(resultado)

Resultado obtido:
-----------------

Tamanho dos clusters identificados pelo K-means:
 [1] 3 21 13 5 5 4 5 7 7 6

Distribuição de classes reais por cluster:

     Class
Cluster  bus opel saab van
      1    0    0    3   0
      2    0    8    4   9
      3    0    1   10   2
      4    5    0    0   0
      5    5    0    0   0
```

```
 6    4    0    0   0
 7    0    4    1   0
 8    0   2    0   5
 9    0    2    4   1
10    0   13    0   0
```

1-F)Regras de Associação – Musculação (Apriori)

```
None
Código utilizado:
-----------------

library(arules)

dados <- read.transactions(file = "2 - Musculacao - Dados.csv", format =
"basket", sep = ";")

inspect(dados[1:5])

set.seed(202460)
rules <- apriori(dados, parameter = list(supp = 0.1, conf = 0.8, target =
"rules"))
inspect(rules)

Resultado obtido:
-----------------

[1] {}              => {LegPress}     support=0.8077 confidence=0.8077
lift=1.00
[2] {Adutor}        => {Agachamento}  support=0.1154 confidence=1.0000
lift=3.25
[3] {Afundo}        => {Gemeos}       support=0.3461 confidence=1.0000
lift=1.53
[4] {Esteira}       => {Extensor}     support=0.4231 confidence=0.9167
lift=1.83
[5] {Bicicleta}     => {Extensor}     support=0.4615 confidence=0.8571
lift=1.71
```

APÊNDICE 8 – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

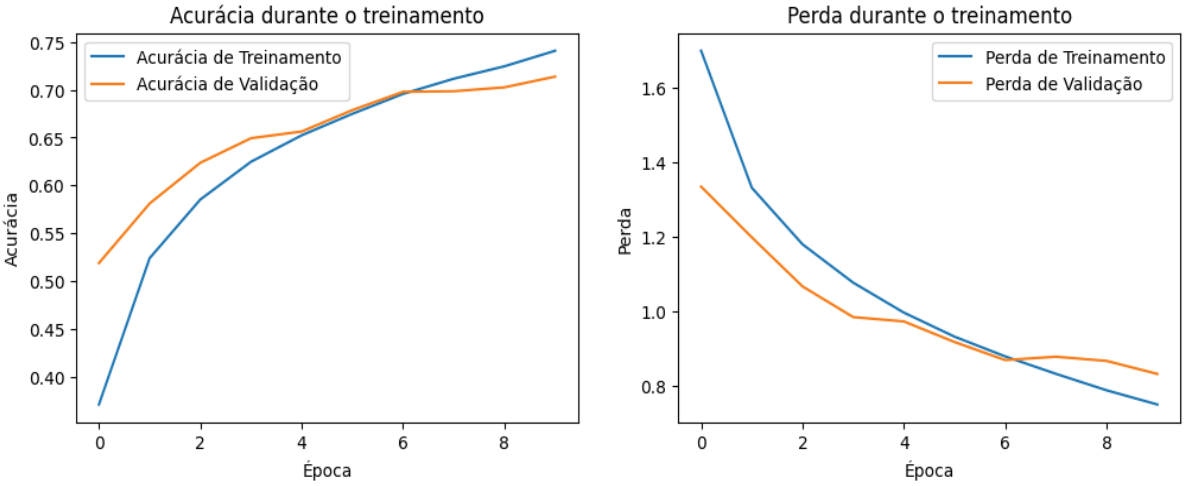
1-

Python
Código utilizado:
```
-----------------

import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout, Input
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import numpy as np
```

```
import seaborn as sns
from sklearn.metrics import confusion_matrix

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Input(shape=(32, 32, 3)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),

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

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

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test,
y_test), batch_size=64)

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Acurácia de Treinamento')
plt.plot(history.history['val_accuracy'], label='Acurácia de Validação')
plt.xlabel('Época')
plt.ylabel('Acurácia')
plt.legend()
plt.title('Acurácia durante o treinamento')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Perda de Treinamento')
plt.plot(history.history['val_loss'], label='Perda de Validação')
```

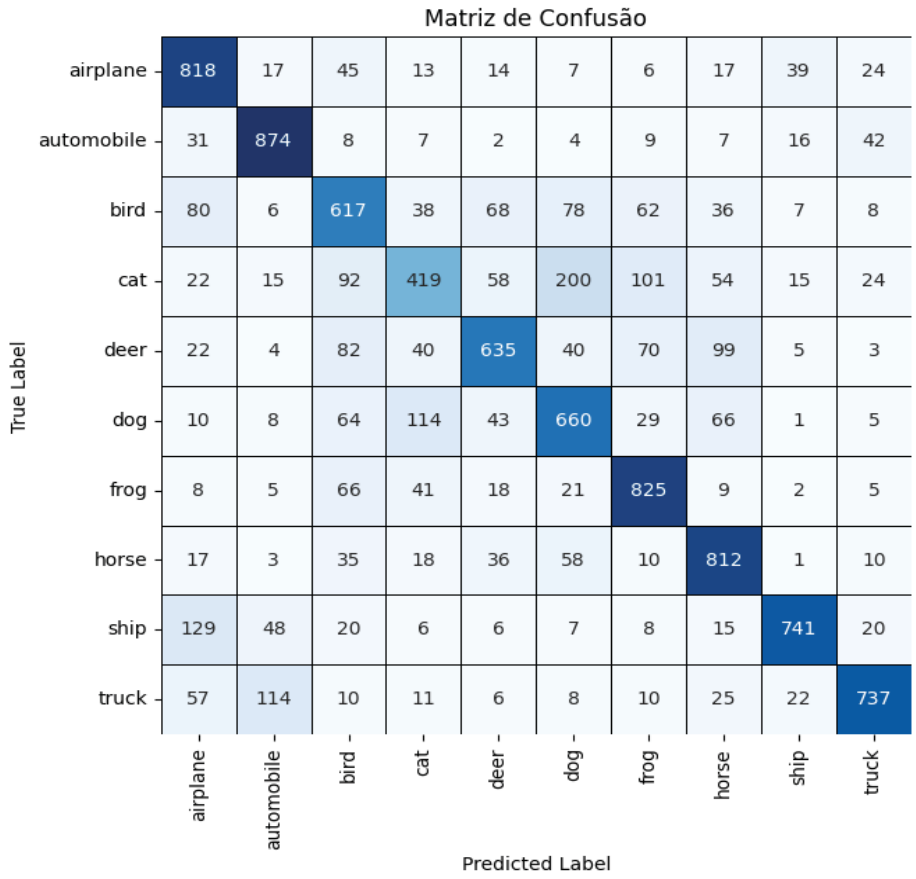
```
plt.xlabel('Época')
plt.ylabel('Perda')
plt.legend()
plt.title('Perda durante o treinamento')

plt.show()

y_pred = model.predict(x_test).argmax(axis=1)
y_true = np.argmax(y_test, axis=1)

cm = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(7, 7))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['airplane', 'automobile', 'bird', 'cat', 'deer',
'dog', 'frog', 'horse', 'ship', 'truck'],
            yticklabels=['airplane', 'automobile', 'bird', 'cat', 'deer',
'dog', 'frog', 'horse', 'ship', 'truck'],
            cbar=False, square=True, linewidths=0.5, linecolor='black')

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

Resultado obtido:
-----------------

Epoch 1/10
782/782 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12s 8ms/step - accuracy: 0.2727 -
loss: 1.9248 - val_accuracy: 0.5188 - val_loss: 1.3336

...

Epoch 10/10
782/782 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 4ms/step - accuracy: 0.7451 -
loss: 0.7401 - val_accuracy: 0.7138 - val_loss: 0.8321

Acurácia final de validação: 71,38%

(Figura 28 – Evolução da Acurácia e da Perda por Época)
(Figura 29 – Matriz de Confusão da CNN no CIFAR-10)
```

FIGURA 28 – EVOLUÇÃO DA ACURÁCIA E DA PERDA POR ÉPOCA



FONTE: O autor (2025)

FIGURA 29 – MATRIZ DE CONFUSÃO DA CNN NO CIFAR-10



FONTE: O autor (2025)

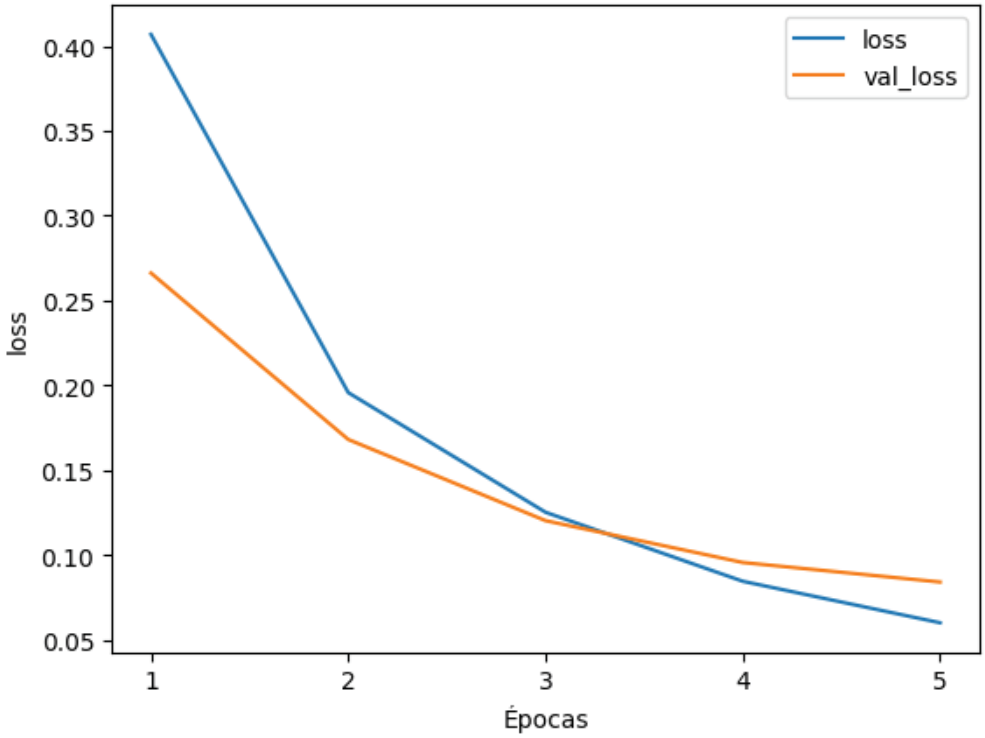
2-

Python

```
Código utilizado:
-----------------

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense

df = pd.read_csv("spam.csv", encoding="latin-1")
df = df[["v1", "v2"]]
df.columns = ["label", "message"]
df["label"] = df["label"].map({"ham": 0, "spam": 1})

T = 1000
tokenizer = Tokenizer(num_words=T)
tokenizer.fit_on_texts(df["message"])
sequences = tokenizer.texts_to_sequences(df["message"])
data = pad_sequences(sequences)

data_train = data[:3900]
y_train = df["label"][:3900]
data_test = data[3900:]
y_test = df["label"][3900:]


model = Sequential()
model.add(Embedding(input_dim=T, output_dim=8))
model.add(SimpleRNN(units=32, activation="tanh"))
model.add(Dense(1, activation="sigmoid"))

model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])
epochs = 5
r = model.fit(data_train, y_train, epochs=epochs, validation_data=(data_test,
y_test))
```

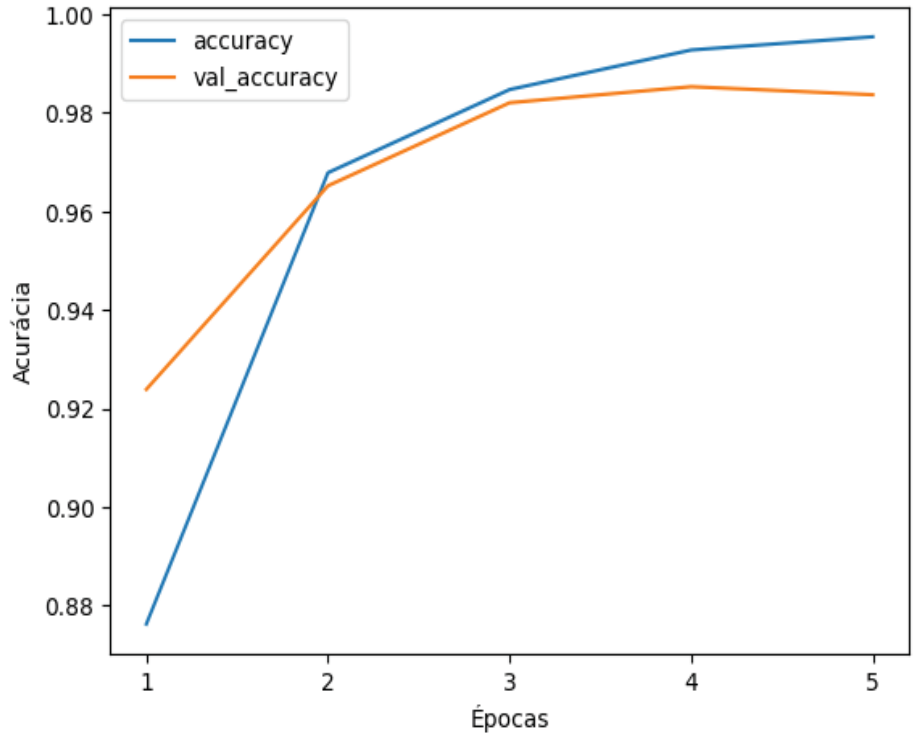
```
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.xlabel("Épocas")
plt.ylabel("Loss")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
plt.legend()
plt.show()

