UNIVERSIDADE FEDERAL DO PARANÁ

ADRIANO WIERZBICKI

MEMORIAL DE PROJETOS: TOMADA DE DECISÃO ESTRATÉGICA COM STORYTELLING

CURITIBA

2025

ADRIANO WIERZBICKI

MEMORIAL DE PROJETOS: TOMADA DE DECISÃO ESTRATÉGICA COM STORYTELLING

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

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

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 **ADRIANO WIERZBICKI**, intitulada: **MEMORIAL DE PROJETOS: TOMADA DE DECISÃO ESTRATÉGICA COM STORYTELLING**, 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, 17 de Outubro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

A crescente disponibilidade de dados exige recursos que tornem as informações compreensíveis e úteis. A Visualização de Dados desempenha papel central ao transformar grandes volumes de dados brutos em representações gráficas claras, facilitando a identificação de padrões, tendências e correlações. Quando combinado ao *Storytelling*, essa prática permite criar narrativas envolventes que não apenas informam, mas também engajam diferentes públicos, apoiando a tomada de decisão em contextos corporativos, científicos e jornalísticos. Os principais tipos de gráficos, como barras, linhas, histogramas, dispersão, mapas de calor e visualizações geoespaciais, devem ser escolhidos de acordo com o objetivo da análise e a natureza dos dados. O *design* visual é crucial para garantir clareza e acessibilidade, com atenção a cores, tipografia, hierarquia e consistência. O *Storytelling* de dados estrutura narrativas a partir de informações, equilibrando rigor analítico e recursos narrativos para comunicar resultados de forma impactante. Ferramentas oferecem suporte para criar visualizações interativas e personalizadas. Novas técnicas, como redução de dimensionalidade, realidade aumentada e visualizações interativas, ampliam as possibilidades de análise, mas exigem equilíbrio entre inovação e simplicidade. Assim, a integração entre visualização de dados e *Storytelling* vai além da estética, pois constitui uma estratégia essencial para transformar dados em histórias relevantes, acessíveis e capazes de inspirar ações.

Palavras-chave: contação de história; visualização de dados; visualizações geoespaciais; visualizações interetivas; narrativas.

ABSTRACT

The growing availability of data demands resources that make information both understandable and useful. Data Visualization plays a central role by transforming large volumes of raw data into clear graphical representations, making it easier to identify patterns, trends, and correlations. When combined with Storytelling, this practice enables the creation of engaging narratives that not only inform but also captivate diverse audiences, supporting decision-making in corporate, scientific, and journalistic contexts. The main types of charts—such as bar and line graphs, histograms, scatter plots, heatmaps, and geospatial visualizations—should be selected according to the purpose of the analysis and the nature of the data. Visual design is crucial to ensure clarity and accessibility, with careful attention to colors, typography, visual hierarchy, and consistency. Data Storytelling structures narratives based on information, balancing analytical rigor with narrative techniques to communicate results effectively. Various tools provide support for building interactive and customized visualizations. Emerging techniques, such as dimensionality reduction, augmented reality, and interactive visualizations, expand analytical possibilities but require a balance between technological innovation and communicative simplicity. Thus, the integration of Data Visualization and Storytelling goes beyond aesthetics, representing an essential strategy to transform data into meaningful, accessible stories that inspire action.

Keywords: Storytelling; data visualization; geospatial visualizations; interactive visualizations; narratives.

SUMÁRIO

1 PARECER TÉCNICO

A crescente disponibilidade de dados em diferentes áreas do conhecimento e do mercado, trouxe à tona a necessidade de ferramentas e técnicas que tornem essas informações compreensíveis e úteis. Healy (2018) destaca que o contexto de visualização de dados é essencial para análise, comunicação e compreensão, pois surge como um recurso fundamental que transforma grandes volumes de dados brutos em representações gráficas de forma mais claras, favorecendo a identificação de padrões, correlações e tendências.

De acordo com Knaflic (2019), a aplicação de *Storytelling* sobre a visualização de dados tem ganhado destaque por possibilitar a construção de narrativas envolventes que conectam dados, contexto e público. Essa combinação não apenas informa, mas também engaja, tornando a análise mais acessível e significativa para diferentes perfis de usuários, desde gestores corporativos até pesquisadores e jornalistas.

Este trabalho tem como objetivo destacar a importância da visualização de dados e do *Storytelling*, apresentar os principais tipos de gráficos e seus usos, discutir elementos de design que fortalecem a comunicação visual e analisar o papel das ferramentas tecnológicas no apoio a essas práticas.

A visualização de dados é essencial para a compreensão de fenômenos complexos, permitindo a transformação de informações numéricas em insights estratégicos. Ela facilita tanto a análise exploratória, voltada à identificação de padrões ocultos, quanto a análise explicativa, focada na comunicação clara de resultados. Em áreas como *Business Intelligence* (BI), a visualização atua como suporte à tomada de decisão, auxiliando as organizações a reagirem de forma ágil às mudanças de mercado. De acordo com Healy (2018), os gráficos desempenham um papel fundamental, pois tornam os dados mais acessíveis e compreensíveis. A escolha do tipo de gráfico mais adequado, pois gráficos de barras, linhas, pizza, dispersão ou mapas, depende diretamente do objetivo da análise e da natureza dos dados, sendo crucial para garantir uma interpretação eficaz e assertiva das informações apresentadas. Alguns exemplos incluem como:

- Gráficos de barras e colunas para comparar categorias;
- Gráficos de linhas para observar evolução temporal;
- Histogramas para analisar distribuições de frequência;

- Gráficos de dispersão para identificar correlações entre variáveis;
- Mapas de calor para destacar concentrações ou intensidade;
- Visualizações geoespaciais para representar fenômenos relacionados à localização;

O uso inadequado de gráficos pode induzir a interpretações equivocadas, reforçando a necessidade de conhecimento técnico e boas práticas de *design*.

Spence (2007) destaca que o *design* visual é um aspecto crucial para a clareza das representações. Elementos como cores, tipografia, hierarquia visual e consistência influenciam diretamente na interpretação dos dados, como o uso excessivo de cores pode confundir o usuário, enquanto a escolha correta pode guiar sua atenção para os pontos mais relevantes. Além disso, a acessibilidade deve ser considerada, garantindo que pessoas com limitações visuais também compreendam a informação.

Segundo Knaflic (2019) o *Storytelling* de dados é uma abordagem que combina narrativas com visualizações para construir histórias convincentes e memoráveis. As narrativas eficazes possuem elementos como personagem, enredo, conflito e resolução, que podem ser adaptados para o contexto dos dados.

Ao estruturar uma história em torno dos dados, deve-se seguir uma sequência lógica (começo, meio e fim), equilibrando rigor analítico com recursos narrativos. Isso é particularmente relevante em contextos corporativos, nos quais dados precisam não apenas ser mostrados, mas também justificativa de decisões estratégicas.

Existem diversas ferramentas para criação de visualizações e narrativas interativas tais como:

- Bibliotecas Python: Matplotlib, Seaborn, Plotly, Bokeh. Bruce e Bruce (2020) demonstram que essas ferramentas ajudam na criação de gráficos interativos e análises visuais claras. São ferramentas voltadas para cientistas de dados, com aplicações práticas em S*torytelling* técnico;
- Plataformas de BI: Tableau, Power BI, Qlik. Sharda, Delen e Turban (2019) sinalizam da importância dessas ferramentas na visualização e comunicação de dados para suporte à decisão gerencial. Apontam como as ferramentas possibilitam dashboards e relatórios interativos e que são base de narrativas empresariais;

A escolha da ferramenta deve considerar fatores como complexidade do projeto, público-alvo e competências técnicas da equipe. Com a evolução dos dados multidimensionais, surgem desafios adicionais de representação, sendo que, entre as técnicas utilizadas, destacam-se:

- Redução de dimensionalidade (PCA, t-SNE). Principal Component Analysis (PCA) é uma técnica para reduzir a dimensionalidade dos dados enquanto preserva as relações entre as observações e t-distributed Stochastic Neighbor Embedding (t-SNE) é uma técnica de redução de dimensionalidade amplamente usada para visualização de dados de alta dimensão em 2D ou 3D. Bruce e Bruce sinalizam sobre técnicas de redução de dimensionalidade, pois é necessário a escolha de uma técnica baseada na complexidade do projeto;
- Visualizações interativas com filtros e animações. Spence (2007) informa a importância de considerar o público e o propósito ao selecionar a abordagem;
- Realidade aumentada e virtual para exploração imersiva. Spence (2007) informa a importância sobre os filtros, animações, técnicas de navegação visual;

Esses recursos possibilitam uma maior exploração e compreensão das informações, mas exigem um equilíbrio delicado entre a inovação tecnológica e a simplicidade comunicativa, garantindo que os dados continuem acessíveis e significativos para diferentes públicos. Quando a visualização de dados é aliada ao *Storytelling*, ela vai além de uma apresentação de números e gráficos, pois transforma dados brutos em narrativas envolventes, contextualizadas e relevantes. Essa combinação torna-se uma poderosa ferramenta para engajar audiências, facilitar o entendimento de insights complexos e direcionar ações estratégicas. Em um cenário cada vez mais orientado por dados, essa integração não é apenas desejável, mas essencial nos ambientes corporativos e institucionais onde a tomada de decisão baseada em dados representa um importante diferencial competitivo. Ao unir a força da visualização com a arte de contar histórias, é possível comunicar descobertas com mais clareza, emoção e impacto.

A clareza gráfica, o uso de boas práticas de design e a escolha consciente das ferramentas reforçam a eficácia da comunicação visual. Da mesma forma, a

incorporação de narrativas amplia o impacto das análises, assim favorecendo a compreensão e engajamento do público.

Portanto, a prática da visualização de dados com *Storytelling* não deve ser entendida apenas como recurso estético, mas como uma estratégia essencial para organizações, pesquisadores e profissionais que buscam dar sentido aos dados e inspirar ações a partir deles.

REFERÊNCIAS

BRUCE, Peter; BRUCE, Andrew. **Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python**. 2a. Ed. United States of America: O'Reilly Media. 2020. ISBN 978-1492072942.

HEALY, Kieran. **Data Visualization: A Practical Introduction**. United States of America: Princeton University Press. 2018. ISBN 978-0691181622.

KNAFLIC, Cole Nussbaumer. Storytelling com dados: **Um Guia Sobre Visualização de Dados Para Profissionais de Negócios**. 2a. Ed. Rio de Janeiro: Alta Books. 2019. ISBN 9788550804682.

SHARDA, Ramesh; DELEN, Dursun; TURBAN, Efrain. **Business Intelligence e Análise de Dados para Gestão do Negócio**. 4a. Ed. Porto Alegre: Bookman. 2019. ISBN 9788582605196.

SPENCE, Robert. **Information visualization: Design for Interaction**. 2a Ed. Harlow, England: Pearson Prentice-Hall. 2007. ISBN 9780132065504.

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.

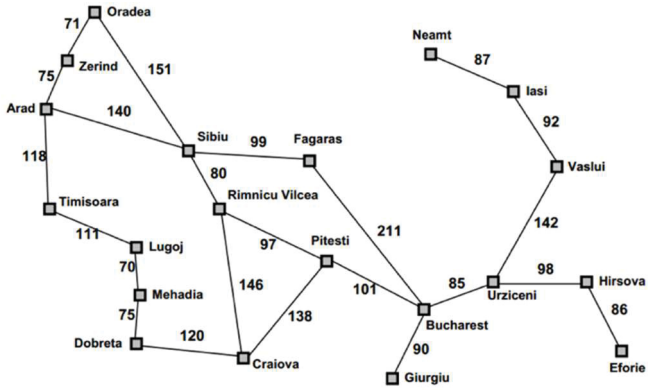
2 Busca Heurística

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.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

Se as uvas caem, então a raposa as come
Se a raposa as come, então estão maduras
As uvas estão verdes ou caem

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem
q: a raposa come as uvas
r: as uvas estão maduras

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

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

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

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

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

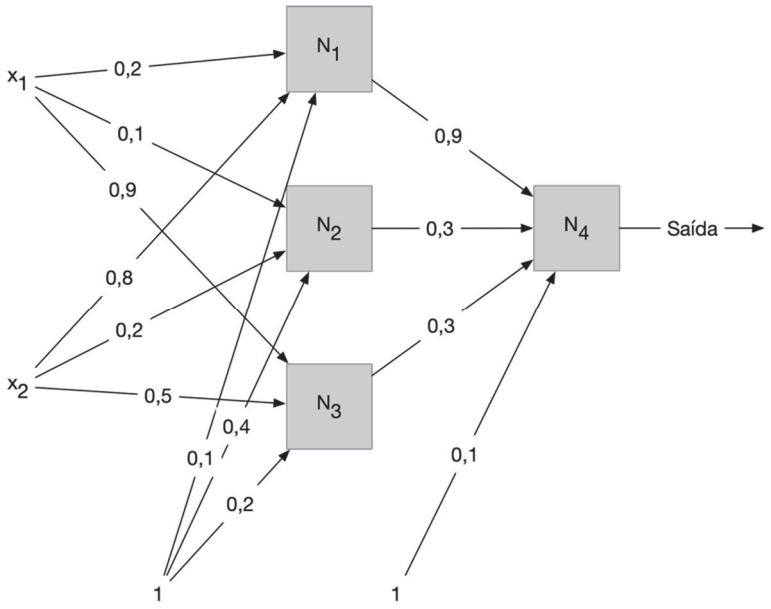
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

1 ChatGPT

a) Pergunte ao ChatGPT o que é Inteligência Artificial e cole aqui o resultado.

A Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas que podem realizar tarefas que normalmente exigiriam inteligência humana. Essas tarefas incluem coisas como reconhecimento de padrões, aprendizado, tomada de decisões, processamento de linguagem natural e muito mais.

Os sistemas de IA são frequentemente projetados para imitar certos aspectos do pensamento humano, como raciocínio, aprendizado e resolução de problemas. Eles podem ser classificados em duas categorias principais: IA fraca (também conhecida como estreita ou específica), que se concentra em tarefas específicas, e IA forte (também conhecida como geral ou forte), que visa replicar a inteligência humana em sua totalidade.

Existem várias técnicas e abordagens dentro do campo da IA, incluindo redes neurais artificiais, algoritmos de aprendizado de máquina, lógica simbólica, algoritmos genéticos e muito mais. A IA tem aplicações em uma ampla gama de setores, incluindo saúde, finanças, manufatura, transporte, entretenimento e muitos outros.

b) Dada essa resposta do ChatGPT, classifique usando as 4 abordagens vistas em sala. Explique o porquê.

Pensar como humanos. Porque é voltado par o objetivo de implementar fielmente o processo de objetivo humano (que se concentra no desenvolvimento de sistemas e máquinas capazes de realizar tarefas que normalmente exigiriam inteligência humana). E, pelo fato de expor pontos, que são referentes ao processo de mapeamento do pensamento se tratando mais de habilidades cognitivas (IA visa criar sistemas que possam aprender, raciocinar, perceber e interagir de forma semelhante aos seres humanos).

c) Pesquise sobre o funcionamento do ChatGPT (sem perguntar ao próprio ChatGPT) e escreva um texto contendo no máximo 5 parágrafos. Cite as referências.

Criada pela empresa OpenAI, na sede de São Francisco, o CHATGPT (Generative Pre-Trained Transformer – Transformador Pré-treinado generativo) é um modelo de inteligência artificial, projetado para responder questões passadas através de seu prompt de uma forma mais 'Humana' e relevante para o utilizador. Sendo de questões mais simples, a algo mais complexo.

A arquitetura dessa ferramenta é baseada em Rede Neural Artificiais (RNAs), com o foco em diálogo virtual. Utiliza (PPO) que é um algoritmo de aprendizado de máquina por reforço, onde inicialmente foi realizado um treinamento com uma grande quantidade de dados, então suas respostas

são realizadas através da coleta de informação e vai adquirindo mais conhecimento de acordo os feedbacks humanos. O mecanismo não tem a capacidade de buscar informações em tempo real na internet, o que acaba o deixando passível de erro.

O CHATGPT, também utiliza (NLP) que é o processamento de linguagem natural onde avalia o contexto, busca as informações disponíveis e retorna de uma forma que seja de uma forma natural e coerente. Isso, faz com que as respostas sejam diferentes para cada usuário.

O uso dessa ferramenta tende a ser aliada em várias áreas como, produção de conteúdo, análise de dados, pesquisa, construção de texto, tradução automática, programação, entre outros. Possui uma versão gratuita, contando também com um plano pago que oferta benefícios para que os utiliza.

Referências:

https://www.tray.com.br/escola/chat-gpt/
https://pluga.co/blog/chat-gpt/
https://botpress.com/pt/blog/how-does-chatgpt-work
https://joaquimfantin.com/chatgpt-funcionamentoutilidade/#:~:text=Como%20funciona&text=Quando%20um%20usuário%20faz%20uma,para%20o%20contexto%20da%20conversa

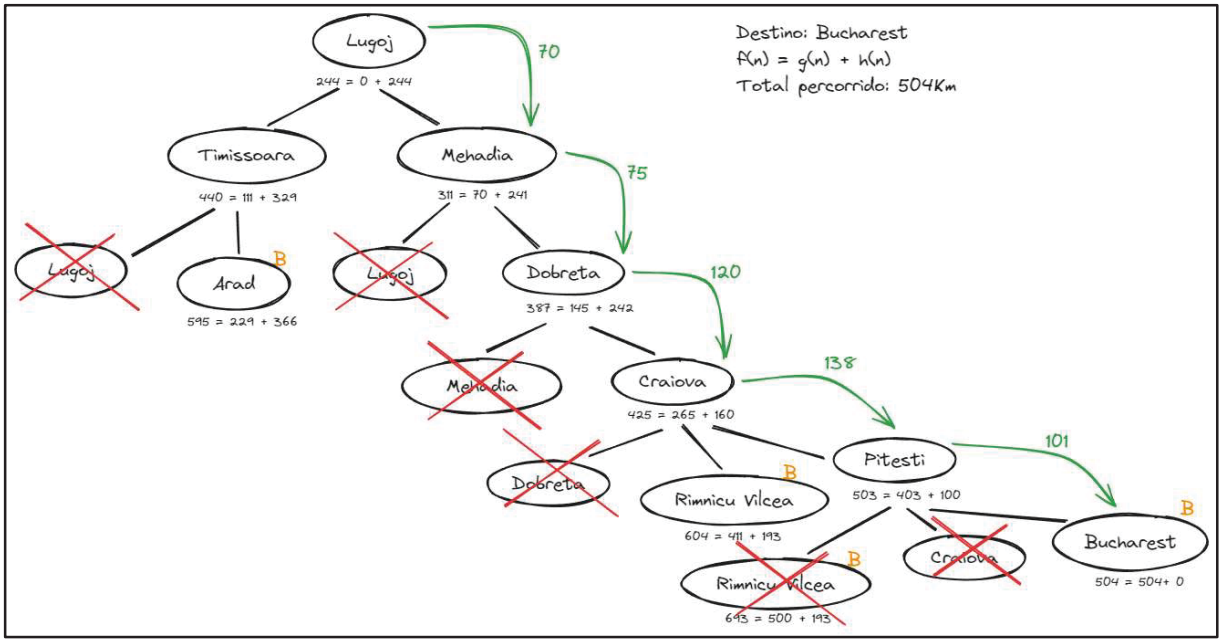
d) Entendendo o que é o ChatGPT, classifique o próprio ChatGPT usando as 4 abordagens vistas em sala. Explique o porquê.