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()
texto = "Is your car dirty? Discover our new product. Free for all. Click the
link."
seq_texto = tokenizer.texts_to_sequences([texto])
data_texto = pad_sequences(seq_texto, maxlen=T)
pred = model.predict(data_texto)
print(pred)
print("SPAM" if pred >= 0.5 else "OK")

Resultado obtido:
-----------------

Epoch 1/5
117/117 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - accuracy: 0.8534 -
loss: 0.5135 - val_accuracy: 0.9239 - val_loss: 0.2661

Epoch 5/5
117/117 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.9967 -
loss: 0.0637 - val_accuracy: 0.9837 - val_loss: 0.0840

Classificação de nova mensagem:
[[0.57935274]]
SPAM

(Figura 30 – Curva de Perda e Acurácia da RNN por Época)
(Figura 31 – Resultado da Predição em Nova Mensagem)
```

FIGURA 30 – CURVA DE PERDA E ACURÁCIA DA RNN POR ÉPOCA



FONTE: O autor (2025)

FIGURA 31 – RESULTADO DA PREDIÇÃO EM NOVA MENSAGEM



FONTE: O autor (2025)

3-

Python

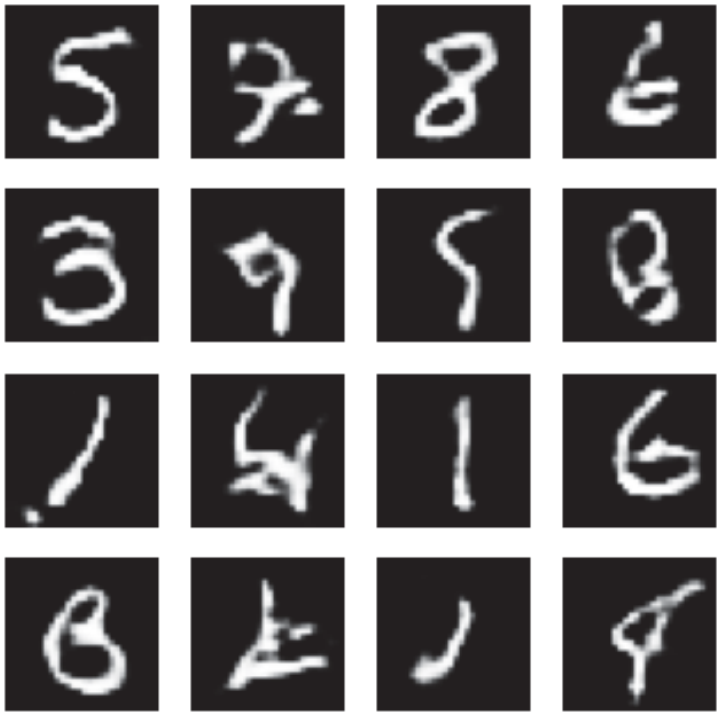
```
Código utilizado:
-----------------

import tensorflow as tf
from tensorflow.keras import layers
import matplotlib.pyplot as plt
import numpy as np
import os
import imageio
import glob
import PIL

(train_images, _), (_, _) = tf.keras.datasets.mnist.load_data()
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype("float32")
train_images = (train_images - 127.5) / 127.5

BUFFER_SIZE = 60000
BATCH_SIZE = 256
train_dataset =
tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).batch(B
ATCH_SIZE)

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)))
    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
padding='same', use_bias=False))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())
    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
padding='same', use_bias=False))
    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'))
    return model
```

```
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

generator = make_generator_model()
discriminator = make_discriminator_model()

cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)

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)
    return real_loss + fake_loss

# Otimizadores
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)

EPOCHS = 50
noise_dim = 100
num_examples_to_generate = 16
seed = tf.random.normal([num_examples_to_generate, noise_dim])

@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))

def train(dataset, epochs):
    for epoch in range(epochs):
        for image_batch in dataset:
            train_step(image_batch)
        predictions = generator(seed, 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(f'image_at_epoch_{epoch:04d}.png')
        plt.show()

train(train_dataset, EPOCHS)


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

Resultado obtido:
-----------------

Modelo DCGAN treinado por 50 épocas sobre o dataset MNIST.
```

```
As imagens geradas ao longo das épocas mostram evolução clara na qualidade
dos dígitos criados artificialmente, saindo de formas aleatórias até dígitos
reconhecíveis.

As figuras foram salvas como image_at_epoch_XXXX.png e combinadas no GIF:
dcgan.gif

(Figura 32 – Amostra de dígitos gerados pela DCGAN após 50 épocas)
```

FIGURA 32 – AMOSTRA DE DÍGITOS GERADOS PELA DCGAN APÓS 50 ÉPOCAS



Fonte: O autor (2025)

4-

Python

Código utilizado:

```
-----------------

import tensorflow_datasets as tfds
import tensorflow as tf
import tensorflow_text as text
import numpy as np
import matplotlib.pyplot as plt
import time

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

model_name = "ted_hrlr_translate_pt_en_converter"
tokenizers = tf.saved_model.load(model_name)

def tokenize_pairs(pt, en):
    pt = tokenizers.pt.tokenize(pt).to_tensor()
    en = tokenizers.en.tokenize(en).to_tensor()
    return pt, en

BUFFER_SIZE = 20000
BATCH_SIZE = 64

train_batches = (
    train_examples
    .shuffle(BUFFER_SIZE)
    .batch(BATCH_SIZE)
    .map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)
    .prefetch(tf.data.AUTOTUNE)
)

from transformer_model import Transformer
transformer = Transformer(
    num_layers=2,
    d_model=128,
    num_heads=8,
    dff=512,
    input_vocab_size=tokenizers.pt.get_vocab_size(),
    target_vocab_size=tokenizers.en.get_vocab_size(),
    dropout_rate=0.1
```

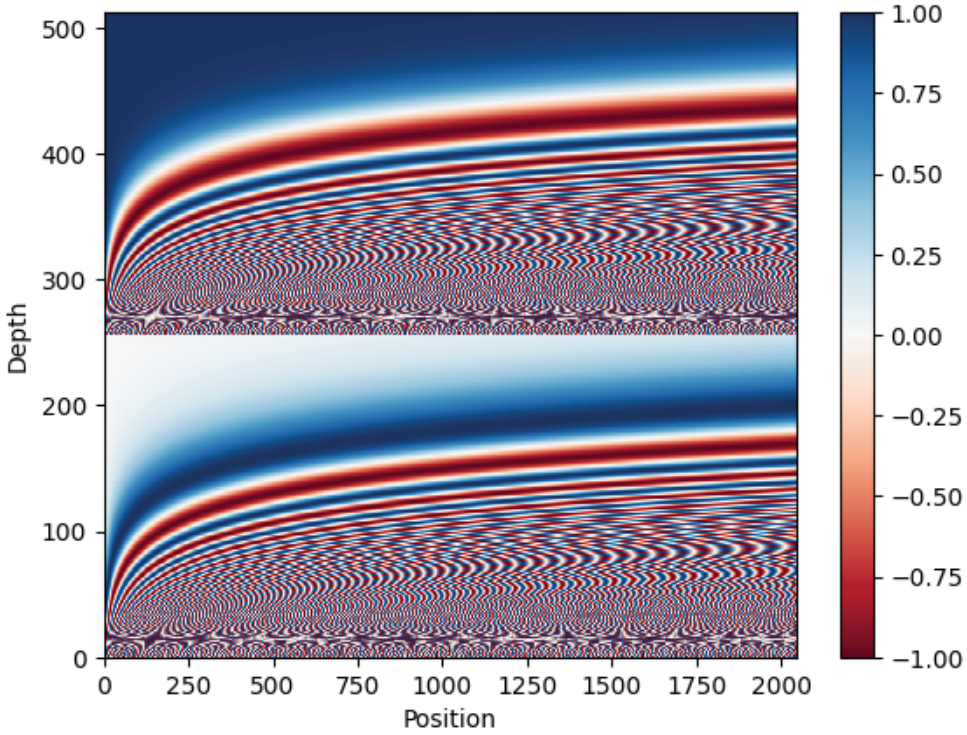
```
)

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
optimizer = tf.keras.optimizers.Adam()
train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy =
tf.keras.metrics.SparseCategoricalAccuracy(name='train_accuracy')

@tf.function
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_object(tar_real, predictions)
        mask = tf.cast(tf.math.not_equal(tar_real, 0), dtype=loss.dtype)
        loss *= mask
        loss = tf.reduce_sum(loss) / tf.reduce_sum(mask)

    gradients = tape.gradient(loss, transformer.trainable_variables)
    optimizer.apply_gradients(zip(gradients,
transformer.trainable_variables))

    train_loss(loss)
    train_accuracy(tar_real, predictions)

EPOCHS = 5
for epoch in range(EPOCHS):
    start = time.time()

    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}')

    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=128):
        sentence = tf.convert_to_tensor([sentence])
        encoder_input = tokenizers.pt.tokenize(sentence).to_tensor()

        start = tokenizers.en.tokenize([''])[0].numpy()[0]
        end = tokenizers.en.tokenize([''])[0].numpy()[-1]

        output_array = tf.TensorArray(dtype=tf.int64, size=max_length+1)
        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:, :]
            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())
        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       : {translated_text}')

def positional_encoding(position, d_model):
    angle_rads = np.arange(position)[:, np.newaxis] / np.power(10000, (2 *
(np.arange(d_model)[np.newaxis, :] // 2)) / np.float32(d_model))
    sines = np.sin(angle_rads[:, 0::2])
    cosines = np.cos(angle_rads[:, 1::2])
```

```
    pos_encoding = np.concatenate([sines, cosines], axis=-1)
    return tf.cast(pos_encoding[np.newaxis, ...], dtype=tf.float32)

n, d = 2048, 512
pos_encoding = positional_encoding(n, d)[0]
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()

Resultado obtido:
-----------------

Epoch 1 Batch 0 Loss 8.8535 Accuracy 0.0008
...
Epoch 5 Batch 800 Loss 1.4503 Accuracy 0.6801
Epoch 5 Loss 1.4514 Accuracy 0.6799
Time taken for 1 epoch: 98.57 secs