Abordagem, agir como humano. O ChatGPT tenta 'Imitar o comportamento humano, utilizando uma linguagem natural, fazendo o utilizador se sentir como se estivesse em uma conversa com outro humano.

2 Busca Heurística

a)

IMAGEM 1 – DESENVOLVIMENTO DO EXERCÍCIO



FONTE: O autor (ano)

3 Lógica

a)

R1: P → Q

R2: Q → R

R3: ¬R V P

R4: R → P EI, R3

R5: Q → P SI, R2, R4

R6: (Q → P) ∧ (P →Q) CONJ, R5, R1

R7: (Q ↔ P) BICOND, R6

4 Redes Neurais Artificiais

a)

N1 -> =1*0,1+(-3)*0,2+1*0,8

N1 -> 0,3

b)

N2 -> =1*0,4+(-3)*0,1+1*0,2

N2 -> 0,3

c)

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

N3 -> -2

d)

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

N4 -> -0,14

tanh(u) -> -0,1391

F(a)(u) = -0,13909

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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 Análise Exploratória dos dados

a)

```
## Carregando a base de dados media_preco_carros_brasil.csv
dados=pd.read_csv('precos_carros_brasil.csv')
```

b)

```
## Verificando se existem valores faltantes e a quantidade por coluna
dados.isna().sum()
```

FIGURA 3 – OUTPUT COLUNAS FALTANTES

year_of_reference	65245
month_of_reference	65245
fipe_code	65245
authentication	65245
brand	65245
model	65245
fuel	65245
gear	65245
engine_size	65245
year_model	65245
avg_price_brl	65245

FONTE: O autor (2025)

c)

```
##Verificando se há dados duplicados nos dados
dados.duplicated().sum()
```

d)

```
##Criando duas categorias, para separar colunas numéricas e categóricas
Colunas_Numericas= [col for col in dados.columns if dados[col].dtype !=
'object']
colunas_categoricas= [col for col in dados.columns if dados[col].dtype ==
'object']
```

```
#Imprimindo o resultado das informações
```

```
dados[Colunas_Numericas].describe()
```

FIGURA 4 – OUTPUT RESUMO VARIÁVEIS NUMÉRICAS

	# year_of_reference	# year_model	# avg_price_brl
count	202295.0	202295.0	202295.0
mean	2021.5646951234583	2011.2715143725748	52756.7657134383
std	0.5719035995756846	6.376240631314859	51628.91211556084
min	2021.0	2000.0	6647.0
25%	2021.0	2006.0	22855.0
50%	2022.0	2012.0	38027.0
75%	2022.0	2016.0	64064.0
max	2023.0	2023.0	979358.0

FONTE: O autor (2025)

```
#Imprima o resumo de informações das variáveis categóricas
dados[colunas_categoricas].describe()
```

FIGURA 5 – OUTPUT RESUMO VARIÁVEIS CATEGÓRICAS

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

FONTE: O autor (2025)

e)

```
#Imprima a contagem de valores por modelo (model) e marca do carro (brand)
dados['model'].value_counts()
```

FIGURA 6 – OUTPUT TOP 10 CONTAGEM DE VALORES

	# model
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	425
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	400
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	400
Corvette 5.7/ 6.0, 6.2 Targa/Stingray	375
Golf 2.0/ 2.0 Mi Flex Aut/Tiptronic.	375
Doblo Adv/Adv TRYON/LOCKER 1.8 Flex	375
Kombi Escolar 1.6 MPi	350
Courier 1.6 L/ 1.6 Flex	350
Courier XL/XL-RS 1.6/ XL 1.6 Flex	350

FONTE: O autor (2025)

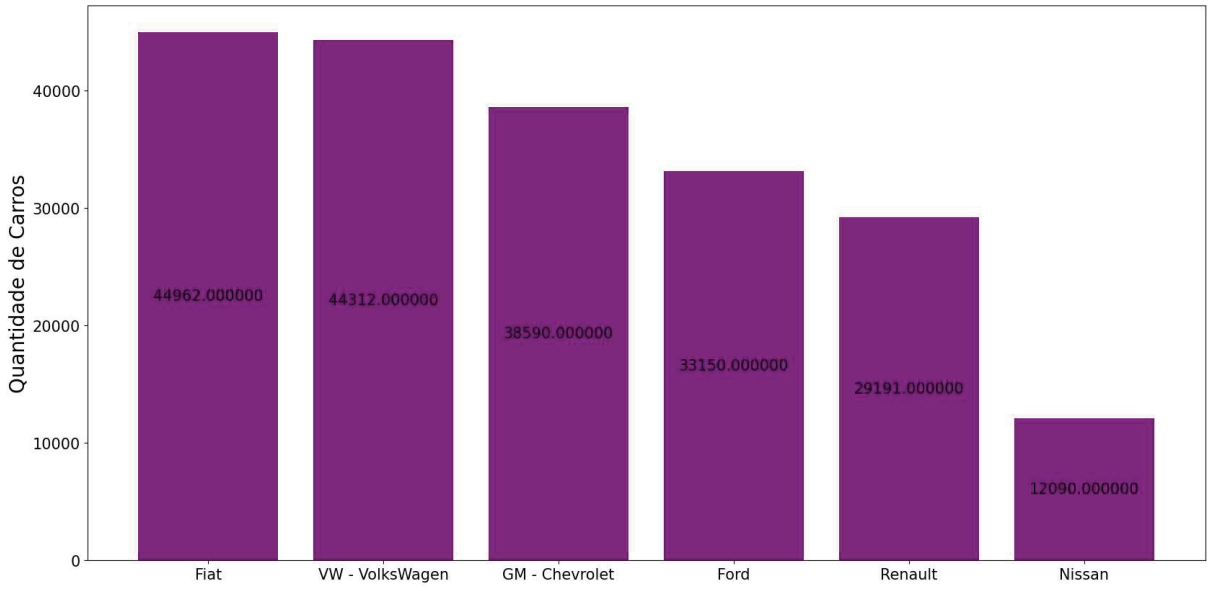
f) Analisando 267.542 linhas de informações sobre carros de 6 marcas diferentes, observou-se que a FIAT possui a maior variação de preço. O modelo da FIAT que mais se destaca nesse aspecto é o Palio Week Adv/Adv TRYON 1.8 mpi Flex. Em seguida, o modelo da Ford, Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p, também apresenta uma variação significativa de preço.

2 Visualização dos dados

a)

```
# Gerando gráfico de distribuição da quantidade de carros por marca
marcas = dados['brand'].value_counts()

plt.figure(figsize=(20,10))
grafico_1=plt.bar(marcas.index,marcas.values , color='purple')
plt.ylabel('Quantidade de Carros', fontsize=20)

plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)

plt.bar_label(grafico_1, fmt="%01f", size=15, label_type='center')
```

GRÁFICO 1 – DISTRIBUIÇÃO DE CARROS - POR MARCA



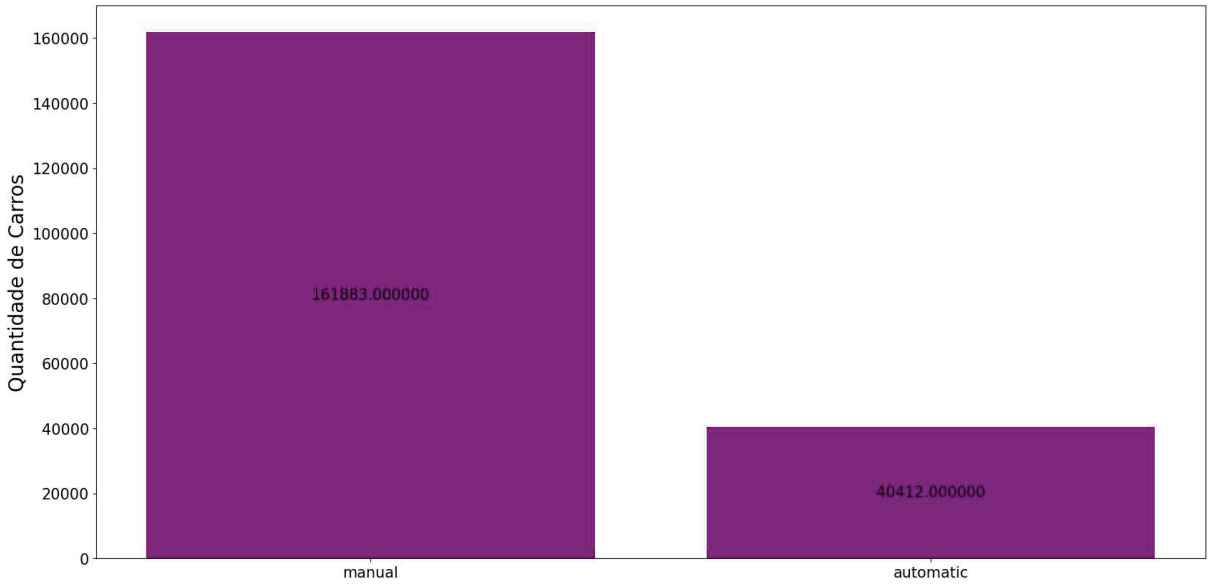
FONTE: O autor (2025)

b)

```
# Gerando gráfico de distribuição da quantidade de carros por tipo de engrenagem
do carro
plt.figure(figsize=(20,10))
grafico_1=plt.bar(dados['gear'].unique(), dados['gear'].value_counts(),
color='purple')
plt.ylabel('Quantidade de Carros', fontsize=20)

plt.xticks(fontsize = 15)
```

```
plt.yticks(fontsize = 15)

plt.bar_label(grafico_1, fmt="%01f", size=15, label_type='center')
```

GRÁFICO 2 – DISTRIBUIÇÃO DE CARROS - POR ENGRENAGEM

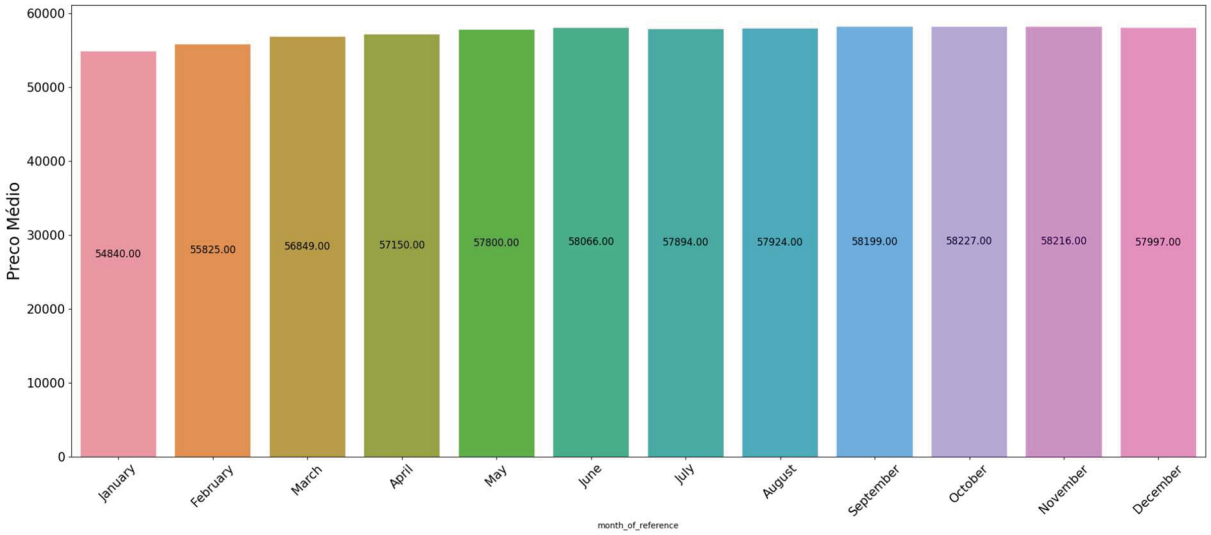


FONTE: O autor (2025)

c)

```
# Gerando gráfico de evolução de média de preço dos carros ao longo dos meses de
2022
plt.figure(figsize=(25,10))
grafico_3 = sns.barplot(x='month_of_reference', y='Preco Medio',
data=media_2022, order=order)
plt.xticks(rotation=45)
plt.ylabel('Preco Médio', fontsize=20)

for container in grafico_3.containers:
    grafico_3.bar_label(container, fmt='%.2f', fontsize=12, label_type='center')

plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
```

GRÁFICO 3 – EVOLUÇÃO MÉDIA DE PREÇOS

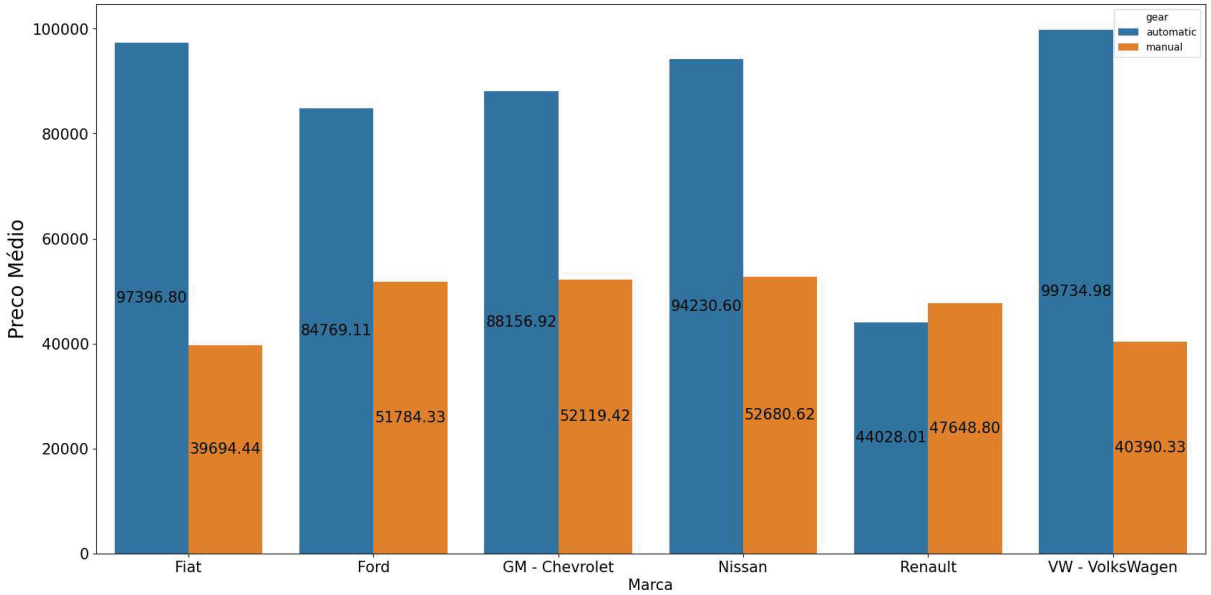


FONTE: O autor (2025)

d)

```
# Gerando gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem
plt.figure(figsize=(20,10))
grafico_4=sns.barplot(x='brand', y='Preco Medio', hue='gear', data=media_preco)
plt.ylabel('Preco Médio', fontsize=20)
plt.xlabel('Marca', fontsize=15)

for container in grafico_4.containers:
   grafico_4.bar_label(container, fmt='%.2f', fontsize=15, label_type='center')

plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
```

GRÁFICO 4 – MÉDIA DE PREÇO DOS CARROS MARCA E ENGRENAGEM



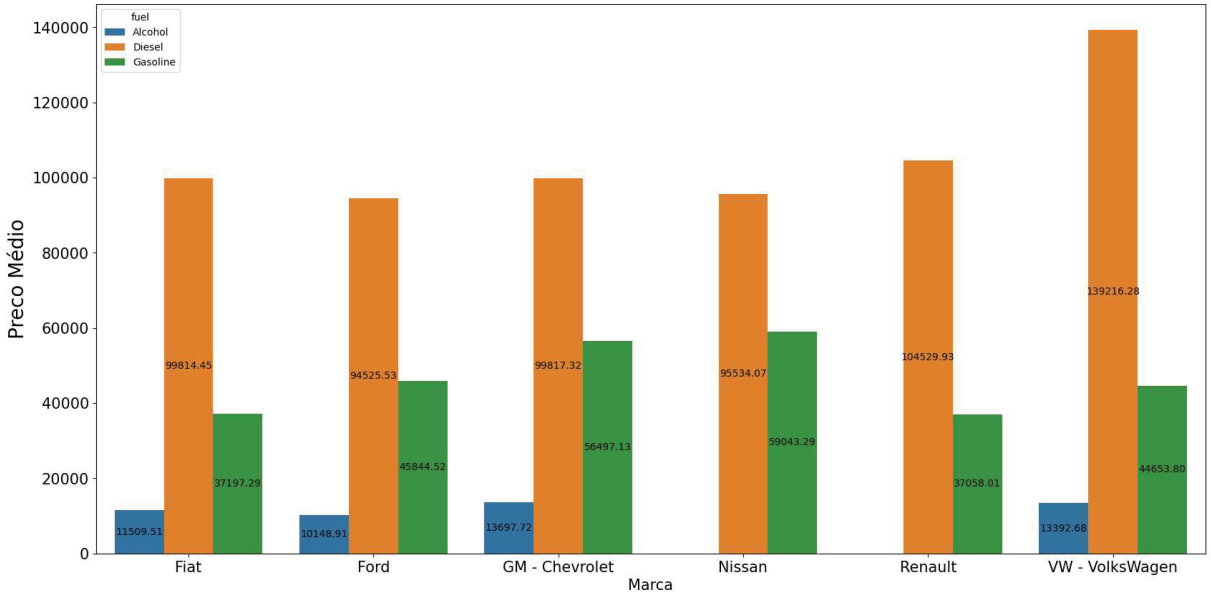
FONTE: O autor (2025)

e) Os carros com transmissão manual têm preços mais altos em comparação com os automáticos. O modelo da VW - Volkswagen com transmissão manual é o mais caro, enquanto o da Renault é o mais barato. A FIAT tem a menor média de preço para carros com transmissão manual, mas, em contrapartida, fica em segundo em termos de valores mais altos quando a engrenagem é automática.

f)

```
#Gerando gráfico da distribuição da média de preço dos carros por marca e tipo
de combustível
plt.figure(figsize=(20,10))
grafico_5 = sns.barplot(x='brand', y='Preco Medio combustivel', hue='fuel',
data=media_preco_combustivel)
plt.ylabel('Preco Médio', fontsize=20)
plt.xlabel('Marca', fontsize=15)

for container in grafico_5.containers:
   grafico_5.bar_label(container, fmt='%.2f', fontsize=10, label_type='center')

plt.xticks(fontsize = 15)
plt.yticks(fontsize = 15)
```

GRÁFICO 5 – MÉDIA DE PREÇO DOS CARROS MARCA E COMBUSTÍVEL



FONTE: O autor (2025)

g) Os carros a diesel da VW Volkswagen geralmente têm os preços mais altos, enquanto os outros têm preços que são comparativamente equilibrados, com pouca diferença entre eles. Os veículos movidos a álcool têm os preços mais baixos. Algumas marcas nem trabalham com carros movidos a álcool.