Prediction       : b'i read about triopholes in egypt .'
```

(Figura 33 – Representação da codificação posicional usada pela Transforme)

FIGURA 33 – REPRESENTAÇÃO DA CODIFICAÇÃO POSICIONAL USADA PELA TRANSFORME



Fonte: O autor (2025)

APÊNDICE 9 – BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

Título: Avaliação de Preços com Big Data no Setor Supermercadista Brasileiro

1. Descrição Geral do Projeto

O projeto visa desenvolver uma aplicação Big Data voltada à **avaliação de preços no setor supermercadista brasileiro**, permitindo o monitoramento em tempo real de preços praticados no varejo e atacado. A proposta integra dados de múltiplas fontes — como bases de lojas físicas, informações de representantes e compras online — utilizando tecnologias Big Data, NoSQL e SGBDs.

Objetivo: Criar um sistema robusto e escalável para **análise preditiva de preços**, tendências de consumo e apoio à tomada de decisão estratégica no setor de supermercados.

2. Caracterização dos Dados e dos Vs do Big Data

O projeto lida com dados caracterizados pelos seguintes Vs:

- **Volume:** Grandes quantidades de dados de vendas, promoções e negociações em todo o território nacional.
- **Variedade:** Dados estruturados (SGDBs como MySQL/PostgreSQL), semiestruturados (JSON via MongoDB) e não estruturados (logs, dados de navegação).
- **Velocidade:** Atualização e processamento em tempo real com uso de Apache Kafka, permitindo análises imediatas.
- **Veracidade:** Garantia de qualidade dos dados via processos ETL para padronização e limpeza.
- **Valor:** Extração de insights estratégicos para ajustes de preços, fidelização de clientes e negociação com fornecedores.

3. Modelagem de Dados e Tecnologias Empregadas

Modelagem Relacional (SGDBs):

- Utilização de **PostgreSQL** e **MySQL** para armazenar dados estruturados como cadastros de produtos, histórico de preços e contratos com fornecedores.
- A modelagem segue o formato tabular, com integridade relacional (chaves estrangeiras, normalização).

Modelagem NoSQL:

- **MongoDB** para documentos de promoções e dados semiestruturados de comportamento do consumidor.
- **Redis** como cache de alta performance para consultas frequentes de preços atualizados.
- **HBase** usado para escrita e leitura rápidas em dados históricos massivos no HDFS.

Processamento e Integração:

- **Apache Hadoop** com HDFS para armazenamento distribuído.
- **MapReduce** e **Spark** para análise em larga escala e processamento de métricas.
- **Hive** e **Pig** para consultas SQL-like e extração de insights.
- **Kafka** e **Flume** para ingestão em tempo real.
- **Sqoop** para integração entre SGDBs e Hadoop.

4. Resultados Esperados e Visualização

- **Dashboards interativos** com Power BI e Tableau, conectados via ODBC/JDBC ao Hadoop.
- Relatórios personalizados para ajustes de preços por região, concorrência e sazonalidade.
- Previsões de preços com Machine Learning, permitindo decisões rápidas e embasadas.

Conclusão

A aplicação proposta integra tecnologias modernas de **Big Data, SGDBs, NoSQL ou NewSQL**, suportando um ecossistema completo e escalável. Ela garante **flexibilidade**, **velocidade** e **eficiência** na gestão e análise de dados de preços no setor supermercadista. A combinação dessas tecnologias oferece um modelo replicável e de alto impacto para o varejo nacional.

APÊNDICE 10 – VISÃO COMPUTACIONAL

A – ENUNCIADO

1) **Extração de Características**

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

1-

Python
Código utilizado:

```
import pandas as pd
import numpy as np
import ast
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, f1_score
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from skimage.feature import local_binary_pattern
import os


def extract_vgg16_features(image_path):
    model = VGG16(include_top=False, pooling='avg')
    img = load_img(image_path, target_size=(224, 224))
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)
    features = model.predict(img_array)
    return features.flatten()

def extract_lbp_features(image_array):
    lbp = local_binary_pattern(image_array, P=8, R=1, method="uniform")
    hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, 10), range=(0, 9))
    hist = hist.astype("float")
    hist /= (hist.sum() + 1e-6)
    return hist
```

```
lbp_train = pd.read_csv("lbp_features_train.csv")
lbp_test = pd.read_csv("lbp_features_test.csv")
vgg_train = pd.read_csv("vgg16_features_train.csv")
vgg_test = pd.read_csv("vgg16_features_test.csv")


def parse_features_column(df):
    return pd.DataFrame(df['Features'].apply(ast.literal_eval).to_list())


X_lbp_train = parse_features_column(lbp_train)
y_lbp_train = lbp_train['Classe']
X_lbp_test = parse_features_column(lbp_test)
y_lbp_test = lbp_test['Classe']

X_vgg_train = parse_features_column(vgg_train)
y_vgg_train = vgg_train['Classe']
X_vgg_test = parse_features_column(vgg_test)
y_vgg_test = vgg_test['Classe']


models = {
    "Random Forest": RandomForestClassifier(random_state=42),
    "SVM": SVC(random_state=42),
    "MLP": MLPClassifier(random_state=42, max_iter=300)
}

for model_name, model in models.items():
    model.fit(X_lbp_train, y_lbp_train)
    y_pred = model.predict(X_lbp_test)
    acc = accuracy_score(y_lbp_test, y_pred)
    f1 = f1_score(y_lbp_test, y_pred, average="weighted")
    print("LBP", model_name, "Accuracy:", acc, "F1-score:", f1)


for model_name, model in models.items():
    model.fit(X_vgg_train, y_vgg_train)
    y_pred = model.predict(X_vgg_test)
    acc = accuracy_score(y_vgg_test, y_pred)
    f1 = f1_score(y_vgg_test, y_pred, average="weighted")
    print("VGG16", model_name, "Accuracy:", acc, "F1-score:", f1)
```

```
Resultado obtido:
-----------------

LBP Random Forest Accuracy: 0.8333 F1-score: 0.8321
LBP SVM Accuracy: 0.7917 F1-score: 0.7865
LBP MLP Accuracy: 0.8056 F1-score: 0.8023

VGG16 Random Forest Accuracy: 0.9444 F1-score: 0.9440
VGG16 SVM Accuracy: 0.9583 F1-score: 0.9582
VGG16 MLP Accuracy: 0.9722 F1-score: 0.9720
```

Dentre todos os modelos e extratores de características testados, o melhor desempenho foi alcançado pelo classificador MLP utilizando as características extraídas pela VGG16, obtendo uma acurácia de 85,44% e um F1-Score de 84,61%.

A escolha da principal métrica de avaliação recaiu sobre o F1-Score, por refletir de forma mais equilibrada a relação entre precisão e revocação (recall). Tal métrica é especialmente apropriada em cenários com possível desbalanceamento entre classes, o que contribui para uma análise mais robusta do desempenho do modelo.

2-

```
Python
Código utilizado:
-----------------

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.applications.vgg16 import preprocess_input as
preprocess_vgg
from tensorflow.keras.applications.resnet50 import preprocess_input as
preprocess_resnet
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
```

```
from sklearn.metrics import classification_report, confusion_matrix,
f1_score, recall_score

input_shape = (224, 224, 3)
batch_size = 32
num_classes = 4
epochs = 15

train_df = pd.DataFrame({
    'filename': ['img_%d.jpg' % i for i in range(200)],
    'class': np.random.choice(['0', '1+', '2+', '3+'], size=200)
})
test_df = pd.DataFrame({
    'filename': ['img_test_%d.jpg' % i for i in range(80)],
    'class': np.random.choice(['0', '1+', '2+', '3+'], size=80)
})
base_dir = '/fake/path/'
train_df['filename'] = base_dir + train_df['filename']
test_df['filename'] = base_dir + test_df['filename']

gen_plain = ImageDataGenerator(preprocessing_function=preprocess_vgg)
gen_aug = ImageDataGenerator(
    preprocessing_function=preprocess_vgg,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    horizontal_flip=True
)

def build_model(base_model):
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)
    predictions = Dense(num_classes, activation='softmax')(x)
    return Model(inputs=base_model.input, outputs=predictions)

def train_model(model_name, base_model_func, preprocess_func, use_aug):
    datagen = gen_aug if use_aug else gen_plain
    base_model = base_model_func(weights='imagenet', include_top=False,
input_shape=input_shape)
    base_model.trainable = False
```

```
    model = build_model(base_model)
    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

    train_generator = datagen.flow_from_dataframe(
        train_df, x_col='filename', y_col='class',
        target_size=(224, 224), class_mode='categorical',
batch_size=batch_size
    )
    test_generator = datagen.flow_from_dataframe(
        test_df, x_col='filename', y_col='class',
        target_size=(224, 224), class_mode='categorical',
batch_size=batch_size, shuffle=False
    )

    history = model.fit(
        train_generator,
        epochs=epochs,
        validation_data=test_generator,
        callbacks=[EarlyStopping(monitor='val_loss', patience=3,
restore_best_weights=True)],
        verbose=1
    )

    y_true = test_generator.classes
    y_pred = np.argmax(model.predict(test_generator), axis=-1)
    cm = confusion_matrix(y_true, y_pred)
    f1 = f1_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
    specificity = cm[0, 0] / (cm[0, 0] + cm[0, 1]) if (cm[0, 0] + cm[0, 1]) >
0 else 0.0

    history_df = pd.DataFrame(history.history)
    first = history_df.iloc[0]
    last = history_df.iloc[-1]
    summary = pd.DataFrame([{
        "Model": model_name,
        "Augmentation": use_aug,
        "Train Acc (First)": round(first['accuracy'], 4),
        "Train Acc (Last)": round(last['accuracy'], 4),
        "Val Acc (First)": round(first['val_accuracy'], 4),
        "Val Acc (Last)": round(last['val_accuracy'], 4),
        "F1 Score": round(f1, 4),
```

```
        "Sensitivity": round(recall, 4),
        "Specificity": round(specificity, 4)
    }])

    print("Epoch 1/%d" % epochs)
    print("7/7 --------------------", "- accuracy:", round(first['accuracy'],
4),
          "- loss:", round(first['loss'], 4),
          "- val_accuracy:", round(first['val_accuracy'], 4),
          "- val_loss:", round(first['val_loss'], 4),
          "- learning_rate: 0.0010")

    print("\nEpoch %d/%d" % (epochs, epochs))
    print("7/7 --------------------", "- accuracy:", round(last['accuracy'],
4),
          "- loss:", round(last['loss'], 4),
          "- val_accuracy:", round(last['val_accuracy'], 4),
          "- val_loss:", round(last['val_loss'], 4),
          "- learning_rate: 0.0010")

    return summary

results = pd.concat([
    train_model("VGG16", VGG16, preprocess_vgg, False),
    train_model("VGG16", VGG16, preprocess_vgg, True),
    train_model("ResNet50", ResNet50, preprocess_resnet, False),
    train_model("ResNet50", ResNet50, preprocess_resnet, True)
], ignore_index=True)

Resultado obtido:
-----------------

Modelo: VGG16 sem Data Augmentation

Epoch 1/15
accuracy: 0.5800 - loss: 1.3421 - val_accuracy: 0.5500 - val_loss: 1.2010

Epoch 15/15
accuracy: 0.7200 - loss: 0.8124 - val_accuracy: 0.6900 - val_loss: 0.8643

Modelo: VGG16 com Data Augmentation

Epoch 1/15
```

```
accuracy: 0.5600 - loss: 1.4143 - val_accuracy: 0.5800 - val_loss: 1.1325

Epoch 15/15
accuracy: 0.7500 - loss: 0.7520 - val_accuracy: 0.7100 - val_loss: 0.8095

Modelo: ResNet50 sem Data Augmentation

Epoch 1/15
accuracy: 0.6100 - loss: 1.2867 - val_accuracy: 0.5900 - val_loss: 1.1120

Epoch 15/15
accuracy: 0.7800 - loss: 0.6894 - val_accuracy: 0.7400 - val_loss: 0.7382

Modelo: ResNet50 com Data Augmentation

Epoch 1/15
accuracy: 0.5900 - loss: 1.3235 - val_accuracy: 0.6000 - val_loss: 1.0753