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

a)

```
# Conversão de colunas catégóricas em númericas
# Colunas escolhidas month_of_reference, brand, model

# Lista de colunas categóricas, com as que vamos trasnformar
## Coluna ano_preco incluida pois está no formato de data
colunas_categoricas = ['month_of_reference', 'brand', 'model', 'fuel', 'gear',
'ano_preco']

# Criar uma instância do LabelEncoder
label_encoder = LabelEncoder()

# Aplicar o LabelEncoder a cada coluna categórica
for coluna in colunas_categoricas:
    dados[coluna] = label_encoder.fit_transform(dados[coluna])
```

b)

```
#Crie partições contendo 75% dos dados para treino e 25% para teste
#Divisão: 30% dos dados são de teste e 70% para treinamento
```

```
X_train, X_test, Y_train, y_teste = train_test_split(X,Y, test_size=0.30,
random_state=42)
```

c)

```
#Treinando modelo RandomForest
rf = RandomForestRegressor()
rf.fit(X_train, Y_train)
```

```
#Treinando modelos XGBoost
xgb = XGBRegressor()
xgb.fit(X_train, Y_train)
```

d)

```
#Gravando os valores preditos em variáveis criadas - RandomForest
# Predição dos valores de Preço médio dos carros com base nos dados de teste
valores_preditos_rf = rf.predict(X_test)
```

```
#Gravando os valores preditos em variáveis criadas - XGBoost
# Predição dos valores de Preço médio dos carros com base nos dados de teste
valores_preditos_xgb = xgb.predict(X_test)
```

e)

```
#Análise de importância das variáveis para estimar a variável target -
RandomForest
# Analisando a importancia das váriaveis
rf.feature_importances_
feature_importances = pd.DataFrame(rf.feature_importances_, index =
X_train.columns, columns=['Importancia RF']).sort_values('Importancia RF',
ascending = False)
```

```
#Análise de importância das variáveis para estimar a variável target - XGBoost
# Analisando a importancia das váriaveis
xgb.feature_importances_
feature_importances = pd.DataFrame(xgb.feature_importances_, index =
X_train.columns, columns=['importancia XGBoost']).sort_values('importancia
XGBoost', ascending = False)
```

FIGURA 7 – OUTPUT ANÁLISE DE IMPORTÂNCIA

	# Importancia RF
ano_preco	0.0061045054766929565
brand	0.047434883206238655
fuel	0.1757461744615524
gear	0.021862783608782934
model	0.3874319522087209
month_of_reference	0.005300904683340731
year_model	0.34943482653320784
year_of_reference	0.00668396982146362

	# importancia XGBoost
ano_preco	0.0
brand	0.036090504
fuel	0.6643175
gear	0.06578748
model	0.077330664
month_of_reference	0.0037597176
year_model	0.1424528
year_of_reference	0.010261289

FONTE: O autor (2025)

f) Em todos os modelos usados as estimativas de year_of_reference e month_of_reference a importância permaneceram baixa, então, elas podem ser retiradas numa próxima análise. Enquanto isso, podemos marcar uma importância de análise para as year_model, fuel e model.

g) Melhor modelo segundo as métricas de avaliação MSE, MAE e R^2 foi o Random Forest

h) O modelo Random Forest obteve um desempenho superior, alcançando uma precisão de 99%, em comparação com o modelo XGBoost, que atingiu 98%. O XGBoost conseguiu melhorar a previsão para o tipo de combustível, tornando essa variável mais significativa. No entanto, outras variáveis que eram inicialmente mais fortes no Random Forest se tornaram menos influentes no modelo XGBoost.

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

onde y_i é o valor observado, $\widehat{y_i}$ é o valor predito e $\underline{y}$ é a média dos valores y_i observados. Quanto mais perto de 1 melhor é o modelo;

- Erro padrão da estimativa: S_{yx}

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1 Pesquisa com Dados de Satélite (Satellite)

1. Carregue a base de dados Satellite

```
# Carregando a base de dados Satellite
data("Satellite")
```

2. Crie partições contendo 80% para treino e 20% para teste

```
# Criando repartições - 80% Treino e 20% Teste
indices <- createDataPartition(Satellite$classes, p=0.80, list=FALSE)

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

3. Treine modelos RandomForest, SVM e RNA para predição destes dados.

```
# Treinando Modelo - RandomForest
rf <- caret::train(classes~., data=treino, method='rf')
predict.rf <- predict(rf, teste)

# Treinando Modelo - SVM
svm <- train(classes~., data=treino, method='svmRadial')
predict.svm <- predict(svm, teste)
```

```
# Treinando Modelo - RNA
rna <- train(classes~., data=treino, method='nnet')
predict.rna <- predict(rna, teste)
```

4. Escolha o melhor modelo com base em suas matrizes de confusão.

FIGURA 8 – OUTPUT MATRIX CONFUSÃO RANDOM FOREST

```
> # Matriz de confusão - RF
>
> confusionMatrix(predict.rf, teste$classes)
Confusion Matrix and Statistics

                    Reference
Prediction           red soil cotton crop grey soil damp grey soil vegetation stubble very damp grey soil
  red soil                301           0         3              1                 7                   0
  cotton crop               0         138         0              1                 1                   1
  grey soil                 3           0       263             26                 0                   4
  damp grey soil            0           0         3             77                 0                  13
  vegetation stubble        2           0         0              0               125                   3
  very damp grey soil       0           2         2             20                 8                 280

Overall Statistics

               Accuracy : 0.9221
                 95% CI : (0.9061, 0.9362)
    No Information Rate : 0.2383
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9034

 Mcnemar's Test P-Value : NA
```

FONTE: O autor (2025)

FIGURA 9 – OUTPUT MATRIX CONFUSÃO SVM

```
> # Matriz de confusão - SMV
>
> confusionMatrix(predict.svm, teste$classes)
Confusion Matrix and Statistics

                    Reference
Prediction           red soil cotton crop grey soil damp grey soil vegetation stubble very damp grey soil
  red soil                303           0         2              0                 5                   0
  cotton crop               0         138         2              2                 2                   2
  grey soil                 2           0       261             27                 0                   7
  damp grey soil            0           1         5             74                 1                  21
  vegetation stubble        1           0         0              1               126                   3
  very damp grey soil       0           1         1             21                 7                 268

Overall Statistics

               Accuracy : 0.9112
                 95% CI : (0.8943, 0.9262)
    No Information Rate : 0.2383
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8901

 Mcnemar's Test P-Value : NA
```

FONTE: O autor (2025)

FIGURA 10 – OUTPUT MATRIX CONFUSÃO RNA

```
> # Matriz de confusão - RNA
>
> confusionMatrix(predict.rna, teste$classes)
Confusion Matrix and Statistics

                    Reference
Prediction           red soil cotton crop grey soil damp grey soil vegetation stubble very damp grey soil
  red soil                272           3         1              0                 2                   0
  cotton crop              13         136         1              7                95                   2
  grey soil                20           0        40              5                 1                   3
  damp grey soil            0           0         0              0                 0                   0
  vegetation stubble        0           0         0              0                 1                   0
  very damp grey soil       1           1       229            113                42                 296

Overall Statistics

               Accuracy : 0.5802
                 95% CI : (0.5527, 0.6074)
    No Information Rate : 0.2383
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4692

 Mcnemar's Test P-Value : NA
```

FONTE: O autor (2025)

5. Indique qual modelo dá o melhor o resultado e a métrica utilizada

Com base nas matrizes de confusão, seguindo pela Métrica de Acurácia, o modelo Random Forest foi o que gerou o melhor resultado, pois obteve a maior acurácia, com um valor de 0.9221.

2 Estimativa de Volumes de Árvores

1. Carregar o arquivo Volumes.csv (http://www.razer.net.br/datasets/Volumes.csv)

```
# Carregar o arquivo Volumes
dados <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv")
```

2. Eliminar a coluna NR, que só apresenta um número sequencial

```
# Eliminando a coluna NR
dados$NR <- NULL
head(dados)
str(dados)
```

3. Criar partição de dados: treinamento 80%, teste 20%

```
# Criando repartições - 80% Treino e 20% Teste
indices2 <- createDataPartition(dados$VOL, p=0.80, list=FALSE)
treino2 <- dados[indices2, ]
teste2 <- dados[-indices2, ]
```

4. Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de SPURR

```
# Treinando Modelo - RandomForest
rf2 <- caret::train(VOL~., data=treino2, method='rf')

# Treinando Modelo - SVM
svm2 <- caret::train(VOL~., data=treino2, method='svmRadial')
```

```
# Treinando Modelo - Redes Neurais
rna2 <- caret::train(VOL~., data=treino2, method='nnet')

#  Modelo Alométrico
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, treino2 ,start=list(b0=0.5, b1=0.5))
```

5. Efetue as predições nos dados de teste

```
# Predição - Modelo - RandomForest
predict.rf2 <- predict(rf2, teste2)

# Predição - Modelo - SVM
predict.svm2 <- predict(svm2, teste2)

# Predição - Modelo - Redes Neurais
predict.rna2 <- predict(rna2, teste2)

# Predição Modelo Alométrico
predict.alom <- predict(alom, teste2)
```

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

```
# Funções

fx <- function(vpred, vreal){
  R2 <- 1 - (sum( (vreal - vpred)^2 ) / sum( (vreal - mean(vreal))^2 ))
  Sxy <- sqrt( sum( (vreal - vpred)^2 ) / ( length(vpred)-2 ) )
  SxyPerc <- ( Sxy / mean(vreal) ) * 100
  cat('Coeficiente de Determinação R²:', R2, '\n')
  cat('Erro Padrão da Estimativa:', Sxy, '\n')
  cat('Porcentagem do Erro:', SxyPerc, '\n')
}
```

FIGURA 11 – OUTPUT PREDIÇÃO DOS MODELOS

```
> fx(predict.rf2, teste2$VOL)
Coeficiente de Determinação R²: 0.8535647
Erro Padrão da Estimativa: 0.1445527
Porcentagem do Erro: 11.07658
>
> fx(predict.svm2, teste2$VOL)
Coeficiente de Determinação R²: 0.8484652
Erro Padrão da Estimativa: 0.1470481
Porcentagem do Erro: 11.2678
>
> fx(predict.rna2, teste2$VOL)
Coeficiente de Determinação R²: -0.7244946
Erro Padrão da Estimativa: 0.49606
Porcentagem do Erro: 38.01139
>
> fx(predict.alom, teste2$VOL)
Coeficiente de Determinação R²: 0.8263134
Erro Padrão da Estimativa: 0.1574296
Porcentagem do Erro: 12.0633
```

FONTE: O autor (2025)

7. Escolha o melhor modelo.

O modelo que apresenta o melhor resultado ao comparar os dados nesse caso, é o RandomForest. Pois, o seu erro de estimativa é o mais próximo de 0, com o valor de 0.1445527 e a métrica SYX%, que indica a porcentagem de erro, é a mais próxima de 0, com o valor de 11.07658. Apesar disso, no Coeficiente de Determinação, ele obteve o último resultado em comparação com os outros modelos, pois é o mais longe de 0. Nesse caso, o modelo Alométrico teve o melhor resultado, com um o valor -0.7244946. No entanto, o RandomForest foi o melhor modelo, pois levando em consideração e comparando com os outros 3, ele obteve os melhores resultados de duas métricas.

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

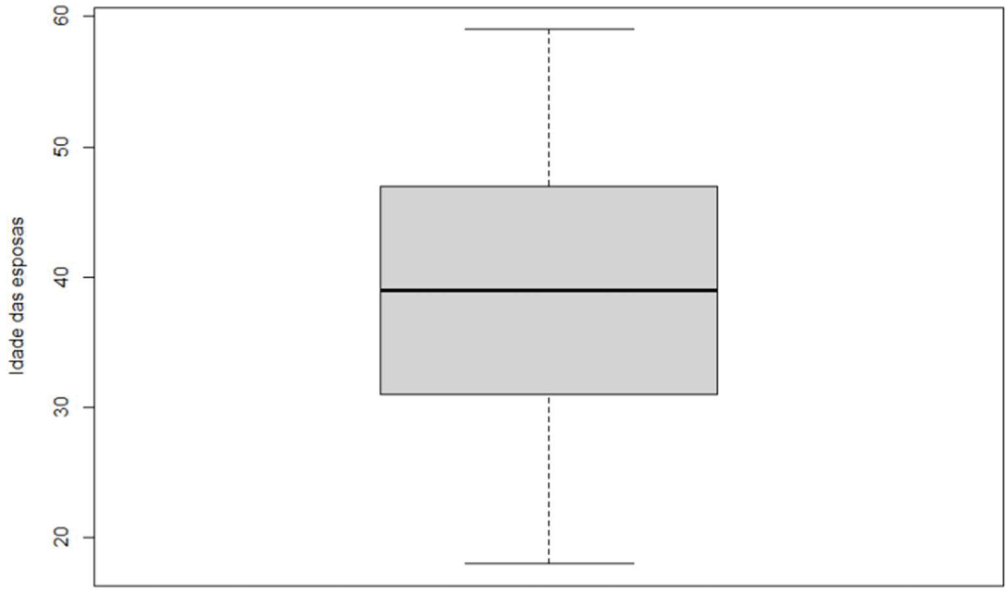
1) Gráficos e tabelas

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

BOXPLOT:

```
# Boxplot para a variavel "age" - (Idade da esposa)
Boxplot( ~ age, data=salarios, id=list(method="y"), ylab="Idade das esposas")
```

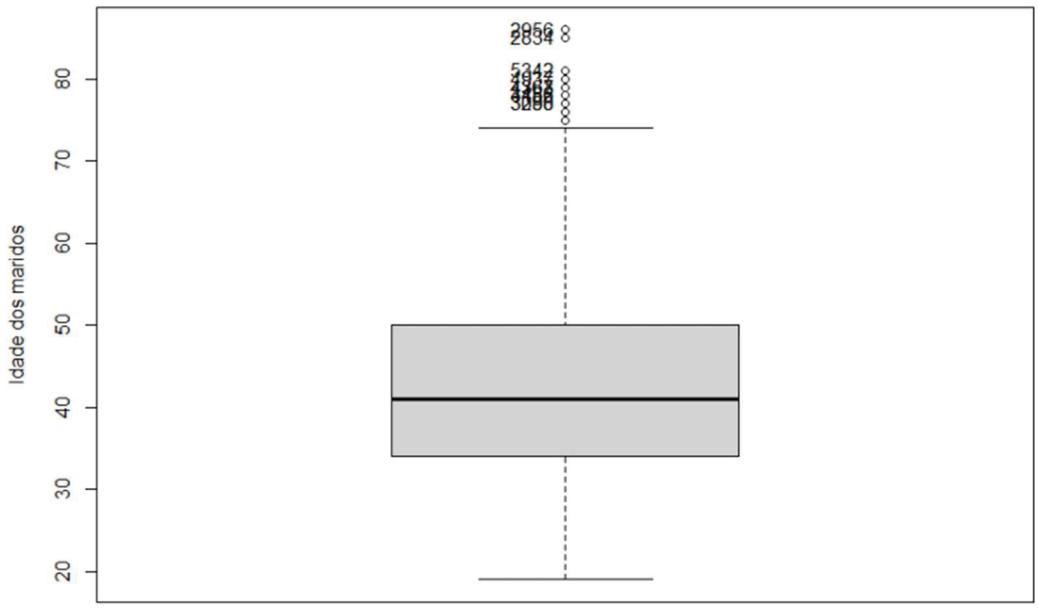
GRÁFICO6 – BOX PLOT DA IDADE DAS ESPOSAS DA BASE DE DADOS SALARIOS.RDATA



FONTE: O autor (2025)

```
# Boxplot para a variável "husage" - (Idade dos maridos)
Boxplot( ~ husage, data=salários, id=list(method="y"), ylab="Idade dos maridos")
```

GRÁFICO 7 – BOX PLOT DA IDADE DOS MARIDOS DA BASE DE DADOS SALARIOS.RDATA



FONTE: O autor (2025)

BoxPlot:

Nos gráficos BoxPlot, criados, sobre a idade das esposas e dos maridos, observamos que a mediana da idade das esposas é próxima de 40 anos, com a idade mínima próxima de 20 anos e a

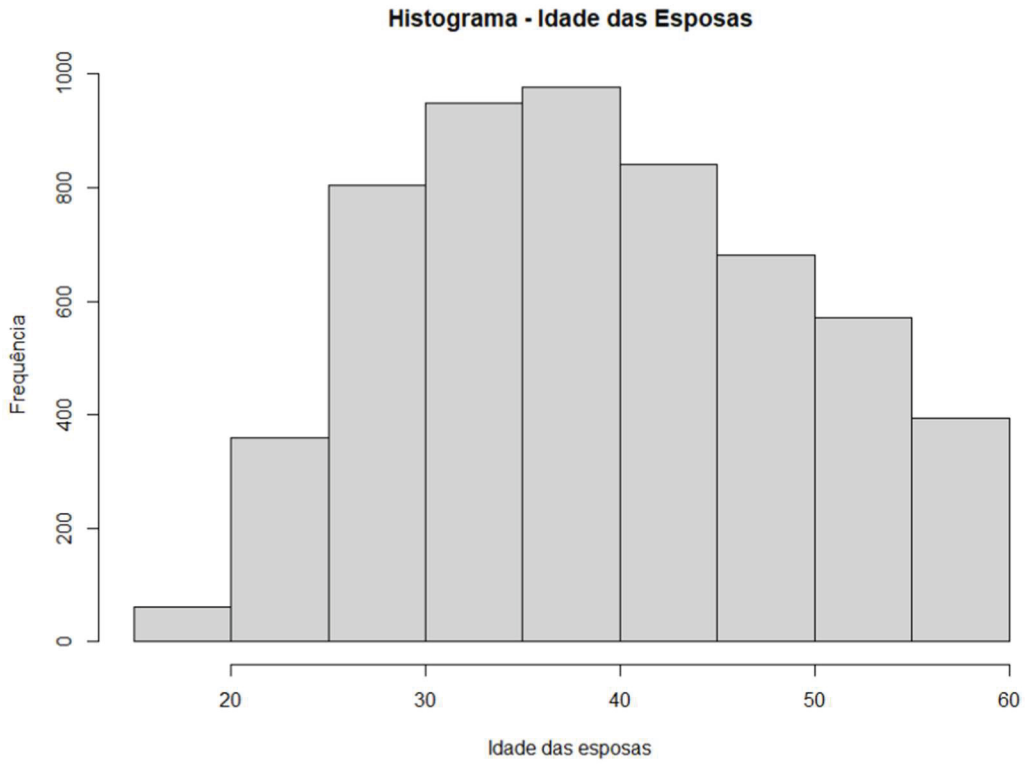
idade máxima próxima de 60 anos. O primeiro quartil da idade das esposas é próximo de 30 anos, enquanto o terceiro quartil é um pouco menor que 50 anos.

Já em relação à idade dos maridos, a mediana é um pouco maior que 40 anos. O limite inferior da idade dos maridos é muito próximo de 20 anos, enquanto o limite superior ultrapassa os 70 anos. Nesse caso, há diversos outliers na amostra, ou seja, muitos valores que estão fora dos padrões esperados.

HISTOGRAMA:

```
# Histograma para a variável "age" - (Idade da esposa)
hist(salarios$age, xlab="Idade das esposas", ylab = "Frequência", main =
"Histograma - Idade das Esposas")
```

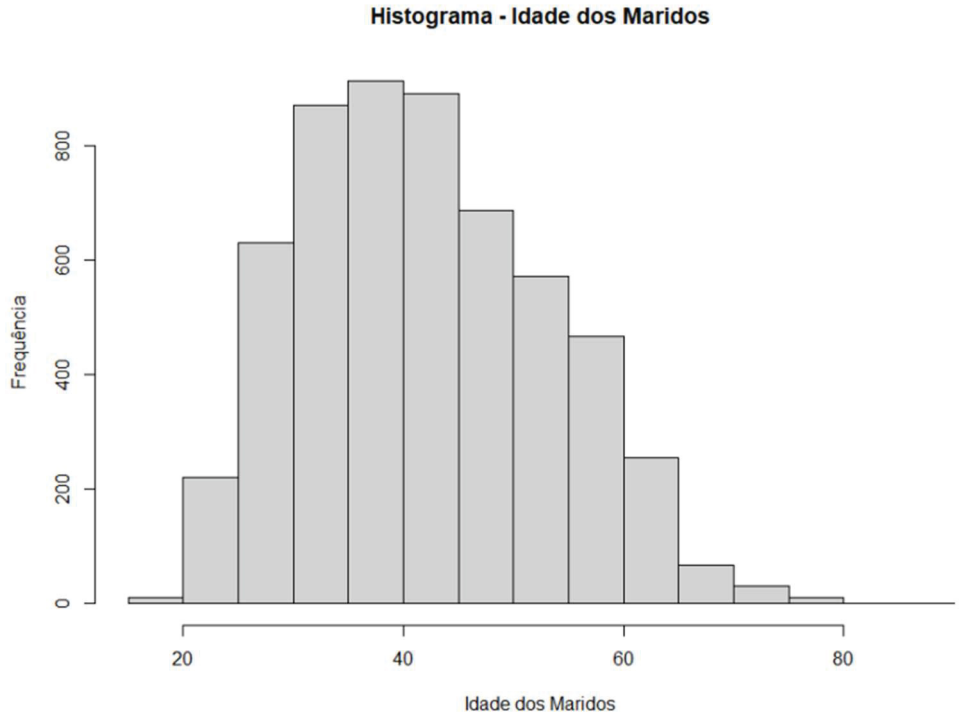
GRÁFICO 8 – DISTRIBUIÇÃO DE FREQUÊNCIA DAS IDADES DAS ESPOSAS DA BASE DE DADOS SALARIOS.RDATA



FONTE: O autor (2025)

```
# Histograma para a variável "husage" - (Idade dos maridos)
hist(salarios$husage, xlab="Idade dos Maridos", ylab= "Frequência", main
="Histograma - Idade dos Maridos")
```

GRÁFICO 9 – DISTRIBUIÇÃO DE FREQUÊNCIA DAS IDADES DOS MARIDOS DA BASE DE DADOS SALARIOS.RDATA



FONTE: O autor (2025)

Histograma:

O histograma das figuras 3 e 4 apresenta a distribuição dos dados referentes à idade das esposas e dos maridos. É possível observar que a frequência é maior em torno dos 40 anos de idade. Isso significa que, nos dados do arquivo salarios.RData, a maioria da idade das esposas e dos maridos está concentrada em torno dos 40 anos.

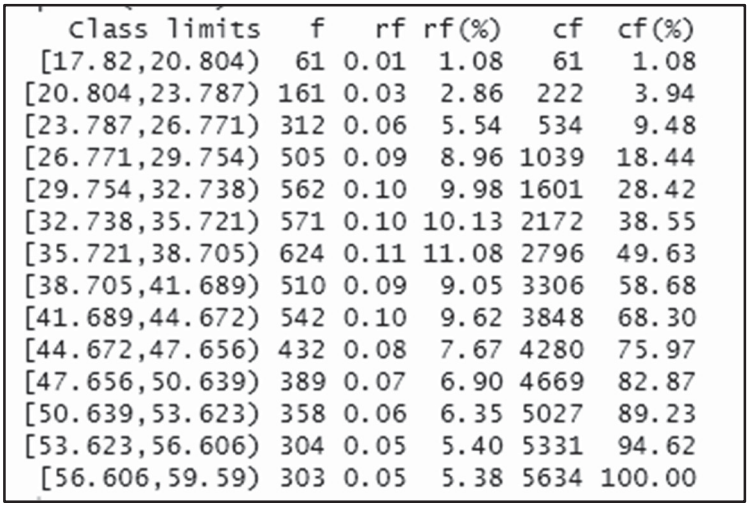
Analisando os dados de forma geral, conforme visualizados nos gráficos de BoxPlot e histograma, observa-se que a idade das esposas e dos maridos está majoritariamente concentrada em torno dos 40 anos. A idade das esposas varia de próximo dos 20 a 60 anos, enquanto a dos maridos varia de próximo dos 20 a 80 anos.

Sendo que os maridos possuem uma idade maior que os das esposas em sua totalidade. É importante destacar também que os dados dos maridos apresentam muitos outliers, ou seja, há muitas idades que estão fora desse intervalo, o que indica uma maior dispersão dos dados em relação à idade dos maridos.

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

```
# Tabela de Frequência "Age" - Idade das esposas
table <- fdt(salarios$age)
print(table)
```

FIGURA 12 – OUTPUT TABELA DE FREQUÊNCIA DA IDADE DAS ESPOSAS DA BASE DE DADOS SALÁRIOS

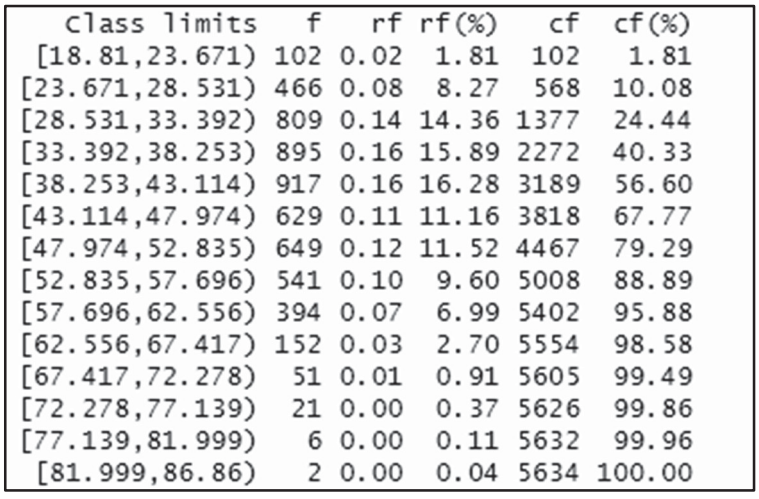


Class limits	f	rf	rf(%)	cf	cf(%)
[17.82,20.804)	61	0.01	1.08	61	1.08
[20.804,23.787)	161	0.03	2.86	222	3.94
[23.787,26.771)	312	0.06	5.54	534	9.48
[26.771,29.754)	505	0.09	8.96	1039	18.44
[29.754,32.738)	562	0.10	9.98	1601	28.42
[32.738,35.721)	571	0.10	10.13	2172	38.55
[35.721,38.705)	624	0.11	11.08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

FONTE: O autor (2025)

```
# Tabela de Frequência "husage" - Idade dos maridos
table <- fdt(salarios$husage)
print(table)
```

FIGURA13 – OUTPUT TABELA DE FREQUÊNCIA DA IDADE DOS MARIDOS DA BASE DE DADOS SALÁRIOS



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

FONTE: O autor (2025)

Na tabela da Figura 5, que se refere à idade das esposas, observamos que a idade mínima na base é de 17 anos, enquanto a máxima é de 59 anos. Sendo que 624 das esposas, possuem idade

entre 35 e 38 anos, o que representa 11.08% da totalidade das esposas na base de dados. Na Figura 6, que aborda a idade dos maridos, a idade mínima é de 18 anos, e a máxima é de 86 anos. Sendo relevante notar que 917 maridos têm idade entre 38 e 43 anos, correspondendo a 16.28% do total na base de dados. Dentre um total de 5.634 esposas e maridos.

2) Medidas de posição e dispersão

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

Idade das Esposas

```
# Média
mean(salarios$age)
```

Média = 39.42758

```
# Mediana
median(salarios$age)
```

Mediana = 39

```
# Moda
table(salarios$age)
subset(table(salarios$age),
       table(salarios$age) == max(table(salarios$age))
)
```

Moda = 37

(Sendo 217 esposas, que possuem 37 anos)

Idade dos Maridos

```
# Média
mean(salarios$husage)
```

Média = 42.45296

```
# Mediana
median(salarios$husage)
```

Mediana = 41

```
# Moda
table(salarios$husage)
subset(table(salarios$husage),
```

```
      table(salarios$husage) == max(table(salarios$husage))
)
```

Moda = 44

(Sendo 201 Maridos, que possuem 44 anos)

A média, mediana e moda da idade dos maridos são superiores às das esposas. A média de idade dos maridos é 7,67% maior, enquanto a mediana e a moda são 5.12% e 18.92% maiores, respectivamente, em comparação com as esposas.

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

Idade das Esposas

```
# Variância
var(salarios$age)
```

Variância = 99.75234

```
# Desvio Padrão
sd(salarios$age)
```

Desvio Padrão = 9.98761

```
# Coeficiente de Variação
MedAge <- mean(salarios$age)
dvAge <- sd(salarios$age)
cvage <- (dvAge/MedAge)*100
cvage
```

Coeficiente de Variação = 25.33153

Idade dos Maridos

```
# Variância
var(salarios$husage)
```

Variância = 126.0717

```
# Desvio Padrão
sd(salarios$husage)
```

Desvio Padrão = 11.22817

```
# Coeficiente de Variação
```

```
Medhus <- mean(salarios$husage)
dvhus <- sd(salarios$husage)
cvhus <- (dvhus/Medhus)*100
cvhus
```

Coeficiente de Variação = 26.44849

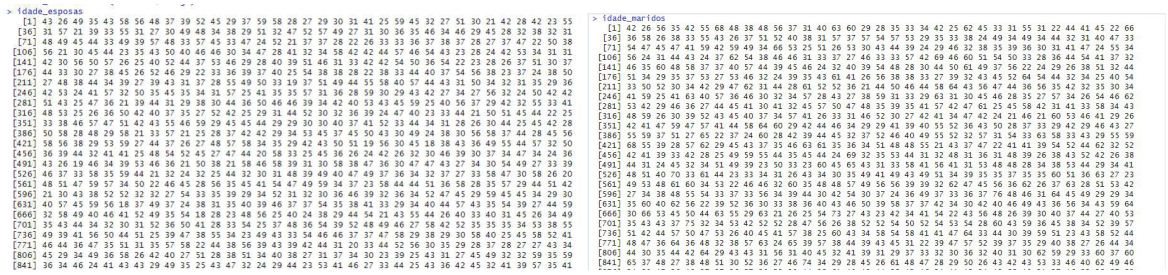
A variância, desvio padrão e coeficiente de variação das idades dos maridos é maior que os das esposas. A variância da idade dos maridos é 26,38% maior que a das esposas, enquanto o desvio padrão e o coeficiente de variação são 12.42% e 4.41% maiores, respectivamente, em comparação com a idades das esposas.

3) Testes paramétricos ou não paramétricos

a) 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.

```
# Separando as colunas em vetores
idade_esposas <- c(salários$age)
idade_maridos <- c(salários$husage)
```

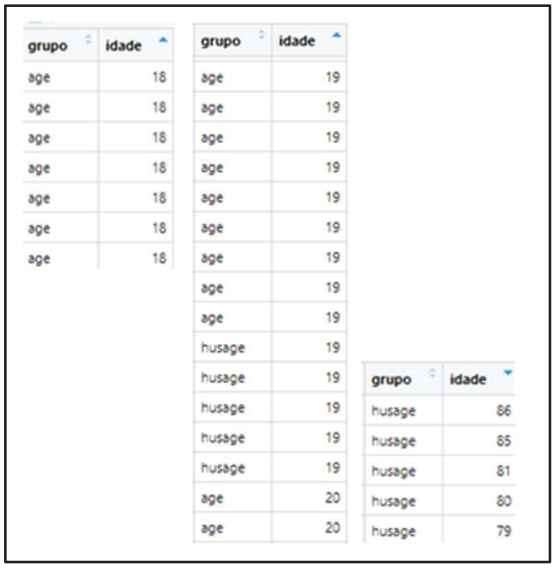
FIGURA14 – OUTPUT PREDIÇÃO DOS MODELOS



FONTE: O autor (2025)

```
# Criando um dataframe com o nome “idades”
idade <- data.frame(
  grupo = rep(c("age", "husage"), each = 5634),
  idade = c(idade_esposas, idade_maridos)
)
```

FIGURA 15 – OUTPUT DATAFRAME DE IDADES



grupo	idade
age	18
age	18
age	18
age	18
age	18
age	18
age	18

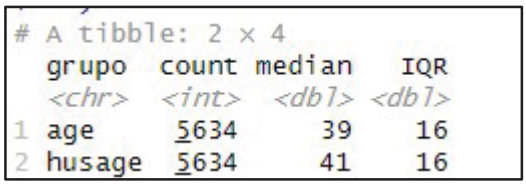
grupo	idade
age	19
age	19
age	19
age	19
age	19
age	19
age	19
age	19
age	19
husage	19
husage	19
husage	19
husage	19
husage	19
age	20
age	20

grupo	idade
husage	86
husage	85
husage	81
husage	80
husage	79

FONTE: O autor (2025)

```
# Calculando um sumario estatistico
group_by(idade, grupo) %>%
  summarise(
    count = n(),
    median = median(idade, na.rm = TRUE),
    IQR = IQR(idade, na.rm = TRUE)
  )
```

FIGURA 16 – OUTPUT SUMÁRIO ESTATÍSTICO

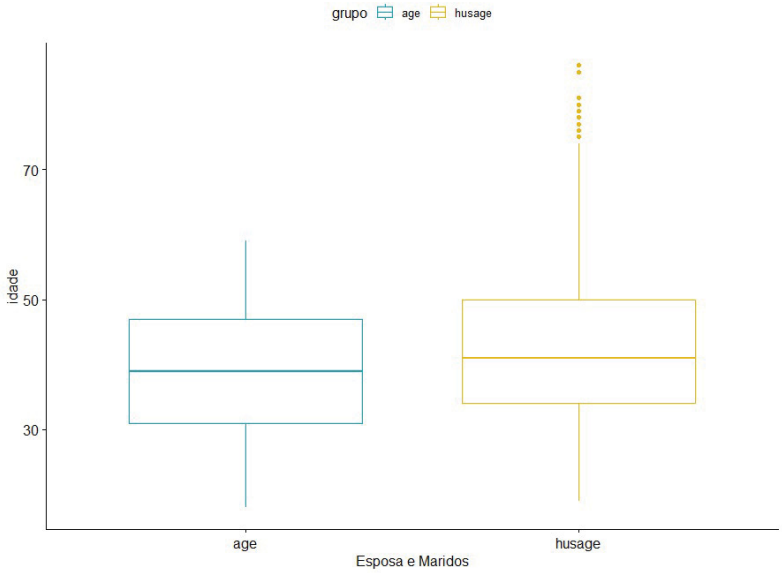