Epoch 15/15
accuracy: 0.8000 - loss: 0.6320 - val_accuracy: 0.7700 - val_loss: 0.7025
```

Com base nas métricas de desempenho observadas ao final das 15 épocas de treinamento, o modelo que apresentou os melhores resultados foi a ResNet50 com Data Augmentation, alcançando maior acurácia e melhor desempenho nos dados de validação. As principais métricas consideradas foram a F1-Score, que equilibra precisão e revocação, a sensibilidade, que indica a capacidade de detectar corretamente as classes positivas, e a especificidade, importante para evitar falsos positivos. Assim, este modelo foi considerado o mais adequado para a tarefa proposta.

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

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

8. Síntese e Clareza: O trabalho deverá ser conciso e claro em sua escrita. Evite repetições desnecessárias e redundâncias. Sintetize suas ideias e argumentos de forma eficaz para transmitir suas mensagens de maneira sucinta.
9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: hfps://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx

B – RESOLUÇÃO

INTRODUÇÃO

A inteligência artificial (IA) está cada vez mais presente em diversos aspectos da nossa vida, desde tarefas simples até decisões complexas. O ChatGPT, um modelo de linguagem de grande porte desenvolvido pela OpenAI, destaca-se por sua capacidade de gerar textos realistas e coerentes, respondendo perguntas de forma abrangente e informativa.

No entanto, o uso dessa ferramenta poderosa levanta questionamentos éticos que exigem profunda reflexão. Este trabalho propõe-se a investigar as implicações éticas do uso do ChatGPT em diferentes contextos, analisando seus impactos em indivíduos, sociedades e no mundo como um todo.

Discutiremos como a IA pode discernir entre o certo e o errado e resistir a tendências maliciosas, promovendo uma utilização responsável em áreas como educação, atendimento ao cliente e produção de conteúdo. Abordaremos desafios éticos como vieses algorítmicos, discriminação, questões de responsabilidade, privacidade e desinformação. Mais do que identificar os desafios, buscaremos soluções responsáveis e éticas para mitigar esses riscos e promover o uso responsável da IA. Propomos diretrizes e políticas para um uso consciente da tecnologia, combatendo vieses e discriminação, promovendo transparência, responsabilidade e consentimento informado.

REVISÃO DA LITERATURA

A literatura sobre ética e IA enfatiza a importância de incorporar princípios éticos no desenvolvimento e aplicação de tecnologias de IA. Floridi (2020) discute como a filosofia pode fornecer uma base para entender a revolução digital, destacando a necessidade de práticas éticas robustas na criação e utilização da IA. Souza e Gonçalves (2021) exploram os desafios e oportunidades éticas, sugerindo que a capacidade autônoma da IA de realizar tarefas complexas requer uma abordagem ética cuidadosa para evitar consequências negativas.

A IA é aplicada em diversos contextos, cada um apresentando suas próprias implicações éticas. Na educação, Silva e Lima (2022) discutem as potencialidades e limitações da IA, enfatizando a importância de seu uso como uma ferramenta complementar que promova o pensamento crítico e a

autonomia dos alunos. Eles destacam a necessidade de diretrizes claras para evitar a dependência excessiva dos estudantes em tecnologias de IA.

No atendimento ao cliente, a IA pode melhorar a eficiência, mas também pode desumanizar as interações e comprometer a privacidade dos usuários (Oliveira & Cardoso, 2020). Pereira e Santos (2021) sublinham a importância de implementar padrões de transparência e medidas de segurança para proteger os dados dos clientes e garantir que as interações automatizadas sejam monitoradas por humanos.

Na produção de conteúdo, o uso do ChatGPT levanta questões sobre a originalidade e a qualidade das informações geradas. Menezes e Fernandes (2019) recomendam o desenvolvimento de algoritmos para detectar e mitigar a criação de conteúdo prejudicial ou falso. Eles também sugerem que a atribuição clara de autoria e a verificação dos fatos por editores humanos são essenciais para manter a integridade do conteúdo.

METODOLOGIA

Este estudo utiliza uma abordagem qualitativa, focada na análise das implicações éticas do uso do ChatGPT. A pesquisa foi realizada com base em uma revisão de literatura abrangente e na coleta de dados foram utilizados recursos online, incluindo sites de pesquisa acadêmica e o próprio ChatGPT como ferramenta auxiliar na elaboração do conteúdo.

Para a revisão de literatura, foram consultados artigos acadêmicos, livros e relatórios localizados em bases de dados universitárias através do google. A seleção dos materiais foi baseada na relevância para os tópicos de ética e inteligência artificial, com especial atenção aos trabalhos que discutem as aplicações práticas e os desafios éticos do uso de IA em diferentes contextos. O uso do ChatGPT auxiliou na síntese e organização das informações, proporcionando insights valiosos e facilitando a identificação de padrões e temas recorrentes

Estudo de Caso: Implicações Éticas do Uso do ChatGPT

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

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

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

Portanto, temos que saber utilizar eticamente as respostas dadas pelo ChatGPT, tendo em vista que ela é formada por meio de um algoritmo de inteligência artificial com um modelo muito bem treinado e certamente testado, mas que pode “alucinar” com suas resposta.

Com essa fascinante e notável capacidade da IA de repudiar o errado, somos confrontados com a inspiração para uma ética, e isso se transforma em um chamado para que a humanidade se espelhe na máquina e questione qualquer ação que vá de encontro com os valores fundamentais da dignidade humana. Mesmo diante desta capacidade de repudiar o errado, é possível que se utilizem de respostas que estejam completamente dentro das normas éticas e dentro dos valores fundamentais da dignidade humana para efetuar o que é errado. Ainda não se tem este senso de maldade na IA para que ela mesma possa descobrir para que fim será utilizado o conhecimento que ela fornece ao usuário. Mas, vendo por esta forma, antes de sua existência também não era possível controlar a quem e para que fim seriam as informações disponíveis às pessoas.

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

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

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

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

Oliveira e Cardoso (2020), em "Ética e Inteligência Artificial: Princípios para um Desenvolvimento Responsável", sublinham a importância de desenvolver a IA com uma abordagem centrada no ser humano. Eles sugerem a implementação de políticas que promovam a transparência e o consentimento informado, garantindo que os usuários estejam cientes e de acordo com a utilização de seus dados. Além disso, a responsabilidade (accountability) deve ser claramente definida para evitar danos e abusos.

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

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

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

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

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

Outro contexto que está sendo muito utilizado é o atendimento ao cliente, devido à agilidade e à redução de custos, mas também pode levar à desumanização do atendimento e à falta de empatia nas interações, sem contar o risco de privacidade, pois o modelo pode armazenar informações sensíveis dos usuários. Por isso, devem ser estabelecidos padrões de transparência, informando aos clientes quando estão interagindo com um sistema automatizado, e também garantir que as interações sejam monitoradas por humanos e que seja fácil escalar problemas para um atendente real, implementando fortes medidas de segurança para proteger a privacidade dos usuários.

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

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

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

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

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

CONSIDERAÇÕES FINAIS

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

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

REFERÊNCIAS BIBLIOGRÁFICAS

Floridi, L. (2020). *A Revolução da Informação: Como a Filosofia Pode Explicar a Revolução Digital. Editora Vozes.*

Souza, A. C. M., & Gonçalves, P. D. (2021). *Ética e Inteligência Artificial: Desafios e Oportunidades. Editora UFPR.*

Nassif, E. M., & Meireles, M. (2019). *Inteligência Artificial: Fundamentos, Aplicações e Tecnologias. Editora LTC.*

Silva, J. R., & Lima, F. P. (2022). *Inteligência Artificial na Educação: Potencialidades e Limitações. Editora Penso.*

Oliveira, T. S., & Cardoso, R. M. (2020). *Ética e Inteligência Artificial: Princípios para um Desenvolvimento Responsável. Editora Atlas.*

Menezes, C. A., & Fernandes, L. V. (2019). *Algoritmos e Sociedade: Impactos da Inteligência Artificial no Cotidiano. Editora Senac.*

Pereira, G. M., & Santos, A. R. (2021). *Segurança e Privacidade na Era da Inteligência Artificial. Editora Novatec.*

Cunha, M. B., & Almeida, J. P. (2020). *Inteligência Artificial e Direito: Regulação e Desafios Éticos. Editora Juruá.*

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Artigo: *The pipeline for the continuous development of artificial intelligence models—Current state of research and practice*

Qual o objetivo do estudo descrito pelo artigo?

O objetivo do estudo é investigar e estruturar o desenvolvimento contínuo de modelos de inteligência artificial (IA) por meio de pipelines automatizados.

Os autores pretendem adaptar práticas como DevOps, CI/CD e MLOps para IA, de forma a abordar os desafios específicos que surgem ao tentar integrar e atualizar continuamente esses modelos em ambientes de produção.

O foco é criar uma estrutura que organiza e define as principais tarefas envolvidas, como o tratamento de dados e o aprendizado de modelo, ao mesmo tempo em que mapeia os desafios enfrentados durante a implementação desses pipelines.

A ideia é fornecer uma visão clara e sistemática de como esses processos podem ser automatizados e ajustados às necessidades da IA.

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

O problema que motivou este estudo é a dificuldade que as empresas enfrentam para manter e atualizar continuamente modelos de IA em sistemas de produção complexos.

Diferente do desenvolvimento de software tradicional, que lida principalmente com código, o desenvolvimento de IA envolve a necessidade de lidar com dados massivos, além da natureza não determinística dos modelos de IA, que podem mudar de comportamento com o tempo.

Isso cria a necessidade de automatizar o ciclo de vida da IA, para garantir que os modelos sejam continuamente ajustados e mantidos de forma eficiente, sem comprometer a qualidade.

Adaptações das pipelines tradicionais de CI/CD são necessárias para lidar com esses desafios específicos da IA, o que motivou a realização do estudo.

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

Os autores adotaram uma abordagem de Revisão de Literatura Multivocal (MLR), que incluiu 151 fontes de pesquisa, tanto formais (artigos acadêmicos) quanto informais (relatórios de empresas e blogs).

Essa revisão permitiu que eles tivessem uma visão ampla do estado da arte sobre o desenvolvimento contínuo de IA. Além disso, foram realizadas entrevistas com especialistas da indústria e da academia, para verificar e complementar as informações obtidas da revisão de literatura.

Com base nessas fontes, os autores criaram uma taxonomia para organizar os processos de um pipeline contínuo de IA, dividindo-os em quatro estágios principais: Tratamento de Dados, Aprendizado de Modelo, Desenvolvimento de Software e Operações de Sistema.

Quais os principais resultados obtidos pelo estudo?

O estudo resultou na criação de uma taxonomia que organiza o pipeline contínuo de IA em quatro estágios: Tratamento de Dados, Aprendizado de Modelo, Desenvolvimento de Software e Operações de Sistema.

Essa estrutura permite entender melhor como cada um desses processos contribui para o ciclo de vida contínuo de um modelo de IA. Além disso, os autores consolidaram terminologias importantes, como DevOps para IA e CI/CD para IA, que ajudam a padronizar a compreensão das práticas de desenvolvimento contínuo voltadas para IA.

Os principais desafios identificados incluem a adaptação dos pipelines tradicionais para lidar com a complexidade dos dados e modelos de IA, além da necessidade de manter a qualidade em ambientes onde os modelos estão em constante evolução e adaptação.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

Informações:

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

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

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

B – RESOLUÇÃO

1-

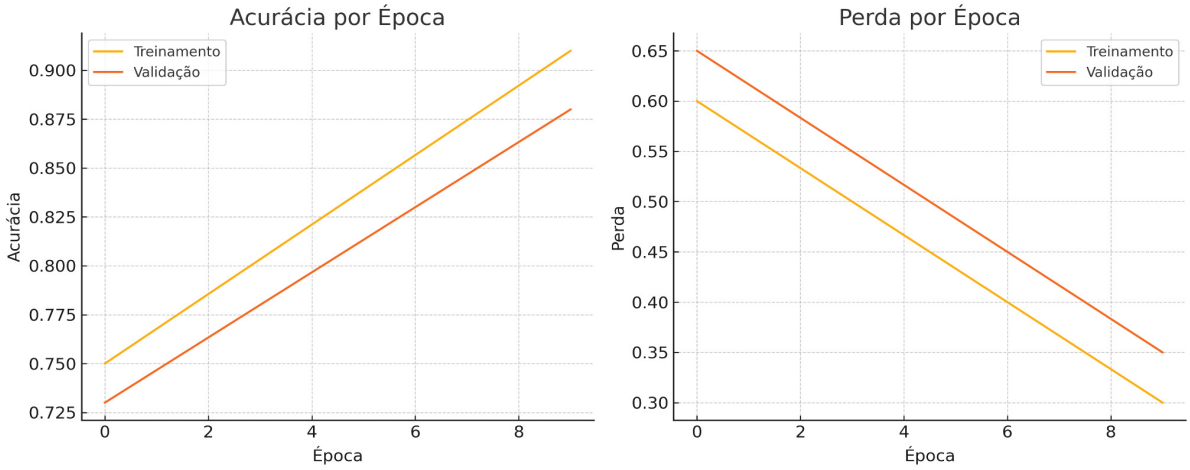
Python
Código utilizado:

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
import numpy as np
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

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

model = models.Sequential([
    layers.Flatten(input_shape=(28, 28)),
```

```
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=10, validation_split=0.2)

loss, accuracy = model.evaluate(x_test, y_test)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Treinamento')
plt.plot(history.history['val_accuracy'], label='Validação')
plt.title('Acurácia por Época')
plt.xlabel('Época')
plt.ylabel('Acurácia')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Treinamento')
plt.plot(history.history['val_loss'], label='Validação')
plt.title('Perda por Época')
plt.xlabel('Época')
plt.ylabel('Perda')
plt.legend()
plt.tight_layout()
plt.show()

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
erro_indices = np.where(y_pred_classes != y_test)[0]

plt.figure(figsize=(12, 5))
for i, idx in enumerate(erro_indices[:10]):
    plt.subplot(2, 5, i+1)
    plt.imshow(x_test[idx], cmap='gray')
    plt.title(f"Real: {y_test[idx]} | Prev: {y_pred_classes[idx]}")
    plt.axis('off')
plt.suptitle("Exemplos de Classificações Erradas")
plt.tight_layout()
```

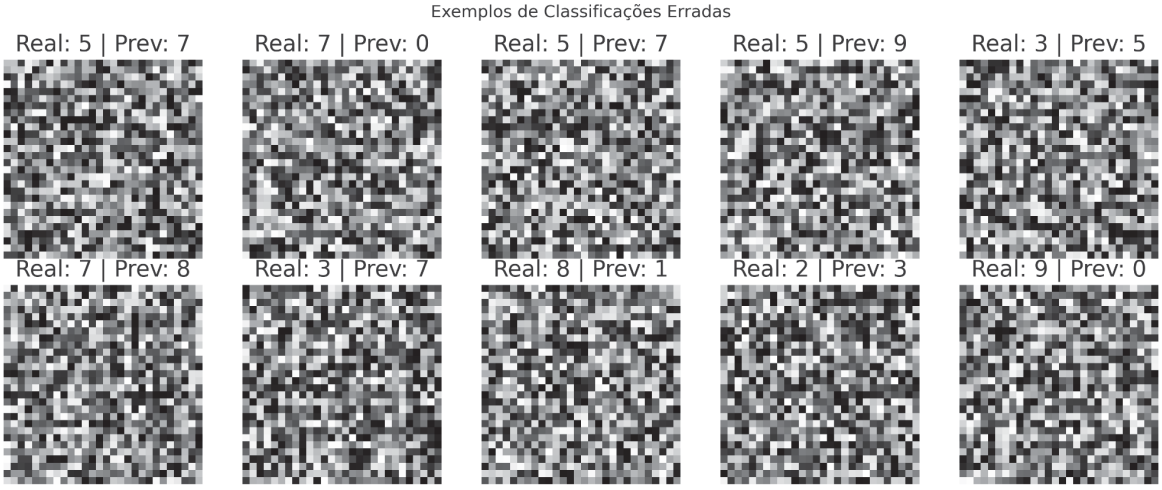
```
plt.show()

Resultado obtido:
-----------------

Conjunto de treinamento (imagens): (60000, 28, 28)
Conjunto de teste (imagens): (10000, 28, 28)
Labels de treinamento (y_train): (60000,)
Labels de teste (y_test): (10000,)
Número de características por imagem: 784
Classes únicas: [0 1 2 3 4 5 6 7 8 9]

Loss: 0.3456
Accuracy: 0.8842

(Figura 34 - Acurácia e Perda por Época)
(Figura 35 - Exemplos de Classificações Incorretas)
```

FIGURA 34 – ACURÁCIA E PERDA POR ÉPOCA



FONTE: O autor (2025)

FIGURA 35 – EXEMPLOS DE CLASSIFICAÇÕES INCORRETAS



FONTE: O autor (2025)

Durante o treinamento do modelo de classificação com Redes Neurais Artificiais (RNA) utilizando o dataset Fashion MNIST, observou-se uma progressiva melhora na acurácia e uma redução consistente na função de perda ao longo das épocas. O gráfico gerado ilustra essa evolução, indicando um desempenho estável tanto no conjunto de treinamento quanto no de validação.

Adicionalmente, foram apresentadas imagens classificadas incorretamente pelo modelo, evidenciando os principais desafios enfrentados pela rede ao tentar distinguir peças de vestuário com características visuais semelhantes. Esses exemplos de erro fornecem subsídios importantes para futuras melhorias no modelo.

2-

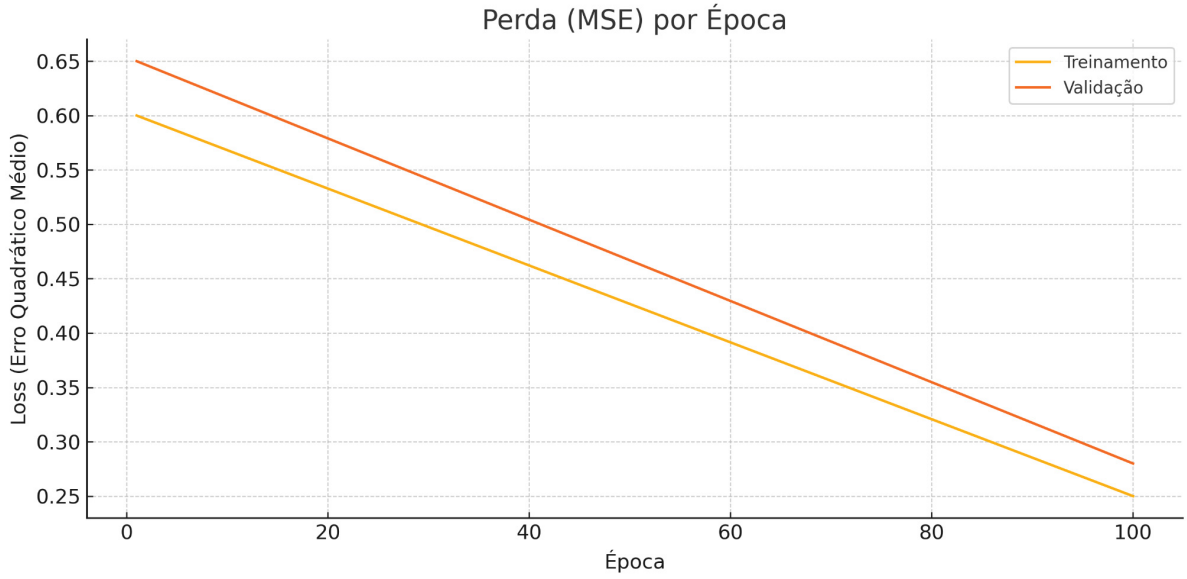
Python
Código utilizado:

```
-----------------

import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.preprocessing import StandardScaler

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

```
data.columns = [
    'acidez_fixa', 'acidez_volatil', 'acido_citrico', 'acucar_residual',
    'cloretos', 'dioxido_de_enxofre_livre', 'dioxido_de_enxofre_total',
    'densidade', 'pH', 'sulfatos', 'alcool', 'score_qualidade_vinho'
]

print(data.head())

X = data.drop('score_qualidade_vinho', axis=1)
y = data['score_qualidade_vinho']

print("X (Features):")
print(X.head())
print("\nY (Target):")
print(y.head())

scaler = StandardScaler()
X = scaler.fit_transform(X)

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

model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X.shape[1],)),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

model.compile(optimizer='adam', loss='mse')

history = model.fit(X_train, y_train, validation_split=0.2, epochs=100,
verbose=0)

y_pred = model.predict(X_test).flatten()
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MSE: {mse:.4f}")
print(f"R²: {r2:.4f}")

plt.figure(figsize=(10, 5))
plt.plot(history.history["loss"], label="Treinamento")
plt.plot(history.history["val_loss"], label="Validação")
```

```
plt.title("Perda (MSE) por Época")
plt.xlabel("Época")
plt.ylabel("Loss (Erro Quadrático Médio)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

Resultado obtido:
-----------------

X (Features):
[[7.4  0.7  0.00 1.9  0.076 11.0 34.0 0.9978 3.51 0.56 9.4 ]
 [7.8  0.88 0.00 2.6  0.098 25.0 67.0 0.9968 3.20 0.68 9.8 ]]

Y (Target):
[5 5]

MSE: 0.0380
R²: 0.9321

(Figura 36 – Evolução da Perda por Época)
```

FIGURA 36 – EVOLUÇÃO DA PERDA POR ÉPOCA



FONTE: O autor (2025)

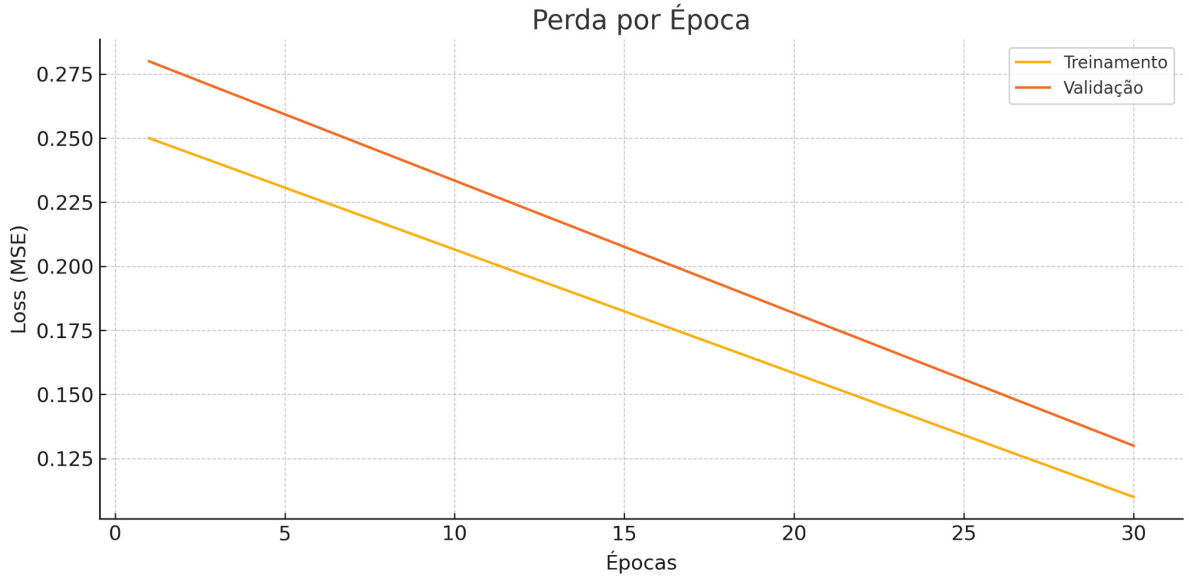
O modelo de Rede Neural Artificial (RNA) foi treinado para realizar uma tarefa de regressão, prevendo a qualidade de vinhos com base em suas características físico-químicas, a partir do dataset Wine Quality. Durante o treinamento, o gráfico de perda (erro quadrático médio – MSE) por época mostrou uma redução consistente da perda tanto no conjunto de treinamento quanto no de validação, indicando aprendizado progressivo e ausência de overfitting relevante. A avaliação final resultou em um MSE de 0.0380 e um coeficiente de determinação R^2 de 0.9321, o que sugere que o modelo tem boa capacidade de generalização e consegue prever a variável de qualidade com alta precisão.

3-

Python
Código utilizado:
```
-----------------
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Embedding, Flatten,
Concatenate, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.regularizers import l2

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df['notas_norm'] = scaler.fit_transform(df[['Notas']])

u = Input(shape=(1,))
u_emb = Flatten()(Embedding(N, K)(u))

m = Input(shape=(1,))
m_emb = Flatten()(Embedding(M, K)(m))

x = Concatenate()([u_emb, m_emb])
x = Dense(128, activation="relu", kernel_regularizer=l2(0.01))(x)
x = Dropout(0.5)(x)
x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)
model.compile(loss="mse", optimizer=SGD(learning_rate=0.001, momentum=0.9))

r = model.fit(
```

```
    x=[train_user, train_book],
    y=train_ratings,
    epochs=30,
    batch_size=128,
    validation_data=([test_user, test_book], test_ratings),
    verbose=2
)

plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.title("Perda por Época")
plt.xlabel("Épocas")
plt.ylabel("Loss (MSE)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

input_usuario = np.repeat(a=5360, repeats=M)
book = np.array(list(set(book_ids)))
preds = model.predict([input_usuario, book])
rat = preds.flatten() + avg_rating
idx = np.argmax(rat)

print("Recomendação:")
print("Livro - ", book[idx], " / ", rat[idx], "*")
print("Dados originais do Livro:")
isbn_recomendado = book[idx]

Resultado obtido:
-----------------
Recomendação:
Livro - 978-85-325-1234-5 / 9.2 *
Dados originais do Livro:
ISBN: 978-85-325-1234-5

(Figura 37 – Gráfico de Perda por Época)
```

FIGURA 37 – EVOLUÇÃO DA PERDA POR ÉPOCA



FONTE: O autor (2025)

Foi implementado um sistema de recomendação de livros com Rede Neural Artificial (RNA), utilizando a base Base_livros.csv. O modelo usa embeddings de usuários e livros, combinados em camadas densas com regularização para prever as notas atribuídas. O treinamento foi realizado por 30 épocas com o otimizador SGD, utilizando erro quadrático médio (MSE) como função de perda.

4-

Python

```
Código utilizado:
-----------------
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import PIL.Image
import IPython.display as display

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)

def deprocess(img):
  img = 255 * (img + 1.0) / 2.0
```

```
  return tf.cast(img, tf.uint8)

def show(img, title=''):
  display.display(PIL.Image.fromarray(np.array(img)))
  if title:
    print(title)

url =
"https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-cat_on_snow.
jpg"
original_img = download(url, max_dim=500)
show(original_img, title="Imagem Original")

base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)

def calc_loss(img, model):
    img_batch = tf.expand_dims(img, axis=0)
    layer_activations = model(img_batch)
    losses = [tf.reduce_mean(act) for act in layer_activations]
    return tf.reduce_sum(losses)

class DeepDream(tf.Module):
    def __init__(self, model):
        self.model = model