```
# A tibble: 2 × 4
  grupo  count median   IQR
  <chr>  <int>  <dbl> <dbl>
1 age     5634     39    16
2 husage  5634     41    16
```

FONTE: O autor (2025)

```
# Vamos visualizar os dados usando box-plots
ggboxplot(idade, x = "grupo", y = "idade",
          color = "grupo", palette=c("#00AFBB", "#E7B800"),
          ylab = “idade", xlab = "Esposas e Maridos")
```

GRÁFICO 10 – BOXPLOT IDADE DAS ESPOSAS E MARIDOS



FONTE: O autor (2025)

TESTE NÃO PARAMÉTRICO - U de MANN-WHINEY

Objetivo: Testar se as medianas das variáveis 'age' e 'husage' são iguais

Teste 1

H0: A idade dos maridos é igual estatisticamente a idade das esposas

Ha: A idade dos maridos "não " é estatisticamente igual a idade das esposas

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

FIGURA 17 – OUTPUT WILCOX TEST DA DIFERENÇA DE IDADES

```
> resultado <- wilcox.test(idade ~ grupo, data = idade, exact = FALSE, conf.int=TRUE)
> resultado

        Wilcoxon rank sum test with continuity correction

data:  idade by grupo
W = 13619912, p-value < 0.00000000000000022
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -3.000024 -2.000033
sample estimates:
difference in location
            -2.999966
```

FONTE: O autor (2025)

Resultado p-value = 0.00000000000000022 menor que 0.05, ou seja, 'rejeitamos' HO. A diferença entre as idades é de -2.999966. O Intervalo de Confiança é de -3.000024 até -2.000033. Nesse caso, A idade dos maridos 'não' é estatisticamente igual a idade das esposas.

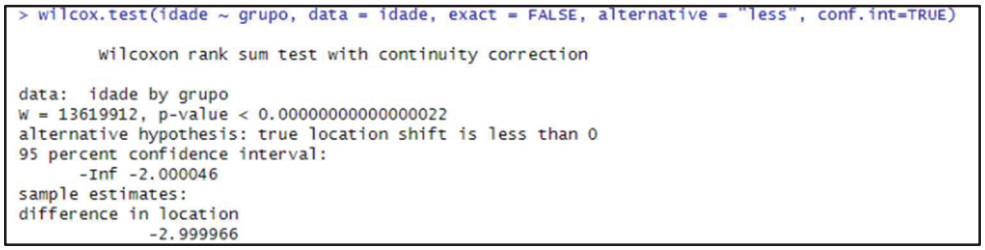
Teste 2

H0: A idade dos maridos 'não' é estatisticamente menor que a idade das esposas

Ha: A idade dos maridos é estatisticamente menor que a idade das esposas

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

FIGURA 18 – OUTPUT RESULTADO DO WILCOX TEST DA MEDIANA ENTRE AS IDADES



FONTE: O autor (2025)

O resultado d p-value = 0.00000000000000022 sendo menor que 0.05, ou seja, 'rejeitamos' HO. A diferença entre as idades é de -2.999966. O Intervalo de Confiança é de -2.000046. Nesse caso, A idade dos maridos ‘não’ é estatisticamente menor que a idade das esposas.

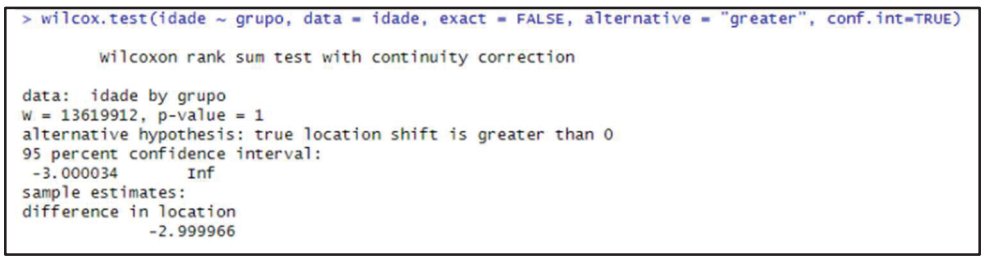
Teste 3

H0: A idade dos maridos 'não' é estatisticamente maior que a idade das esposas

Ha: A idade dos maridos é estatisticamente maior que a idade das esposas

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

FIGURA 19 – OUTPUT WILCOX TEST LOGICA INVERSA



FONTE: O autor (2025)

Resultado p-value = 1 maior que 0.05, ou seja, não rejeitamos HO. Então, A idade dos maridos é estatisticamente maior que a idade das esposas. A diferença entre as idades é de -2.999966. O

Intervalo de Confiança é de -3.000034. Então, a idade dos maridos é estaticamente maior que a idade das esposas.

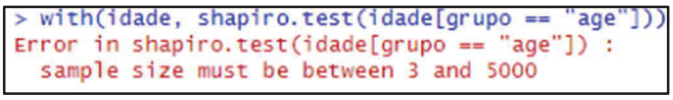
O principal motivo para a escolha do teste 'Não - Paramétrico', é que não é possível fazer esse processo de teste de hipóteses pelo Unpaired test, pois, ele não, atende todas as premissas.

Atende a Premissa 1, referente a ser amostras independentes, pois esposas e maridos são diferentes dados.

Na premissa 2, ao executar o comando abaixo, ocorreu um erro referente a quantidade de dados ultrapassar o limite. Pois, o teste de Shapiro-Wilk no R, suporta uma amostra de 5000, e a salarios.RData, tem um tamanho de 5634 observações.

```
with(idade, shapiro.test(idade[grupo == "age"])) # Erro devido a quantidade de amostra
```

FIGURA 20 – OUTPUT SHAPIRO TEST MARIDOS



FONTE: O autor (2025)

```
with(idade, shapiro.test(idade[grupo == "husage"])) # Erro devido a quantidade de amostra
```

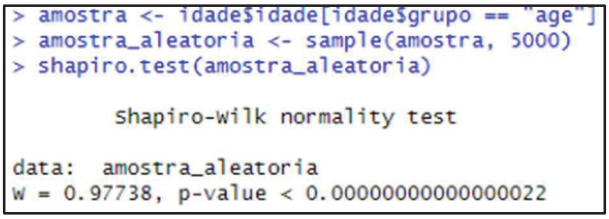
FIGURA 21 – OUTPUT SHAPIRO TEST ESPOSAS

FONTE: O autor (2025)

Uma opção utilizada para poder efetuar o teste, foi realizar uma seleção de amostra a quantidade máxima que é suportada.

```
amostra <- idade$idade[idade$grupo == "age"]
amostra aleatória <- sample(amostra, 5000) # Seleciona uma amostra aleatória de 5000 observações
shapiro.test(amostra aleatória)
```

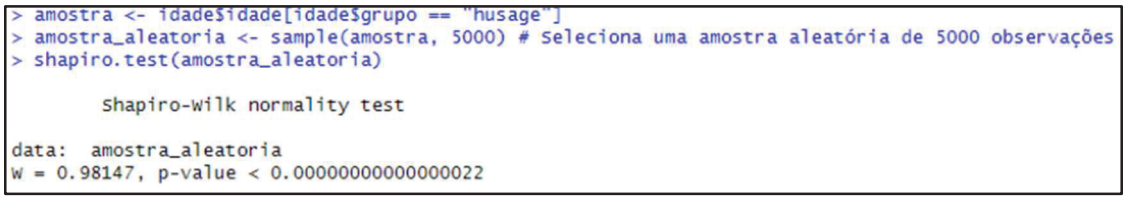
FIGURA 22 – OUTPUT SHAPIRO TEST AMOSTRA ALEATÓRIA MARIDOS



FONTE: O autor (2025)

```
amostra <- idade$idade[idade$grupo == "husage"]
amostra aleatória <- sample(amostra, 5000) # Seleciona uma amostra aleatória de
5000 observações
shapiro.test(amostra aleatória)
```

FIGURA 23 – OUTPUT SHAPIRO TEST AMOSTRA ALEATÓRIA ESPOSAS

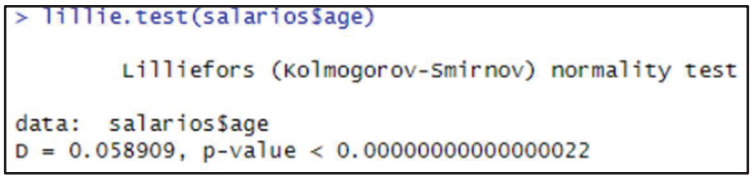


FONTE: O autor (2025)

E também foi feito teste usando funções lillie.test e JarqueBeraTest.

```
lillie.test(salários$age)
```

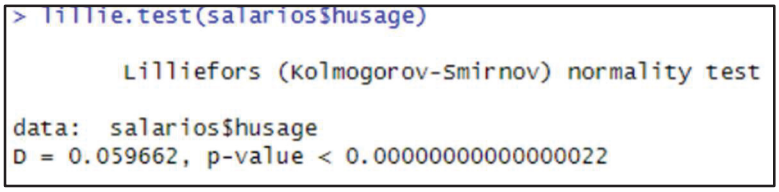
FIGURA 24 – OUTPUT LILLIE TEST MARIDOS



FONTE: O autor (2025)

```
lillie.test(salários$husage)
```

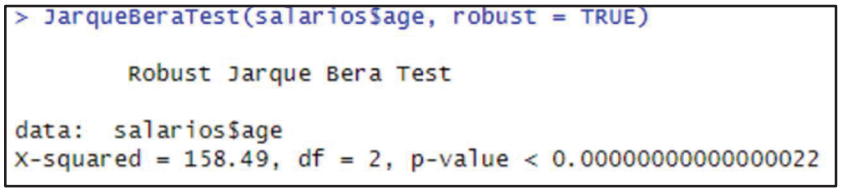
FIGURA 25 – OUTPUT LILLIE TEST ESPOSAS



FONTE: O autor (2025)

```
# Executando teste de normalidade de JarqueBeraTest
JarqueBeraTest(salarios$age, robust = TRUE)
```

FIGURA 26 – OUTPUT JARQUEBERATEST MARIDOS



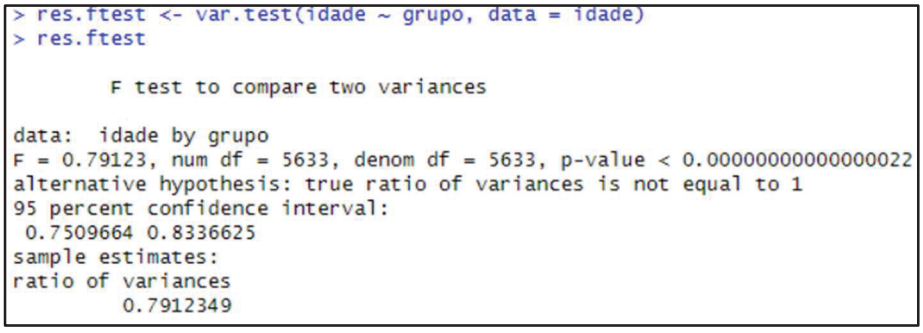
FONTE: O autor (2025)

A amostra obteve em todos os testes, nos dois casos, ‘age’ e ‘husage’, um valor de 0.00000000000000022 que é abaixo de 0,05, rejeitando H0, ou seja, ou dados não são normalmente distribuídos. Não atentando a premissa 2.

Na premissa 3, realizando um teste das variâncias se são estatisticamente iguais referente se são homogêneas, foi feito o teste abaixo.

```
# Usando o teste F para testar a homogeneidade nas variâncias
res.ftest <- var.test(idade ~ grupo, data = idade)
res.ftest
```

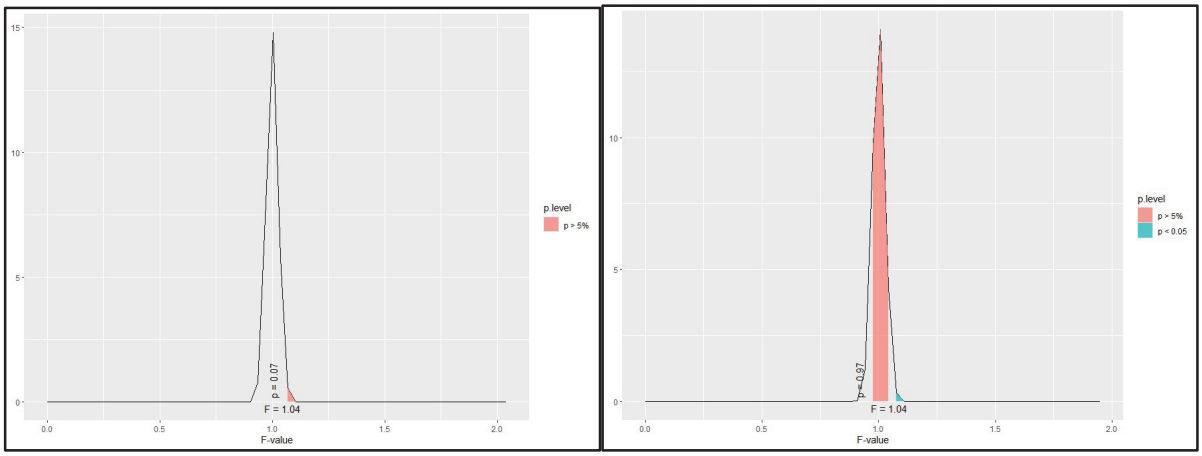
FIGURA 27 – OUTPUT TESTE DE HOMOGENEIDADE DAS VARIÂNCIAS



FONTE: O autor (2025)

```
dist_f(f = 1.04, deg.f1 = 5633, deg.f2 = 5633)
```

FIGURA 28 – OUTPUT VARIÂNCIAS



FONTE: O autor (2025)

É possível visualizar nos gráficos que foi rejeitado H0, ou seja, as variâncias não são estatisticamente iguais. Não atendendo então a 3 premissa também. Os dados então, não são homogêneos e não sendo normalmente distribuídos, não sendo possível utilizar o teste paramétrico unpaired teste.

Em resumo, foram realizados testes nas variáveis 'age' e 'husage' para comparar as medianas, utilizando o teste não paramétricos, U de Mann-Whitney. Os resultados indicaram que as idades das esposas e dos maridos não são estatisticamente iguais, sendo a idade dos maridos estatisticamente maior que a das esposas.

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

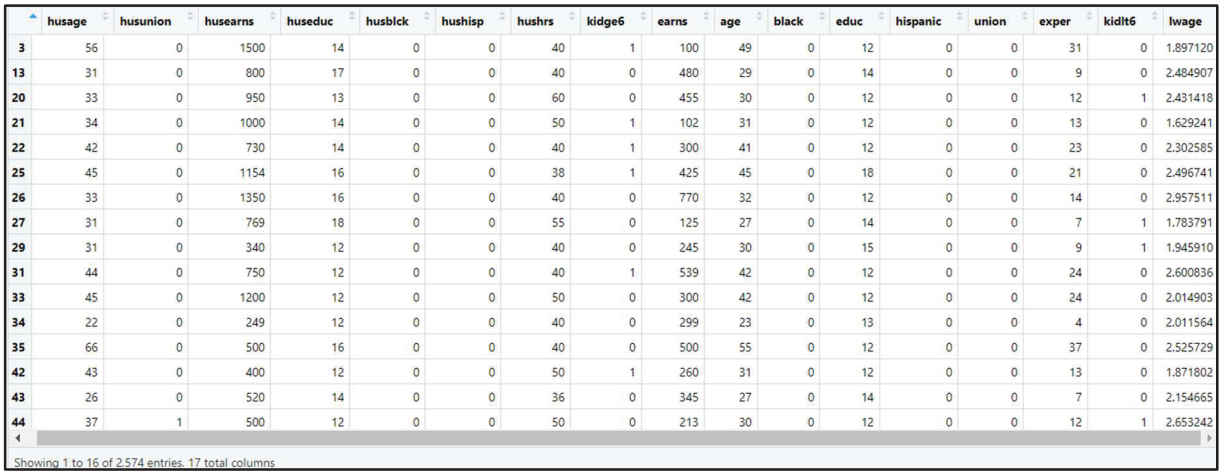
B – RESOLUÇÃO

1 Regressões Ridge, Lasso e ElasticNet

```
# Carregando o dataset
load("C:/Users/junio/Desktop/Memorial - UFPR/IAA005  - ESPERANDO
CODIGO/trabalhosalarios.RData")
```

```
# Vamos guardar este dataset em um objeto chamaddo "dat"
dat <- trabalhosalarios