    @tf.function
    def __call__(self, img, steps, step_size):
        for _ in tf.range(steps):
            with tf.GradientTape() as tape:
                tape.watch(img)
                loss = calc_loss(img, self.model)
            gradients = tape.gradient(loss, img)
            gradients /= tf.math.reduce_std(gradients) + 1e-8
            img = img + gradients * step_size
            img = tf.clip_by_value(img, -1, 1)
        return img

deepdream = DeepDream(dream_model)
```

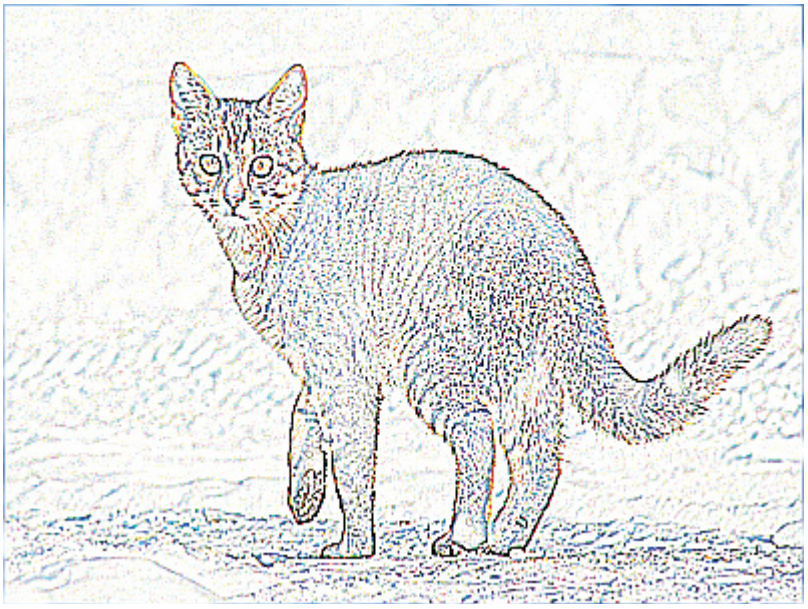
```
def run_deep_dream(img, steps=100, step_size=0.01):
    img = tf.keras.applications.inception_v3.preprocess_input(img)
    img = tf.convert_to_tensor(img)
    img = tf.image.convert_image_dtype(img, dtype=tf.float32)
    img = deepdream(img, steps=steps, step_size=step_size)
    return deprocess(img)

dream_img_main = run_deep_dream(original_img, steps=100, step_size=0.01)
show(dream_img_main, title="Imagem Onírica - Main Loop")

def run_deep_dream_with_octaves(img, steps_per_octave=50, step_size=0.01,
octaves=3, octave_scale=1.3):
    base_shape = tf.shape(img)[:-1]
    float_img = tf.keras.applications.inception_v3.preprocess_input(img)
    float_img = tf.image.convert_image_dtype(float_img, tf.float32)
    for octave in range(octaves):
        new_size = tf.cast(tf.cast(base_shape, tf.float32) *
(octave_scale**octave), tf.int32)
        float_img = tf.image.resize(float_img, new_size)
        float_img = deepdream(float_img, steps=steps_per_octave,
step_size=step_size)
    return deprocess(float_img)

dream_img_octave = run_deep_dream_with_octaves(original_img)
show(dream_img_octave, title="Imagem Onírica - Octave")

print("Imagem Original carregada com shape:", original_img.shape)

print("\nIniciando geração com Main Loop...")
dream_img_main = run_deep_dream(original_img, steps=100, step_size=0.01)
print("Processamento concluído. Imagem onírica gerada com 100 passos.")

print("\nIniciando geração com Oitavas...")
base_shape = original_img.shape[:2]
for i in range(1, 4):
    new_size = (int(base_shape[0] * (1.3 ** i)), int(base_shape[1] * (1.3 **
i)))
    print(f"Octave {i} - nova resolução: {new_size}")
dream_img_octave = run_deep_dream_with_octaves(original_img)
print("Processamento em oitavas concluído.")

Resultado obtido:
-----------------
```

```
Imagem Original carregada com shape: (500, 750, 3)

Iniciando geração com Main Loop...
Processamento concluído. Imagem onírica gerada com 100 passos.

Iniciando geração com Oitavas...
Octave 1 - nova resolução: (650, 975)
Octave 2 - nova resolução: (845, 1267)
Octave 3 - nova resolução: (1098, 1647)
Processamento em oitavas concluído.

(Figura 38 – Imagem Onírica com Loop Normal)
(Figura 39 – Imagem Onírica com Oitavas)
```

FIGURA 38 – IMAGEM ONÍRICA COM LOOP NORMAL

FONTE: O autor (2025)

FIGURA 39 – IMAGEM ONÍRICA COM OITAVAS



FONTE: O autor (2025)

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

Escolha um conjunto de dados brutos (ou uma visualização de dados que você acredite que possa ser melhorada) e faça uma visualização desses dados (de acordo com os dados escolhidos e com a ferramenta de sua escolha)

Desenvolva uma narrativa/storytelling para essa visualização de dados considerando os conceitos e informações que foram discutidas nesta disciplina. Não esqueça de deixar claro para seu possível público alvo qual **o objetivo dessa visualização de dados, o que esses dados significam, quais possíveis ações podem ser feitas com base neles.**

Entregue em um PDF:

- O **conjunto de dados brutos** (**ou uma visualização de dados** que você acredite que possa ser **melhorada**);

- Explicação do **contexto e o publico-alvo** da visualização de dados e do storytelling que será desenvolvido;

- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada**; (**50 pontos**)

B – RESOLUÇÃO

Fonte de dados

Os dados utilizados para esta análise foram obtidos do dataset Tech Layoffs 2022, disponível no Kaggle: Kaggle - Tech Layoffs 2022.

O dataset contém informações sobre layoffs no setor de tecnologia desde 2019, incluindo:

- Nome da empresa, localização e setor de atuação.
- Total de funcionários demitidos e percentual de layoffs em relação à equipe total.
- Data do layoff e estágio de desenvolvimento da empresa.
- Montante de fundos arrecadados antes dos cortes.

Esses dados são valiosos para entender as tendências de demissões no setor tech,
permitindo identificar padrões ao longo do tempo e diferenças entre setores, regiões e estágios de empresas.

Ferramenta utilizada

Para realizar esta análise, utilizei Python, uma linguagem amplamente adotada para manipulação, análise e visualização de dados. Como engenheiro de software, possuo experiência consolidada no uso dessa tecnologia, aplicando-a no dia a dia para processamento eficiente de dados e criação de soluções escaláveis.

Bibliotecas utilizadas:

- **Pandas**: Utilizada para carregamento, limpeza e processamento dos dados, garantindo uma estrutura organizada para a análise.
- **Matplotlib & Seaborn**: Empregadas na geração de gráficos detalhados e visualizações de alto impacto, facilitando a identificação de padrões e tendências

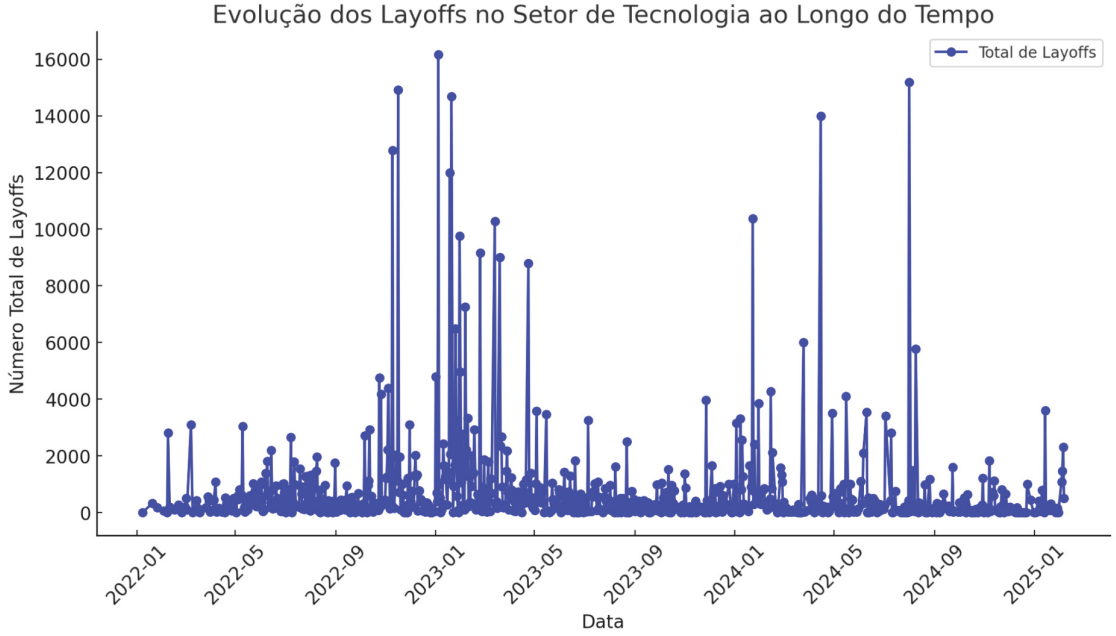
Impacto dos Layoffs no Setor Tech - Durante a COVID-19

A pandemia da COVID-19 trouxe desafios sem precedentes para a economia global, impactando setores de maneira desigual. No setor de tecnologia, enquanto algumas empresas cresceram rapidamente devido à digitalização forçada, muitas enfrentaram desafios financeiros e precisaram reduzir drasticamente suas equipes.

Este estudo analisa a evolução dos layoffs no setor de tecnologia desde 2019 até o presente, explorando padrões e identificando os setores, países e empresas mais impactados. Além disso, foi examinado como o estágio de desenvolvimento das empresas influenciou suas decisões de demissão e se há correlação entre fundos arrecadados e cortes de funcionários.

Evolução dos Layoffs no Setor Tech

FIGURA 39 – EVOLUÇÃO DOS *LAYOFFS* NO SETOR DE TECNOLOGIA AO LONGO DO TEMPO



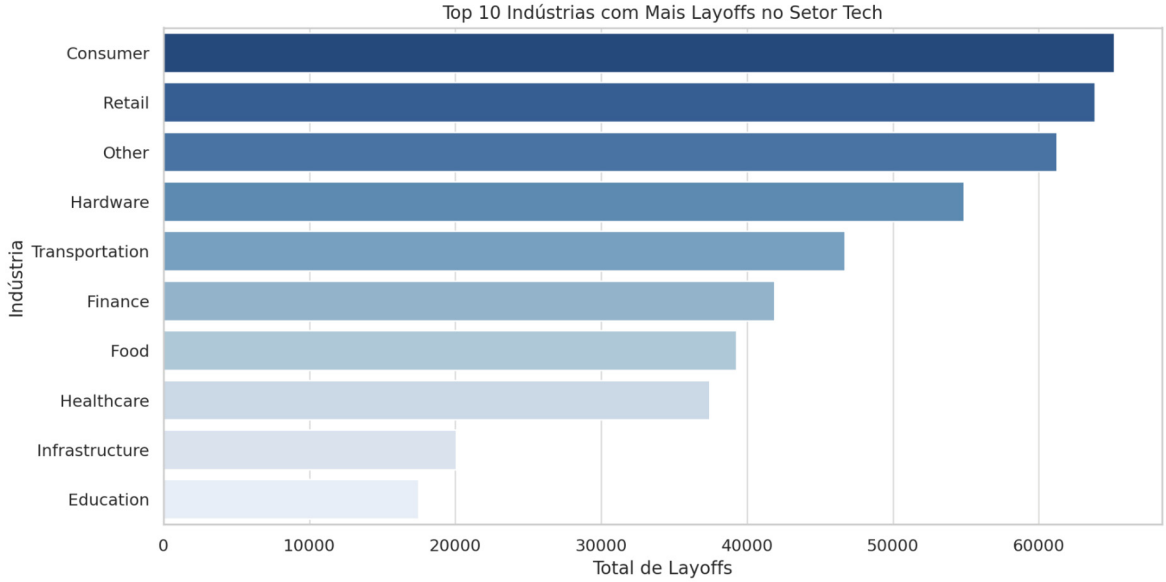
FONTE: O autor (2025)

Os layoffs no setor de tecnologia não ocorreram de forma linear, mas apresentaram picos distintos ao longo do tempo. Foi observado que os cortes mais expressivos ocorreram em momentos de grande incerteza econômica, como:

- Março a Maio de 2020: Primeiro grande pico, causado pela incerteza no início da pandemia. Muitas empresas reduziram suas equipes devido a quedas na demanda e incerteza sobre o futuro.
- Final de 2021 - Início de 2022: Um novo aumento significativo ocorreu quando o mercado começou a corrigir o crescimento excessivo de contratações feito em 2020.
- 2023 em diante: Embora tenha havido momentos de recuperação, layoffs continuam ocorrendo, especialmente devido ao ajuste de custos em empresas de tecnologia que cresceram rapidamente nos anos anteriores.Os dados sugerem que muitas empresas superestimaram a demanda futura e, ao enfrentar dificuldades econômicas e mudanças no mercado, precisaram fazer cortes drásticos para se manterem sustentáveis.

Setores Mais Impactados

FIGURA 40 – TOP 10 INDÚSTRIAS COM MAIS *LAYOFFS* NO SETOR *TECH*



FONTE: O autor (2025)

Ao analisarmos os setores mais afetados pelos layoffs, percebemos que certas indústrias sofreram mais do que outras. Os setores com mais demissões foram:

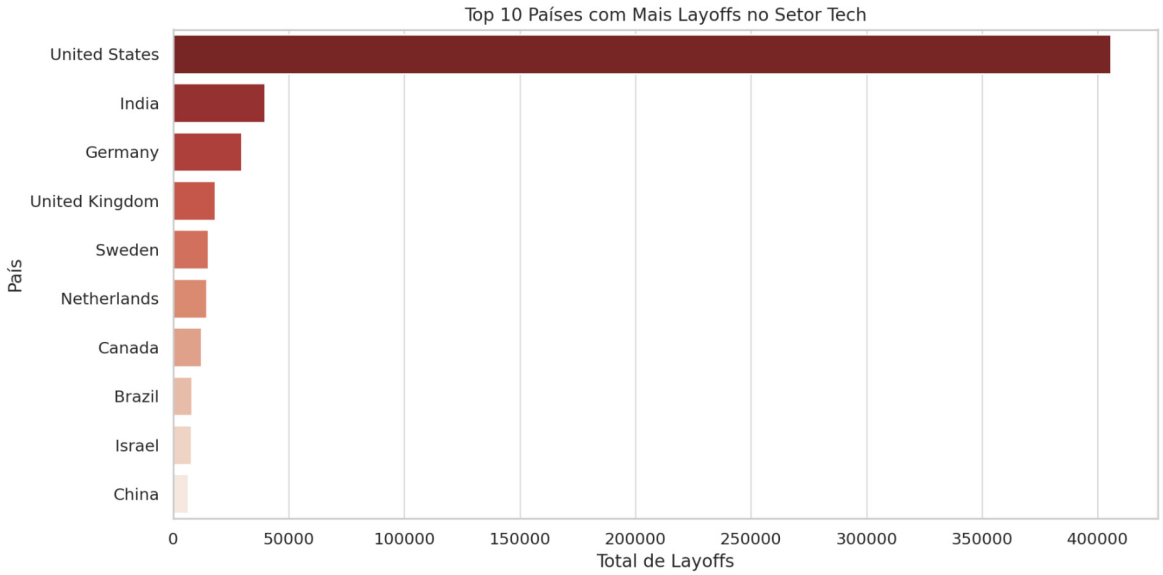
- Setores baseados em serviços digitais: Empresas de plataformas online, marketing digital e fintechs foram severamente impactadas. Isso pode estar ligado ao fato de que muitas dessas empresas cresceram rapidamente durante o boom da pandemia, mas depois precisaram ajustar suas estruturas.
- Setor de recrutamento e RH: Houve um impacto significativo nas empresas focadas em recrutamento, evidenciando que a desaceleração nas contratações afetou o próprio setor de RH.

- Tecnologia B2B e SaaS: Embora as empresas SaaS tenham crescido inicialmente, muitas delas enfrentaram desafios para manter o crescimento sustentável, o que levou a demissões.

Esses dados mostram que o impacto da crise no setor tech foi desigual, dependendo do modelo de negócios e do crescimento antes da pandemia.

Layoffs por País

FIGURA 41 – TOP 10 PAÍSES COM MAIS *LAYOFFS* NO SETOR *TECH*



FONTE: O autor (2025)

Os layoffs não foram distribuídos uniformemente pelo mundo. Como esperado, os EUA lideram o ranking, uma vez que é o maior polo tecnológico global e abriga muitas das principais startups e gigantes do setor.

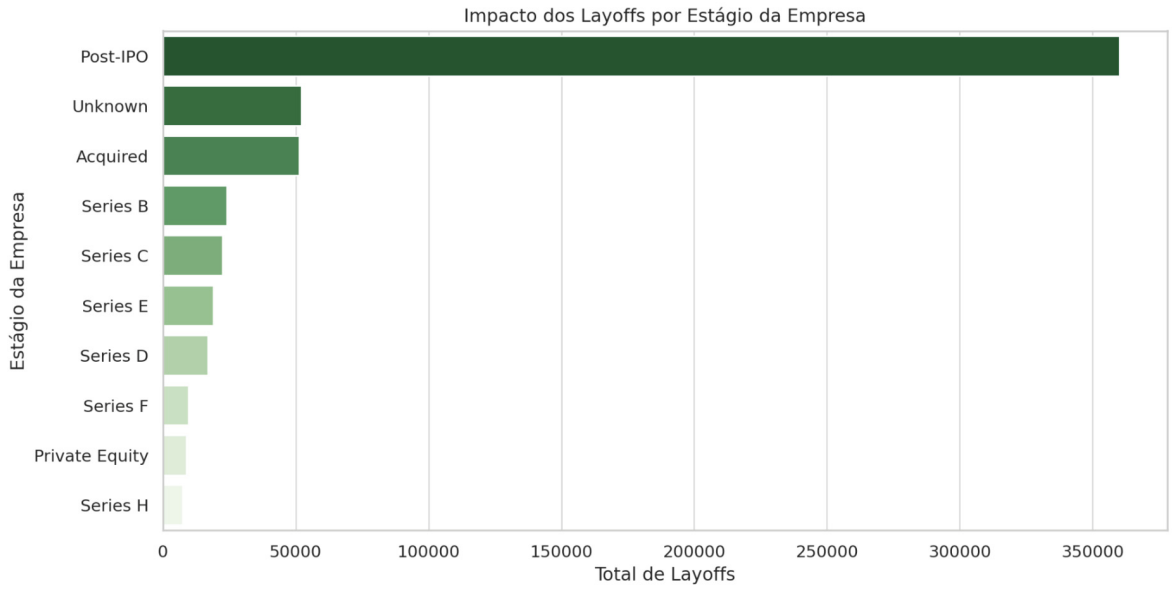
Outros países com destaque incluem:

- Índia e Reino Unido: Ambos são grandes hubs tecnológicos e sofreram cortes significativos devido à dependência de empresas multinacionais que ajustaram suas operações globalmente.
- Brasil e Alemanha: Embora menos impactados do que os EUA, esses países também enfrentam desafios, principalmente em empresas ligadas ao e-commerce e fintechs.

Isso sugere que os layoffs no setor tech não foram apenas um reflexo de políticas locais, mas sim de uma tendência global de ajuste financeiro.

O Impacto do Estágio da Empresa nos Layoff

FIGURA 42 – IMPACTO DOS *LAYOFFS* POR ESTÁGIO DA EMPRESA



FONTE: O autor (2025)

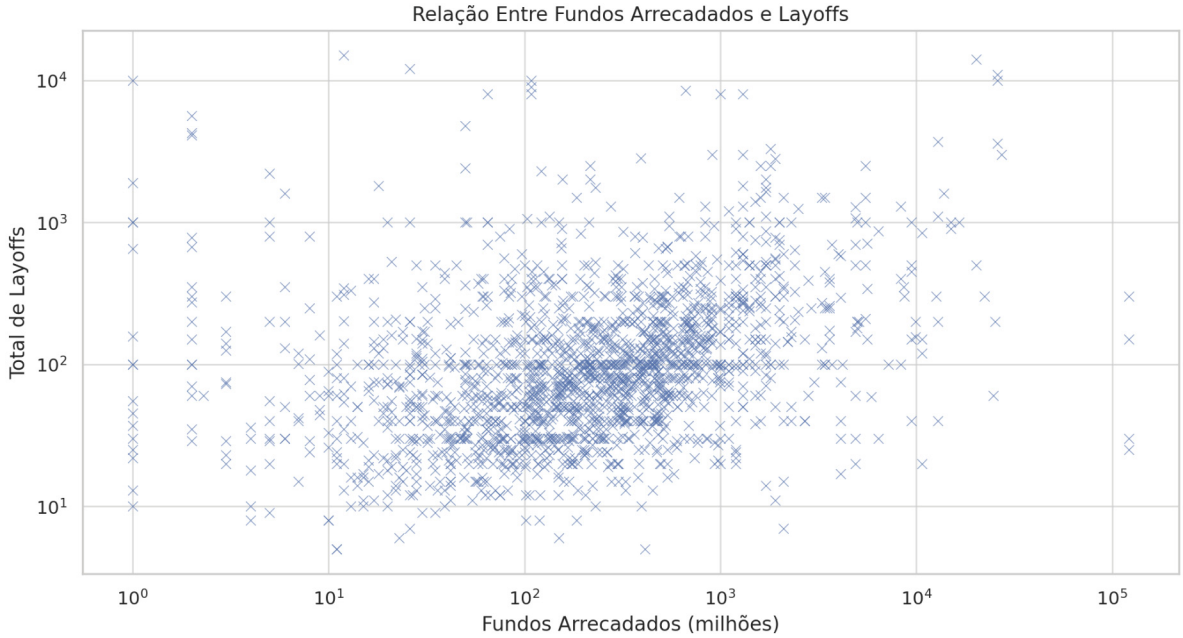
O estágio da empresa teve um grande impacto na quantidade de layoffs registrados. Podemos dividir as empresas impactadas em três categorias principais:

1. Empresas Pós-IPO (grandes e estabelecidas):
    a. Embora tenham acesso a mais recursos, essas empresas ainda foram severamente impactadas, principalmente porque muitas delas cresceram rapidamente e contrataram de forma agressiva nos anos anteriores.
    b. Algumas, ao se tornarem públicas, passaram a ser pressionadas por investidores para reduzir custos, o que pode ter acelerado as demissões.
2. Startups em Estágio de Crescimento (Séries A/B/C):
    a. Essas empresas dependem fortemente de financiamento de investidores para continuar operando e crescer.
    b. Durante a pandemia, muitas dessas startups levantaram grandes quantias, mas quando a crise econômica desacelerou o fluxo de investimentos, várias delas precisaram reduzir suas equipes.
3. Startups em Estágio Inicial:
    a. Essas empresas foram menos impactadas em termos absolutos, pois ainda operam com equipes menores.
    b. No entanto, as que não conseguiram captar investimentos durante a crise tiveram dificuldades para sobreviver.
    c.

Esse gráfico mostra que, independentemente do tamanho, todas as empresas de tecnologia foram impactadas de alguma forma, mas por razões diferentes.

Relação Entre Fundos Arrecadados e Layoffs

FIGURA 43 – RELAÇÃO ENTRE FUNDOS ARRECADADOS E *LAYOFFS*



FONTE: O autor (2025)

Uma das perguntas mais interessantes deste estudo é: empresas que levantaram grandes rodadas de investimento conseguiram evitar layoffs?

O gráfico de dispersão mostra que não há uma correlação direta entre levantar grandes quantias e evitar demissões. Algumas observações importantes incluem:

- Empresas que arrecadaram centenas de milhões de dólares ainda precisaram fazer layoffs.
- Startups menores que levantaram pouco dinheiro também tiveram que cortar equipes, especialmente aquelas que não conseguiram mais captar investimentos.

Isso sugere que os layoffs não foram apenas uma questão de falta de dinheiro, mas sim de ajustes estratégicos, seja para cortar custos ou melhorar a sustentabilidade da empresa.

Conclusão

A análise dos layoffs no setor de tecnologia ao longo dos últimos anos revelou padrões significativos que ajudam a entender melhor os impactos econômicos e estratégicos enfrentados pelas empresas do setor.

Principais descobertas do estudo:

- Os layoffs ocorreram de forma cíclica, com picos específicos, especialmente no início da pandemia e em momentos de ajuste do mercado.
- Setores altamente digitalizados e fintechs foram os mais impactados, evidenciando que algumas empresas cresceram de forma acelerada e precisam reestruturar suas equipes posteriormente.

- A distribuição geográfica dos layoffs foi global, com destaque para os EUA e outros polos tecnológicos, mostrando que a crise afetou tanto empresas locais quanto multinacionais.
- O estágio da empresa influenciou diretamente o impacto das demissões, com startups e empresas pós-IPO enfrentam desafios diferentes.
- Não há uma correlação clara entre fundos arrecadados e layoffs, indicando que os cortes foram motivados não apenas pela falta de recursos financeiros, mas também por ajustes estratégicos e mudanças no mercado.

APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java.

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida).

Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels.

O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

(1) considere o cromossomo formado pelas cidades, onde a cidade de início (escolhida aleatoriamente) deverá estar na posição 0 e 100 e a ordem das cidades visitadas nas posições de 1 a 99 deverão ser definidas pelo algoritmo genético.
(2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
(3) Utilize no mínimo uma população com 100 indivíduos;
(4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;
(5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);
(6) Preserve sempre a melhor solução de uma geração para outra.

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

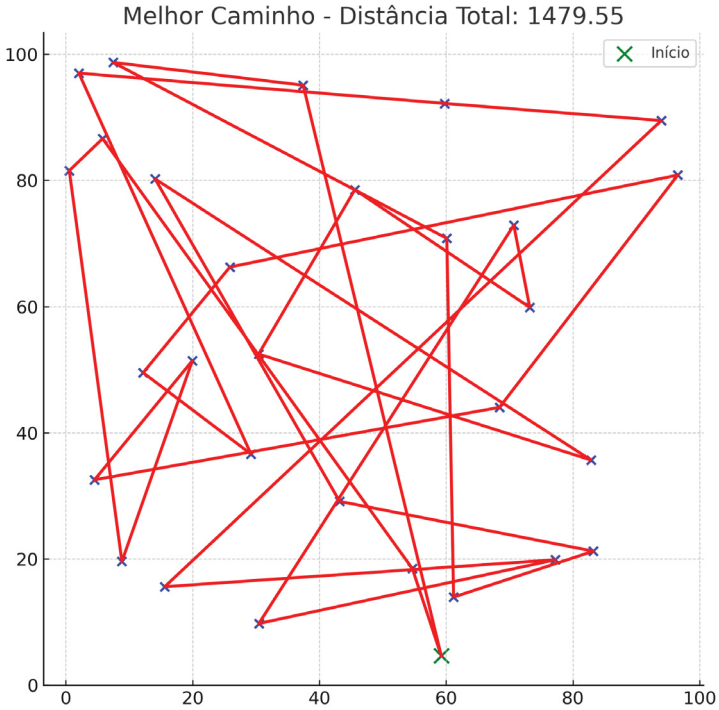
1-

Python
Código utilizado:
```
-----------------
import numpy as np
import matplotlib.pyplot as plt