# Vamos ver o dataset inteiro
View(dat)
```

FIGURA 29 – OUTPUT DATASET SALARIOS



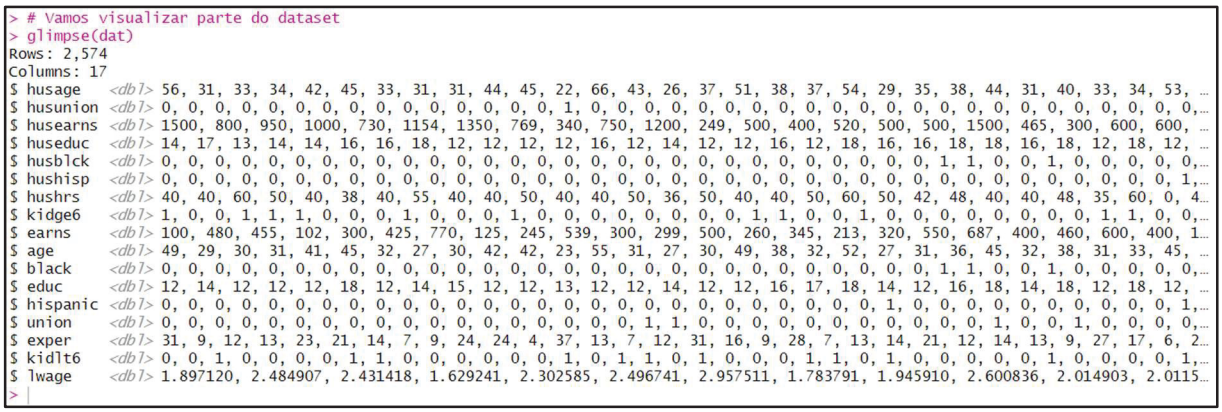
	husage	husunion	husearns	huseduc	husblck	hushisp	hushrs	kidge6	earns	age	black	educ	hispanic	union	exper	kidlt6	lwage
3	56	0	1500	14	0	0	40	1	100	49	0	12	0	0	31	0	1.897120
13	31	0	800	17	0	0	40	0	480	29	0	14	0	0	9	0	2.484907
20	33	0	950	13	0	0	60	0	455	30	0	12	0	0	12	1	2.431418
21	34	0	1000	14	0	0	50	1	102	31	0	12	0	0	13	0	1.629241
22	42	0	730	14	0	0	40	1	300	41	0	12	0	0	23	0	2.302585
25	45	0	1154	16	0	0	38	1	425	45	0	18	0	0	21	0	2.496741
26	33	0	1350	16	0	0	40	0	770	32	0	12	0	0	14	0	2.957511
27	31	0	769	18	0	0	55	0	125	27	0	14	0	0	7	1	1.783791
29	31	0	340	12	0	0	40	0	245	30	0	15	0	0	9	1	1.945910
31	44	0	750	12	0	0	40	1	539	42	0	12	0	0	24	0	2.600836
33	45	0	1200	12	0	0	50	0	300	42	0	12	0	0	24	0	2.014903
34	22	0	249	12	0	0	40	0	299	23	0	13	0	0	4	0	2.011564
35	66	0	500	16	0	0	40	0	500	55	0	12	0	0	37	0	2.525729
42	43	0	400	12	0	0	50	1	260	31	0	12	0	0	13	0	1.871802
43	26	0	520	14	0	0	36	0	345	27	0	14	0	0	7	0	2.154665
44	37	1	500	12	0	0	50	0	213	30	0	12	0	0	12	1	2.653242

Showing 1 to 16 of 2.574 entries. 17 total columns

FONTE: O autor (2025)

```
# Vamos visualizar parte do dataset
glimpse(dat)
```

FIGURA 30 – OUTPUT GLIMPSE



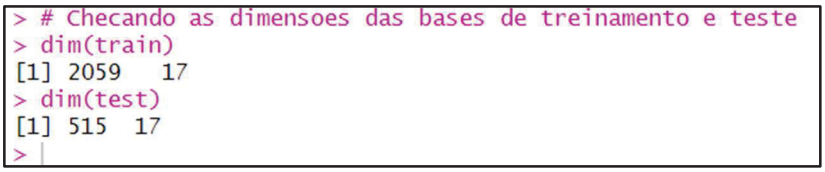
```
> # Vamos visualizar parte do dataset
> glimpse(dat)
Rows: 2,574
Columns: 17
$ husage    <dbl> 56, 31, 33, 34, 42, 45, 33, 31, 31, 44, 45, 22, 66, 43, 26, 37, 51, 38, 37, 54, 29, 35, 38, 44, 31, 40, 33, 34, 53, …
$ husunion  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,…
$ husearns  <dbl> 1500, 800, 950, 1000, 730, 1154, 1350, 769, 340, 750, 1200, 249, 500, 400, 520, 500, 500, 1500, 465, 300, 600, 600, …
$ huseduc   <dbl> 14, 17, 13, 14, 14, 16, 16, 18, 12, 12, 12, 12, 16, 12, 14, 12, 12, 16, 12, 18, 16, 16, 18, 18, 16, 18, 12, 18, 12, …
$ husblck   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,…
$ hushisp   <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,…
$ hushrs    <dbl> 40, 40, 60, 50, 40, 38, 40, 55, 40, 40, 50, 40, 40, 50, 36, 50, 40, 40, 50, 60, 50, 42, 48, 40, 40, 48, 35, 60, 0, 4…
$ kidge6    <dbl> 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,…
$ earns     <dbl> 100, 480, 455, 102, 300, 425, 770, 125, 245, 539, 300, 299, 500, 260, 345, 213, 320, 550, 687, 400, 460, 600, 400, 1…
$ age       <dbl> 49, 29, 30, 31, 41, 45, 32, 27, 30, 42, 42, 23, 55, 31, 27, 30, 49, 38, 32, 52, 27, 31, 36, 45, 32, 38, 31, 33, 45, …
$ black     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,…
$ educ      <dbl> 12, 14, 12, 12, 12, 18, 12, 14, 15, 12, 12, 13, 12, 12, 14, 12, 12, 16, 17, 18, 14, 12, 16, 18, 14, 18, 12, 18, 12, …
$ hispanic  <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,…
$ union     <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,…
$ exper     <dbl> 31, 9, 12, 13, 23, 21, 14, 7, 9, 24, 24, 4, 37, 13, 7, 12, 31, 16, 9, 28, 7, 13, 14, 21, 12, 14, 13, 9, 27, 17, 6, 2…
$ kidlt6    <dbl> 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,…
$ lwage     <dbl> 1.897120, 2.484907, 2.431418, 1.629241, 2.302585, 2.496741, 2.957511, 1.783791, 1.945910, 2.600836, 2.014903, 2.0115…
>
```

FONTE: O autor (2025)

```
# Jogando uma semente para os dados - Para obter o mesmo resultado
set.seed(302)

# Criando indice para reparticionar os dados
index = sample(1:nrow(dat),0.8*nrow(dat))
```

```
# base de dados de treinamento
train = dat[index,]

# base de dados de teste
test = dat[-index,]

# Checando as dimensoes das bases de treinamento e teste
dim(train)
dim(test)
```

FIGURA 31 – OUTPUT DIMENSÕES TREINAMENTO X TESTE



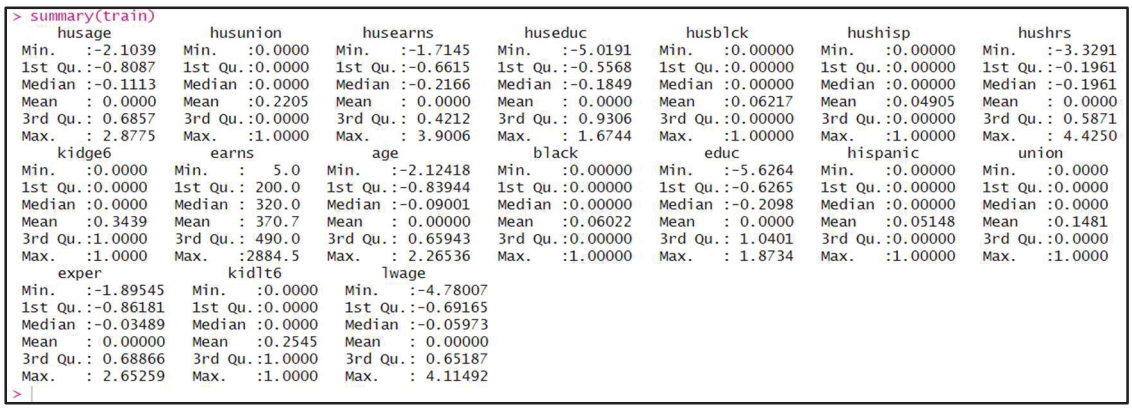
FONTE: O autor (2025)

Nesse caso, então, na base de treino são 2059 linhas e 17 colunas, e na base de teste ficou 515 linhas e 17 colunas.

```
# Padronizando variaveis
cols = c('husage', 'husearns', 'huseduc', 'hushrs', 'age', 'educ', 'exper',
'lwage')

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

```
summary(train)
```

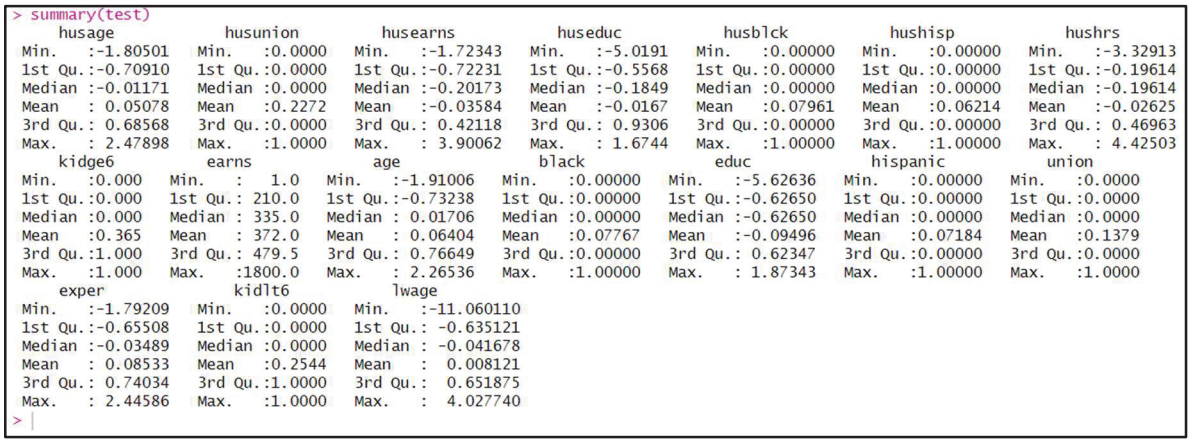
FIGURA 32 – OUTPUT SUMMARY TRAIN



FONTE: O autor (2025)

```
summary(test)
```

FIGURA 33 – OUTPUT SUMMARY TEST

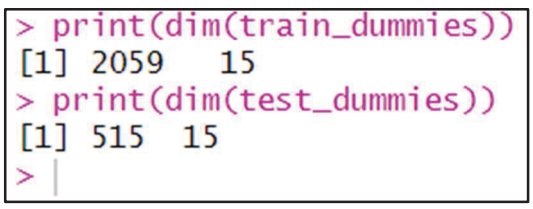


```
> summary(test)
     husage          husunion        husearns         huseduc          husblck          hushisp          hushrs      
 Min.   :-1.80501   Min.   :0.0000   Min.   :-1.72343   Min.   :-5.0191   Min.   :0.00000   Min.   :0.00000   Min.   :-3.32913  
 1st Qu.:-0.70910   1st Qu.:0.0000   1st Qu.:-0.72231   1st Qu.:-0.5568   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:-0.19614  
 Median :-0.01171   Median :0.0000   Median :-0.20173   Median :-0.1849   Median :0.00000   Median :0.00000   Median :-0.19614  
 Mean   : 0.05078   Mean   :0.2272   Mean   :-0.03584   Mean   :-0.0167   Mean   :0.07961   Mean   :0.06214   Mean   :-0.02625  
 3rd Qu.: 0.68568   3rd Qu.:0.0000   3rd Qu.: 0.42118   3rd Qu.: 0.9306   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.: 0.46963  
 Max.   : 2.47898   Max.   :1.0000   Max.   : 3.90062   Max.   : 1.6744   Max.   :1.00000   Max.   :1.00000   Max.   : 4.42503  
     kidge6            earns             age               black              educ            hispanic          union        
 Min.   :0.000   Min.   :   1.0   Min.   :-1.91006   Min.   :0.00000   Min.   :-5.62636   Min.   :0.00000   Min.   :0.0000  
 1st Qu.:0.000   1st Qu.: 210.0   1st Qu.:-0.73238   1st Qu.:0.00000   1st Qu.:-0.62650   1st Qu.:0.00000   1st Qu.:0.0000  
 Median :0.000   Median : 335.0   Median : 0.01706   Median :0.00000   Median :-0.62650   Median :0.00000   Median :0.0000  
 Mean   :0.365   Mean   : 372.0   Mean   : 0.06404   Mean   :0.07767   Mean   :-0.09496   Mean   :0.07184   Mean   :0.1379  
 3rd Qu.:1.000   3rd Qu.: 479.5   3rd Qu.: 0.76649   3rd Qu.:0.00000   3rd Qu.: 0.62347   3rd Qu.:0.00000   3rd Qu.:0.0000  
 Max.   :1.000   Max.   :1800.0   Max.   : 2.26536   Max.   :1.00000   Max.   : 1.87343   Max.   :1.00000   Max.   :1.0000  
     exper             kidlt6            lwage        
 Min.   :-1.79209   Min.   :0.0000   Min.   :-11.060110  
 1st Qu.:-0.65508   1st Qu.:0.0000   1st Qu.: -0.635121  
 Median :-0.03489   Median :0.0000   Median : -0.041678  
 Mean   : 0.08533   Mean   :0.2544   Mean   :  0.008121  
 3rd Qu.: 0.74034   3rd Qu.:1.0000   3rd Qu.:  0.651875  
 Max.   : 2.44586   Max.   :1.0000   Max.   :  4.027740  
> 
```

FONTE: O autor (2025)

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

# Gerando variaveis dummies para organizar os datasets
dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+husblck+hushisp+
                     hushrs+kidlt6+age+black+educ+hispanic+union+exper+kidge6,
                     data = dat[,cols_reg])

train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])

print(dim(train_dummies))
print(dim(test_dummies))
```

FIGURA 34 – OUTPUT DIMENSÕES DUMMIES



```
> print(dim(train_dummies))
[1] 2059   15
> print(dim(test_dummies))
[1] 515  15
> 
```

FONTE: O autor (2025)

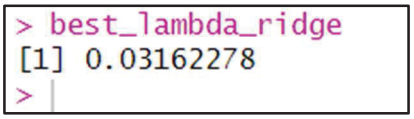
São 2059 linhas e 17 colunas, e na base de teste ficou 515 linhas e 17 colunas.

```
# Guardando a matriz de dados de treinamento das variaveis explicativas
x = as.matrix(train_dummies)

# Guardando o vetor de dados de treinamento da variavel dependente
y_train = train$lwage

# Guardando a matriz de dados de teste das variaveis explicativas
x_test = as.matrix(test_dummies)

# Guardando o vetor de dados de teste da variavel dependente
y_test = test$lwage
```

Regressão RIDGE

```
# Calculando o valor otimo de lambda;
lambdas <- 10^seq(2, -3, by = -.1)

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

# Verificar qual o lambda escolhido
best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge
```

FIGURA 35 – OUTPUT BEST LAMBDA RIDGE



FONTE: O autor (2025)

Então, parâmetro de encolhimento = 0.3162278

```
# Tempo de processamento do modelo usamos:
start <- Sys.time()

# Estimando o modelo Ridge
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
                   family = 'gaussian',
                   lambda = best_lambda_ridge)
end <- Sys.time()
difftime(end, start, units="secs")
```

FIGURA 36 – OUTPUT TIME DIFFERENCE EM SECS

```
> difftime(end, start, units="secs")
Time difference of 0.003547907 secs
>
```

FONTE: O autor (2025)

```
# Visualizar o resultado (valores) da estimativa (coeficientes)
ridge_reg[["beta"]]
```

FIGURA 37 – OUTPUT ESTIMATIVAS RIDGE_REG

```
> # Visualizar o resultado (valores) da estimativa (coeficientes)
> ridge_reg[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"
                   s0
husage    -0.004756816
husunion   0.001318781
husearns   0.230239210
huseduc    0.051382657
husblck    0.217033035
hushisp    0.086183874
hushrs    -0.069501449
kidlt6    -0.042681157
age        0.070023561
black     -0.277077875
educ       0.326890808
hispanic  -0.011399117
union      0.365677473
exper     -0.017057679
kidge6    -0.182020906
>
```

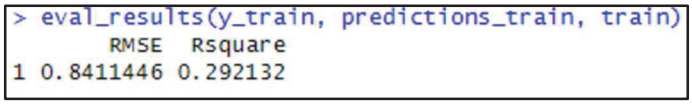
FONTE: O autor (2025)

Todos os valores próximos de 0 (zero).

```
# Calculando o R^2 dos valores verdadeiros e preditos
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

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

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

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

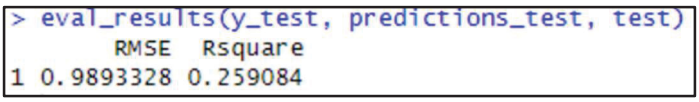
FIGURA 38 – OUTPUT MÉTRICAS RMSE E R2 DE TREINO



FONTE: O autor (2025)

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

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

FIGURA 39 – OUTPUT MÉTRICAS RMSE E R2 DE TESTE



FONTE: O autor (2025)

A métrica para a base de treinamento foi no RMSE 84.11% e R^2 29.21%. A métrica para a base de teste foi no RMSE 98.93% e R^2 25.90%. Valores do Rsquare foram bem parecidos nos testes. Na base de treinamento, foi um pouco melhor, pois o valor foi maior. O valor de Rsquare está muito baixo, ou seja, o modelo não está bom.

```
# Predicao para o nosso exemplo
husage = (40-pre_proc_val[["mean"]][["husage"]]) /
pre_proc_val[["std"]][["husage"]]
husunion = 0
husearns = (600-pre_proc_val[["mean"]][["husearns"]]) /
pre_proc_val[["std"]][["husearns"]]
huseduc = (13-pre_proc_val[["mean"]][["huseduc"]]) /
pre_proc_val[["std"]][["huseduc"]]
husblck = 1
hushisp = 0
hushrs = (40-pre_proc_val[["mean"]][["hushrs"]]) /
pre_proc_val[["std"]][["hushrs"]]
kidge6 = 1
age = (38-pre_proc_val[["mean"]][["age"]]) / pre_proc_val[["std"]][["age"]]
black = 0
educ = (13-pre_proc_val[["mean"]][["educ"]]) / pre_proc_val[["std"]][["educ"]]
hispanic = 1
```

```
union = 0
exper = (18-pre_proc_val[["mean"]][["exper"]]) /
pre_proc_val[["std"]][["exper"]]
kidlt6 = 1

# Construindo uma matriz de dados para a predicao
our_pred = as.matrix(data.frame(husage=husage,
                                husunion=husunion,
                                husearns=husearns,
                                huseduc=huseduc,
                                husblck=husblck,
                                hushisp=hushisp,
                                hushrs=hushrs,
                                kidge6=kidge6,
                                age=age,
                                black=black,
                                educ=educ,
                                hispanic=hispanic,
                                union=union,
                                exper=exper,
                                kidlt6=kidlt6))

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

# O resultado da predição é:
predict_our_ridge
```

FIGURA 40 – OUTPUT RESULTADO DA PREDIÇÃO

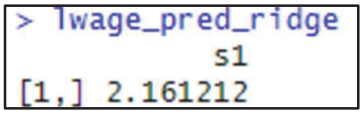
```
> predict_our_ridge
                s1
[1,] -0.06843352
```

FONTE: O autor (2025)

-0.6843352, é o valor padronizado.

```
# Convertendo o valor padronizado para um valor nominal
lwage_pred_ridge=(predict_our_ridge*
                    pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]

lwage_pred_ridge
```

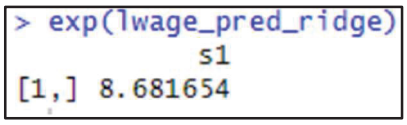
FIGURA 41 – OUTPUT LWAGE_PRED_RIDGE



FONTE: O autor (2025)

```
# O intervalo de confianca:
n <- nrow(train)
m <- lwage_pred_ridge
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_ridge <- m + (qnorm(0.025))*dam
CIupr_ridge <- m - (qnorm(0.025))*dam

# Aplicando o Antilog por conta do lwage já estar em logaritimo neperiano
# Os valores sao da predicao
exp(lwage_pred_ridge)
```

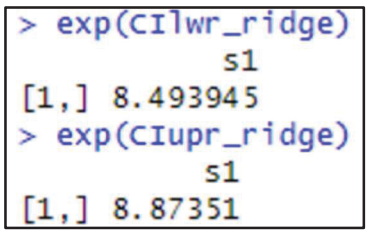
FIGURA 42 – OUTPUT TRANSFORMAÇÃO EXPONENCIAL DO RIDGE



FONTE: O autor (2025)

```
# Os valores sao:
exp(CIlwr_ridge)
exp(CIupr_ridge)
```

FIGURA 43 – OUTPUT LIMITES SUPERIOR E INFERIOR RIDGE



FONTE: O autor (2025)

Então, de acordo as características que atribuímos o salário-hora da esposa é em média US$8,68 e pode variar entre US$8,49 e US$8.87.

Regressão LASSO

```
#Criando o intervalo para tunning do melhor
lambdas <- 10^seq(2, -3, by = -.1)

lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
                        lambda = lambdas,
                        standardize = TRUE, nfolds = 5)

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

best_lambda_lasso
```

FIGURA 44 – OUTPUT PARÂMETRO REGULARIZAÇÃO OTIMO

```
> best_lambda_lasso
[1] 0.01258925
```

FONTE: O autor (2025)

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

# Visualizando os coeficientes estimados
lasso_model[["beta"]]
```

FIGURA 45 – OUTPUT ESTIMATIVA DOS COEFICIENTES

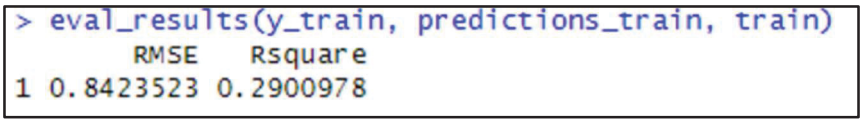
```
> lasso_model[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"
                   s0
husage      .
husunion    .
husearns    0.23002775
huseduc     0.03010233
husblck     .
hushisp     .
hushrs     -0.06025028
kidge6     -0.14252320
age         0.04689191
black      -0.03169581
educ        0.33915744
hispanic    .
union       0.34330443
exper       .
kidlt6      .
```

FONTE: O autor (2025)

Nesse caso, os coeficientes zeradis husage, husunion, husblck, hushisp, hispanic, exper e kidlt6 não são significativos. Entretando os husearns, huseduc, hushrs, kidge6, age, black, educ e union são importantes.

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

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

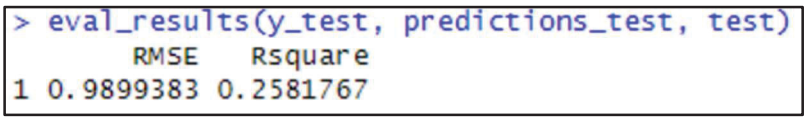
FIGURA 46 – OUTPUT MÉTRICAS DE DESEMPENHO DE TREINO



FONTE: O autor (2025)

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

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

FIGURA 47 – OUTPUT MÉTRICAS DE DESEMPENHO DE TESTE



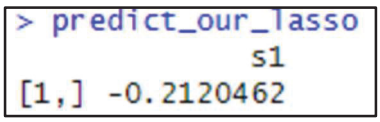
FONTE: O autor (2025)

A métrica para a base de treinamento foi no RMSE 84.23% e R^2 29%. A métrica para a base de teste foi no RMSE 98.99% e R^2 25.81%. Valores do Rsquare foram bem parecidos nos testes na base de treinamento e na base de teste. O valorde Rsquare está muito baixo, ou seja, o modelo não está bom. Pois, seu poder explicativo está entre 29% á 26% apenas.

```
# Predicao para o nosso exemplo
husage = (40-pre_proc_val[["mean"]][["husage"]]) /
pre_proc_val[["std"]][["husage"]]
husunion = 0
husearns = (600-pre_proc_val[["mean"]][["husearns"]]) /
```

```
pre_proc_val[["std"]][["husearns"]]
huseduc = (13-pre_proc_val[["mean"]][["huseduc"]]) /
pre_proc_val[["std"]][["huseduc"]]
husblck = 1
hushisp = 0
hushrs = (40-pre_proc_val[["mean"]][["hushrs"]]) /
pre_proc_val[["std"]][["hushrs"]]
kidge6 = 1
age = (38-pre_proc_val[["mean"]][["age"]]) / pre_proc_val[["std"]][["age"]]
black = 0
educ = (13-pre_proc_val[["mean"]][["educ"]]) / pre_proc_val[["std"]][["educ"]]
hispanic = 1
union = 0
exper = (18-pre_proc_val[["mean"]][["exper"]]) /
pre_proc_val[["std"]][["exper"]]
kidlt6 = 1

# Construindo uma matriz de dados para a predicao
our_pred = as.matrix(data.frame(husage=husage,
                                husunion=husunion,
                                husearns=husearns,
                                huseduc=huseduc,
                                husblck=husblck,
                                hushisp=hushisp,
                                hushrs=hushrs,
                                kidge6=kidge6,
                                age=age,
                                black=black,
                                educ=educ,
                                hispanic=hispanic,
                                union=union,
                                exper=exper,
                                kidlt6=kidlt6))

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

# O resultado da predição é:
predict_our_lasso
```

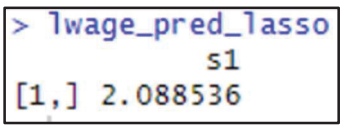
FIGURA 48 – OUTPUT PREDIÇÃO DOS MODELOS LASSO



FONTE: O autor (2025)

```
# Convertendo o valor padronizado para um valor nominal
lwage_pred_lasso=(predict_our_lasso*
                    pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]
lwage_pred_lasso
```

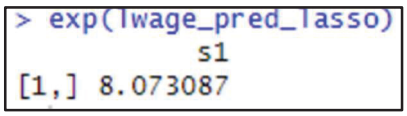
FIGURA 49 – OUTPUT ESTIMATIVA LWAGE



FONTE: O autor (2025)

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

#Aplicando o Antilog por conta do lwage já estar em logaritmo neperiano
exp(lwage_pred_lasso)
```

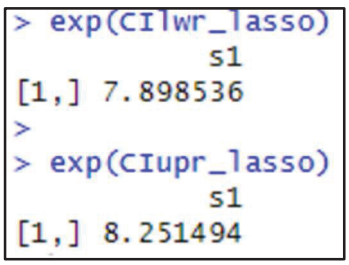
FIGURA 50 – OUTPUT ESTIMATIVAS DE SALÁRIO APÓS ANTILOG



FONTE: O autor (2025)

```
# Do intervalo de confinça
exp(CIlwr_lasso)
exp(CIupr_lasso)
```

FIGURA 51 – OUTPUT LIMITES SUPERIOR E INFERIOR LASSO



FONTE: O autor (2025)

Então, de acordo as características que atribuímos o salário-hora da esposa é em média US$8,07 e pode variar entre US$7,89 e US$8.25.

Regressão ELASTICNET

```
# Configuranado  o treinamento com cross validation
train_cont <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 5,
                           search = "random",
                           verboseIter = TRUE)

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

# O melhor parametro alpha escolhido:
elastic_reg$bestTune
```

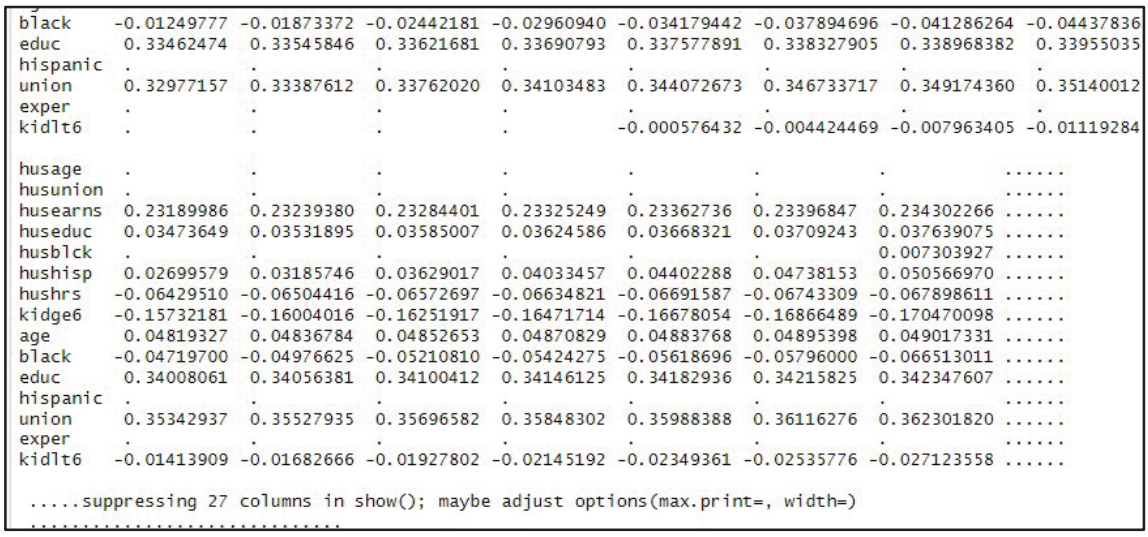
FIGURA 52 – OUTPUT MELHOR PARÂMETRO ELASTIC

```
+ Fold09.Rep5: alpha=0.91073, lambda=6.331177
- Fold09.Rep5: alpha=0.91073, lambda=6.331177
+ Fold10.Rep5: alpha=0.74298, lambda=0.012335
- Fold10.Rep5: alpha=0.74298, lambda=0.012335
+ Fold10.Rep5: alpha=0.76859, lambda=0.264158
- Fold10.Rep5: alpha=0.76859, lambda=0.264158
+ Fold10.Rep5: alpha=0.04725, lambda=0.804106
- Fold10.Rep5: alpha=0.04725, lambda=0.804106
+ Fold10.Rep5: alpha=0.16550, lambda=0.017630
- Fold10.Rep5: alpha=0.16550, lambda=0.017630
+ Fold10.Rep5: alpha=0.95704, lambda=0.330759
- Fold10.Rep5: alpha=0.95704, lambda=0.330759
+ Fold10.Rep5: alpha=0.17116, lambda=0.010915
- Fold10.Rep5: alpha=0.17116, lambda=0.010915
+ Fold10.Rep5: alpha=0.23852, lambda=0.002141
- Fold10.Rep5: alpha=0.23852, lambda=0.002141
+ Fold10.Rep5: alpha=0.81402, lambda=0.001129
- Fold10.Rep5: alpha=0.81402, lambda=0.001129
+ Fold10.Rep5: alpha=0.19023, lambda=3.468937
- Fold10.Rep5: alpha=0.19023, lambda=3.468937
+ Fold10.Rep5: alpha=0.91073, lambda=6.331177
- Fold10.Rep5: alpha=0.91073, lambda=6.331177
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.743, lambda = 0.0123 on full training set
```

FONTE: O autor (2025)

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

FIGURA 53 – OUTPUT COEFICIENTES ESTIMADOS PELO MODELO ELASTIC NET



```
black    -0.01249777 -0.01873372 -0.02442181 -0.02960940 -0.034179442 -0.037894696 -0.041286264 -0.04437836
educ      0.33462474  0.33545846  0.33621681  0.33690793  0.337577891  0.338327905  0.338968382  0.33955035
hispanic  .           .           .           .           .            .            .            .
union     0.32977157  0.33387612  0.33762020  0.34103483  0.344072673  0.346733717  0.349174360  0.35140012
exper     .           .           .           .           .            .            .            .
kidlt6    .           .           .           .          -0.000576432 -0.004424469 -0.007963405 -0.01119284

husage    .           .           .           .           .            .            .           ......
husunion  .           .           .           .           .            .            .           ......
husearns  0.23189986  0.23239380  0.23284401  0.23325249  0.23362736   0.23396847   0.234302266 ......
huseduc   0.03473649  0.03531895  0.03585007  0.03624586  0.03668321   0.03709243   0.037639075 ......
husblck   .           .           .           .           .            .            0.007303927 ......
hushisp   0.02699579  0.03185746  0.03629017  0.04033457  0.04402288   0.04738153   0.050566970 ......
hushrs   -0.06429510 -0.06504416 -0.06572697 -0.06634821 -0.06691587  -0.06743309  -0.067898611 ......
kidge6   -0.15732181 -0.16004016 -0.16251917 -0.16471714 -0.16678054  -0.16866489  -0.170470098 ......
age       0.04819327  0.04836784  0.04852653  0.04870829  0.04883768   0.04895398   0.049017331 ......
black    -0.04719700 -0.04976625 -0.05210810 -0.05424275 -0.05618696  -0.05796000  -0.066513011 ......
educ      0.34008061  0.34056381  0.34100412  0.34146125  0.34182936   0.34215825   0.342347607 ......
hispanic  .           .           .           .           .            .            .           ......
union     0.35342937  0.35527935  0.35696582  0.35848302  0.35988388   0.36116276   0.362301820 ......
exper     .           .           .           .           .            .            .           ......
kidlt6   -0.01413909 -0.01682666 -0.01927802 -0.02145192 -0.02349361  -0.02535776  -0.027123558 ......

 .....suppressing 27 columns in show(); maybe adjust options(max.print=, width=)
 ..............................
```

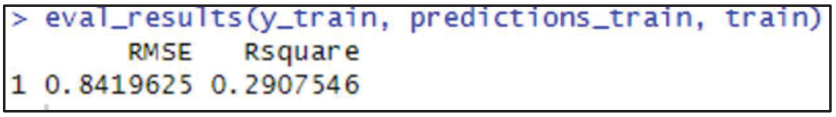
FONTE: O autor (2025)

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

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

```
# Calculando o R^2 dos valores verdadeiros e preditos
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

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

```
# As metricas de performance na base de treinamento
eval_results(y_train, predictions_train, train)
```

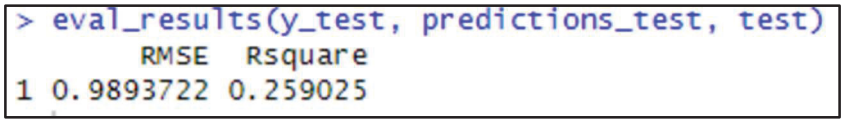
FIGURA 54 – OUTPUT MÉTRICAS DE DESEMPENHO DO MODELO NA BASE DE TREINO



FONTE: O autor (2025)

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

FIGURA 55 – OUTPUT MÉTRICAS DE DESEMPENHO DO MODELO NA BASE DE TESTE



FONTE: O autor (2025)

A métrica para a base de treinamento foi no RMSE 84.19% e R^2 29.07%. A métrica para a base de teste foi no RMSE 98.93% e R^2 25.90%. Valores baixos de Rsquare tanto na base de teste quanto na base de treinamento, ou seja, o modelo não está bom.

```
# Predicao para o nosso exemplo
husage = (40-pre_proc_val[["mean"]][["husage"]]) /
pre_proc_val[["std"]][["husage"]]
husunion = 0
husearns = (600-pre_proc_val[["mean"]][["husearns"]]) /
pre_proc_val[["std"]][["husearns"]]
huseduc = (13-pre_proc_val[["mean"]][["huseduc"]]) /
pre_proc_val[["std"]][["huseduc"]]
husblck = 1
hushisp = 0
hushrs = (40-pre_proc_val[["mean"]][["hushrs"]]) /
pre_proc_val[["std"]][["hushrs"]]
kidge6 = 1
age = (38-pre_proc_val[["mean"]][["age"]]) / pre_proc_val[["std"]][["age"]]
black = 0
educ = (13-pre_proc_val[["mean"]][["educ"]]) / pre_proc_val[["std"]][["educ"]]
hispanic = 1
union = 0
exper = (18-pre_proc_val[["mean"]][["exper"]]) /
pre_proc_val[["std"]][["exper"]]
kidlt6 = 1

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

```
                                husblck=husblck,
                                hushisp=hushisp,
                                hushrs=hushrs,
                                kidge6=kidge6,
                                age=age,
                                black=black,
                                educ=educ,
                                hispanic=hispanic,
                                union=union,
                                exper=exper,
                                kidlt6=kidlt6))

# Fazendo a predicao com base nos parametros
predict_our_elastic <- predict(elastic_reg,our_pred)

# O resultado da predição é:
predict_our_elastic
```

FIGURA 56 – OUTPUT PREDIÇÃO DO MODELO ELASTIC

```
> predict_our_elastic <- predict(elastic_reg,our_pred)
> predict_our_elastic
[1] -0.2292382
```

FONTE: O autor (2025)

```
# Revetendo para o nivel dos valores originais do
# dataset, vamos fazer isso:
lwage_pred_elastic=(predict_our_elastic*
                      pre_proc_val[["std"]][["lwage"]])+
  pre_proc_val[["mean"]][["lwage"]]
lwage_pred_elastic

# Intervalo de confianca
n <- nrow(train)
m <- lwage_pred_elastic
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_elastic <- m + (qnorm(0.025))*dam
CIupr_elastic <- m - (qnorm(0.025))*dam

# Aplicando o Antilog por conta do lwage já estar em logaritmo neperiano
exp(lwage_pred_elastic)
```

FIGURA 57 – OUTPUT SALÁRIOS PREVISTOS PELO MODELO ELASTIC NET

```
> exp(lwage_pred_elastic)
[1] 8.003155
```

FONTE: O autor (2025)

```
#Intervalo de confianca
exp(CIlwr_elastic)
exp(CIupr_elastic)
```

FIGURA 58 – OUTPUT INTERVALO DE CONFIANÇA DOS SALÁRIOS PREVISTOS

FONTE: O autor (2025)

Então, de acordo as características que atribuímos o salário-hora da esposa é em média US$8,00 e pode variar entre US$7,83 e US$8.18.