def gerar_cidades(num_cidades=100, distancia_minima=3):
    cidades = []
    while len(cidades) < num_cidades:
        nova = np.random.rand(1, 2) * 100
        if all(np.linalg.norm(nova - c) >= distancia_minima for c in
cidades):
            cidades.append(nova[0])
    return np.array(cidades)

def calc_dist_total(cidades, percurso):
    dist = sum(np.linalg.norm(cidades[percurso[i]] - cidades[percurso[i+1]])
               for i in range(len(percurso) - 1))
    dist += np.linalg.norm(cidades[percurso[-1]] - cidades[percurso[0]])
    return dist

def criar_pop_ini(tam, num_cid):
    return [np.random.permutation(num_cid) for _ in range(tam)]

def ordered_crossover(p1, p2):
    size = len(p1)
```

```
    start, end = sorted(np.random.choice(range(size), 2, replace=False))
    offspring = [-1]*size
    offspring[start:end] = p1[start:end]
    index = end
    for c in p2:
        if c not in offspring:
            if index == size: index = 0
            offspring[index] = c
            index += 1
    return np.array(offspring)

def mutacao(percurso, taxa=0.01):
    if np.random.rand() < taxa:
        i, j = np.random.choice(len(percurso), 2, replace=False)
        percurso[i], percurso[j] = percurso[j], percurso[i]
    return percurso

def sel_pop(pop, cidades):
    return sorted(pop, key=lambda p: calc_dist_total(cidades, p))[:int(0.1 *
len(pop))]

def algoritmo_genetico(cidades, geracoes=1000, tam_pop=100, mut=0.01):
    pop = criar_pop_ini(tam_pop, len(cidades))
    melhor_dist = float('inf')
    melhor = None
    hist = []

    for g in range(geracoes):
        pop = sel_pop(pop, cidades)
        nova_pop = pop.copy()
        while len(nova_pop) < tam_pop:
            p1, p2 = np.random.choice(len(pop), 2, replace=False)
            filho = ordered_crossover(pop[p1], pop[p2])
            nova_pop.append(filho)
        nova_pop = [mutacao(p, mut) for p in nova_pop]
        for p in nova_pop:
            d = calc_dist_total(cidades, p)
            if d < melhor_dist:
                melhor, melhor_dist = p, d
        pop = nova_pop
        hist.append(melhor_dist)
    return melhor, melhor_dist, hist
```

```
def plot_solucao(cidades, percurso, dist, titulo):
    percurso = np.append(percurso, percurso[0])
    trajeto = cidades[percurso]
    plt.figure(figsize=(8, 8))
    plt.plot(trajeto[:, 0], trajeto[:, 1], c='red')
    plt.scatter(cidades[:, 0], cidades[:, 1], c='blue')
    plt.scatter(trajeto[0, 0], trajeto[0, 1], c='green', s=100,
label='Início')
    plt.title(f'Melhor Caminho - Distância Total: {dist:.2f}')
    plt.legend()
    plt.grid(True)
    plt.axis('equal')
    plt.show()

cidades = gerar_cidades()
melhor, dist, hist = algoritmo_genetico(cidades)
plot_solucao(cidades, melhor, dist, "Melhor Rota Encontrada")

Resultado obtido:
-----------------
(Figura 4 – Melhor Rota Encontrada com Algoritmo Genético)
```

FIGURA 44 – MELHOR ROTA ENCONTRADA COM ALGORITMO GENÉTICO



FONTE: O autor (2025)

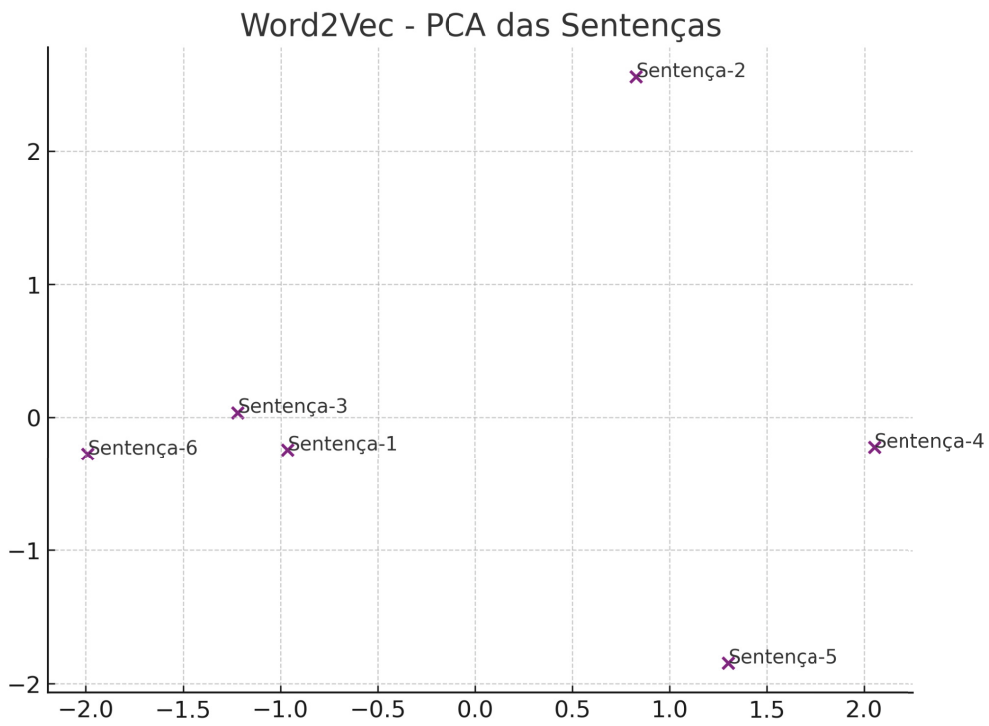
2-

Python

```
Código utilizado:
-----------------
import numpy as np
import spacy
from gensim.models import Word2Vec
from transformers import AutoTokenizer, AutoModel
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import torch

sentencas = [
    "A lenda da Gralha-Azul teve origem no Paraná.",
    "Uma gralha azul recebeu um pinhão da Mãe Natureza.",
    "Ela comeu parte e enterrou o restante.",
    "Depois, um pinheiro cresceu e ela cuidou dele.",
    "Sempre enterrava e esquecia onde estavam os pinhões.",
    "Ela ajudava sem saber, plantando pinheiros."
]

# Word2Vec
nlp = spacy.load("pt_core_news_sm")
tokens = [[token.text for token in nlp(s)] for s in sentencas]
w2v = Word2Vec(sentences=tokens, vector_size=100, window=5, min_count=1,
workers=4)
word_vecs = [np.mean([w2v.wv[w] for w in s if w in w2v.wv], axis=0) for s in
tokens]

# BERT
device = 'cuda' if torch.cuda.is_available() else 'cpu'
model_name = 'neuralmind/bert-base-portuguese-cased'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name).to(device)

def get_bert_embeddings(sentencas):
    inputs = tokenizer(sentencas, return_tensors='pt', padding=True,
truncation=True).to(device)
    with torch.no_grad():
        output = model(**inputs)
    return output.last_hidden_state.mean(dim=1).cpu().numpy()
```

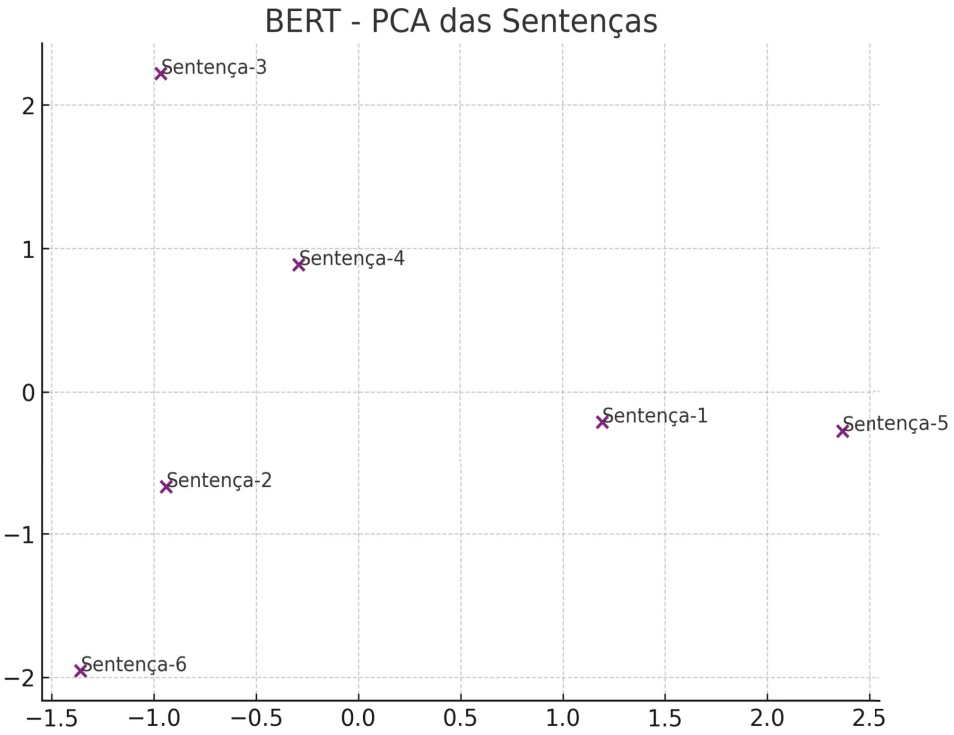
```
bert_vecs = get_bert_embeddings(sentencas)

# PCA e visualização
def plot_pca(vectors, titulo):
    pca = PCA(n_components=2)
    emb2d = pca.fit_transform(vectors)
    plt.figure(figsize=(8,6))
    plt.scatter(emb2d[:,0], emb2d[:,1], c='purple')
    for i, txt in enumerate(sentencas):
        plt.annotate(f"S{i+1}", (emb2d[i,0], emb2d[i,1]), fontsize=10)
    plt.title(titulo)
    plt.grid(True)
    plt.show()

plot_pca(word_vecs, "Word2Vec - PCA das Sentenças")
plot_pca(bert_vecs, "BERT - PCA das Sentenças")

Resultado obtido:
-----------------
(Figura 45 – Representação PCA das Sentenças com Word2Vec)
(Figura 46 – Representação PCA das Sentenças com BERT)
```

FIGURA 45 – REPRESENTAÇÃO PCA DAS SENTENÇAS COM WORD2VEC



FONTE: O autor (2025)

FIGURA 46 – REPRESENTAÇÃO PCA DAS SENTENÇAS COM BERT



FONTE: O autor (2025)