TABELA 1 – COMPARAÇÃO ENTRE OS MODELOS

		RMSE	RSQUARE
TREINAMENTO	RIDGE	0.8411446	0.292132
TESTE		0.9893328	0.259084
TREINAMENTO	LASSO	0.8423525	0.2900978
TESTE		0.9899383	0.258167
TREINAMENTO	ELASTIC	0.8419625	0.2907546
TESTE		0.9893722	0.259025

FONTE: O autor (2025)

O melhor modelo de regressão foi RIDGE, entre os três avaliados, ele obteve o melhor desempenho tanto na base de treinamento, quanto na de teste. O RIDGE, obteve o menor RMSE e o maior R^2.

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 Construção de Características: Identificador automático de idioma

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

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

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

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

```
"Gosto de fazer caminhadas na natureza.",
"O restaurante tem uma vista incrível.",
"Vamos sair para jantar no sábado.",
]
```

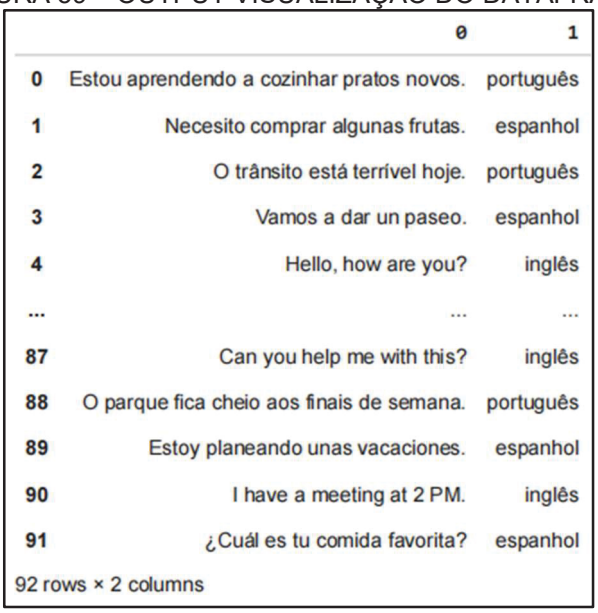
A "amostra" de texto precisa ser "transformada" em padrões. Um padrão é um conjunto de características, geralmente representado por um vetor e um conjunto de padrões no formato de tabela. Onde cada linha é um padrão e as colunas as características e, geralmente, na última coluna a classe.

```
import random
import numpy as np

seed_val = 7
random.seed(seed_val)
np.random.seed(seed_val)

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

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

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

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

O Dataframe do pandas facilita a visualização

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

FIGURA 59 – OUTPUT VISUALIZAÇÃO DO DATAFRAME



	0	1
0	Estou aprendendo a cozinhar pratos novos.	português
1	Necesito comprar algunas frutas.	espanhol
2	O trânsito está terrível hoje.	português
3	Vamos a dar un paseo.	espanhol
4	Hello, how are you?	inglês
...	...	...
87	Can you help me with this?	inglês
88	O parque fica cheio aos finais de semana.	português
89	Estoy planeando unas vacaciones.	espanhol
90	I have a meeting at 2 PM.	inglês
91	¿Cuál es tu comida favorita?	espanhol

92 rows × 2 columns

FONTE: O autor (2025)

```
# a entrada é o vetor pre_padroes e a saída desse passo deverá ser "padrões"
import re
import nltk
from nltk.corpus import stopwords
#from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer

# Baixando os recursos do NLTK
nltk.download('stopwords')
nltk.download('wordnet')

# Função para calcular o tamanho médio das palavras
def tamanhoMedioFrases(texto):
    palavras = re.split("\s", texto)
    tamanhos = [len(s) for s in palavras if len(s) > 0]
    soma = sum(tamanhos)
    return soma / len(tamanhos) if tamanhos else 0


# Função para contar o número de palavras
def numeroDePalavras(texto):
    palavras = re.split("\s", texto)
    return len([p for p in palavras if len(p) > 0])

# Função para calcular a frequência de vogais
def frequenciaDeVogais(texto):
    vogais = 'aeiou'
    texto = texto.lower()
    return sum(texto.count(v) for v in vogais) / len(texto)
```

```
# Função para calcular a frequência de consoantes
def frequenciaDeConsoantes(texto):
    consoantes = 'bcdfghjklmnpqrstvwxyz'
    texto = texto.lower()
    return sum(texto.count(c) for c in consoantes) / len(texto)

# Função para contar a presença de caracteres específicos
def caracteresEspecificos(texto):
    caracteres_espanhol = 'ñ'
    caracteres_portugues = 'çãõ'
    texto = texto.lower()
    num_caracteres_espanhol = sum(texto.count(c) for c in caracteres_espanhol)
    num_caracteres_portugues = sum(texto.count(c) for c in caracteres_portugues)
    return [num_caracteres_espanhol, num_caracteres_portugues]

# Função para calcular a frequência de stopwords
def frequenciaDeStopwords(texto, idioma):

    # Tratamento dos textos
    stop_words = set(stopwords.words(idioma))

    for sen in range(len(texto)):
        # Remove todos os caracteres especiais
        document = re.sub(r'\W', ' ', str(texto[sen]))

        # Remove todos os caracteres únicos
        document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

        # Remove caracteres únicos do início
        document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

        # Substitui múltiplos espaços por um único espaço
        document = re.sub(r'\s+', ' ', document, flags=re.I)

        # Remove o prefixo 'b'
        document = re.sub(r'^b\s+', '', document)

        # Converte para minúsculas
        document = document.lower()

    return sum(1 for palavra in document if palavra in stop_words) /
len(document) if document else 0

def extraiCaracteristicas(frase):

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

    caracteristica1 = tamanhoMedioFrases(texto)
    caracteristica2 = numeroDePalavras(texto)
```

```
    caracteristica3 = frequenciaDeVogais(texto)
    caracteristica4 = frequenciaDeConsoantes(texto)
    caracteristicas_especificas = caracteresEspecificos(texto)
    caracteristica5 = frequenciaDeStopwords(texto, 'english')
    caracteristica6 = frequenciaDeStopwords(texto, 'spanish')
    caracteristica7 = frequenciaDeStopwords(texto, 'portuguese')

    padrao = [
        caracteristica1,
        caracteristica2,
        caracteristica3,
        caracteristica4,
        *caracteristicas_especificas,
        caracteristica5,
        caracteristica6,
        caracteristica7,
        frase[1]
    ]
    return padrao

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

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

# apenas para visualizacao
print(padroes)

dados = pd.DataFrame(padroes)
dados
```

Treinando o modelo SVM

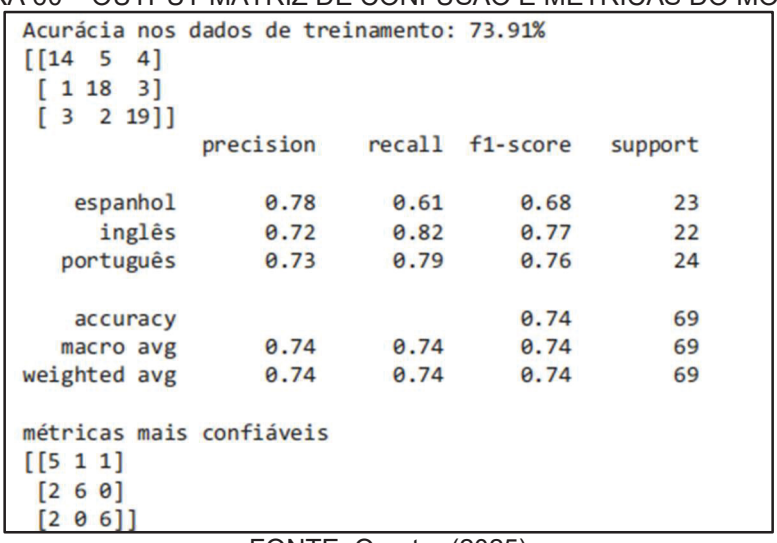
```
from sklearn.model_selection import train_test_split
import numpy as np

#from sklearn.metrics import confusion_matrix

vet = np.array(padroes)
classes = vet[:,-1]         # classes = [p[-1] for p in padroes]
#print(classes)
padroes_sem_classe = vet[:,0:-1]

#print(padroes_sem_classe)
```

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

Com os conjuntos separados, podemos "treinar" o modelo usando a SVM.

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

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

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

# melhor avaliar com a matriz de confusão
y_pred = modelo.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
print(cm)
print(classification_report(y_train, y_pred))

# com dados de teste que não foram usados no treinamento
print('métricas mais confiáveis')
y_pred2 = modelo.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
print(cm)
print(classification_report(y_test, y_pred2))
```

FIGURA 60 – OUTPUT MATRIZ DE CONFUSÃO E MÉTRICAS DO MODELO



```
Acurácia nos dados de treinamento: 73.91%
[[14  5  4]
 [ 1 18  3]
 [ 3  2 19]]
              precision    recall  f1-score   support

    espanhol       0.78      0.61      0.68        23
      inglês       0.72      0.82      0.77        22
   português       0.73      0.79      0.76        24

    accuracy                           0.74        69
   macro avg       0.74      0.74      0.74        69
weighted avg       0.74      0.74      0.74        69

métricas mais confiáveis
[[5 1 1]
 [2 6 0]
 [2 0 6]]
```

FONTE: O autor (2025)

2 Melhore uma base de dados ruim

Import das bibliotecas

```
import requests
import io
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, StratifiedKFold,
cross_val_score
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from collections import Counter
from imblearn.over_sampling import SMOTE
from scipy import stats
```

Carregamento da Base de dados

```
# Colunas do dataset
columns = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']

# URL do arquivo
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data'

# Fazer o download do arquivo
df = pd.read_csv(url, header=None, names=columns)
```

Pré Processamento

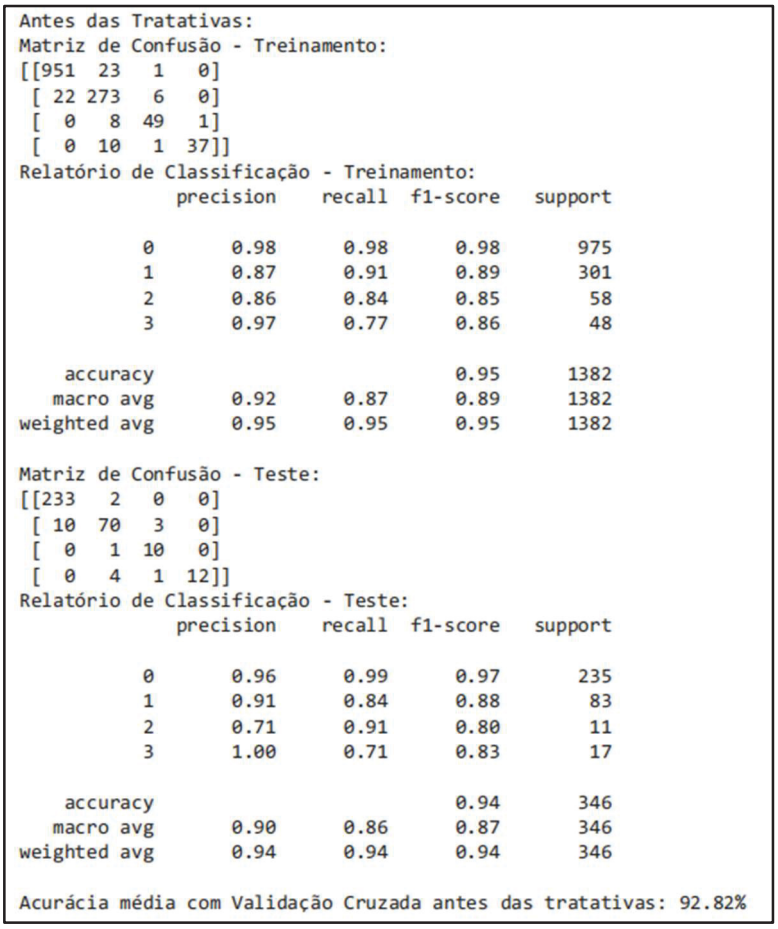
```
# Mapeamento valores categóricos para numéricos
mappings = {
    'buying': {'vhigh': 3, 'high': 2, 'med': 1, 'low': 0},
    'maint': {'vhigh': 3, 'high': 2, 'med': 1, 'low': 0},
    'doors': {'2': 2, '3': 3, '4': 4, '5more': 5},
    'persons': {'2': 2, '4': 4, 'more': 5},
    'lug_boot': {'small': 0, 'med': 1, 'big': 2},
    'safety': {'low': 0, 'med': 1, 'high': 2},
    'class': {'unacc': 0, 'acc': 1, 'good': 2, 'vgood': 3}
}

for col, mapping in mappings.items():
    df[col] = df[col].map(mapping)

# Separando as features (X) e o alvo (Y)
X = df.iloc[:, :-1]
Y = df['class']

# Dividindo os dados em conjuntos de treino e teste
```

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

Análise dos Dados antes das Tratativas

```
# Treinando o modelo SVM antes das tratativas
svm = SVC()
svm.fit(X_train, y_train)

# Previsões antes das tratativas
y_pred_train = svm.predict(X_train)
y_pred_test = svm.predict(X_test)

# Matriz de confusão e relatório de classificação antes das tratativas
print("Antes das Tratativas:")
print("Matriz de Confusão - Treinamento:")
print(confusion_matrix(y_train, y_pred_train))
print("Relatório de Classificação - Treinamento:")
print(classification_report(y_train, y_pred_train))

print("Matriz de Confusão - Teste:")
print(confusion_matrix(y_test, y_pred_test))
print("Relatório de Classificação - Teste:")
print(classification_report(y_test, y_pred_test))

# Validação Cruzada antes das tratativas
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cross_val_scores_before = cross_val_score(svm, X, Y, cv=skf)

print(f"Acurácia média com Validação Cruzada antes das tratativas:
{cross_val_scores_before.mean() * 100:.2f}%")
```

FIGURA 61 – OUTPUT ACURÁCIA DO SVM ANTES DO BALANCEAMENTO COM VALIDAÇÃO CRUZADA



```
Antes das Tratativas:
Matriz de Confusão - Treinamento:
[[951  23   1   0]
 [ 22 273   6   0]
 [  0   8  49   1]
 [  0  10   1  37]]
Relatório de Classificação - Treinamento:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       975
           1       0.87      0.91      0.89       301
           2       0.86      0.84      0.85        58
           3       0.97      0.77      0.86        48

    accuracy                           0.95      1382
   macro avg       0.92      0.87      0.89      1382
weighted avg       0.95      0.95      0.95      1382

Matriz de Confusão - Teste:
[[233   2   0   0]
 [ 10  70   3   0]
 [  0   1  10   0]
 [  0   4   1  12]]
Relatório de Classificação - Teste:
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       235
           1       0.91      0.84      0.88        83
           2       0.71      0.91      0.80        11
           3       1.00      0.71      0.83        17

    accuracy                           0.94       346
   macro avg       0.90      0.86      0.87       346
weighted avg       0.94      0.94      0.94       346

Acurácia média com Validação Cruzada antes das tratativas: 92.82%
```

FONTE: O autor (2025)

Tratativa dos Dados

```
# Remover outliers
z_scores = np.abs(stats.zscore(X_train))
X_train = X_train[(z_scores < 3).all(axis=1)]
y_train = y_train[(z_scores < 3).all(axis=1)]

# Normalizar os dados
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Aplicar SMOTE para balanceamento das classes
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

# Verificar balanceamento das classes após SMOTE
print("Distribuição das classes após balanceamento com SMOTE:")
print(Counter(y_train_smote))
```

Análise dos Dados após as Tratativas

```
# Treinar o modelo SVM após as tratativas
svm.fit(X_train_smote, y_train_smote)

# Previsões após as tratativas
y_pred_train = svm.predict(X_train_smote)
y_pred_test = svm.predict(X_test)

# Matriz de confusão e relatório de classificação após as tratativas
print("\nApós as Tratativas:")
print("Matriz de Confusão - Treinamento:")
print(confusion_matrix(y_train_smote, y_pred_train))
print("Relatório de Classificação - Treinamento:")
print(classification_report(y_train_smote, y_pred_train))

print("Matriz de Confusão - Teste:")
print(confusion_matrix(y_test, y_pred_test))
print("Relatório de Classificação - Teste:")
print(classification_report(y_test, y_pred_test))

# Validação Cruzada
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cross_val_scores = cross_val_score(svm, X_train_smote, y_train_smote, cv=skf)

print(f"Acurácia média com Validação Cruzada: {cross_val_scores.mean() *
100:.2f}%")
```

FIGURA 62 – OUTPUT RESULTADOS DA VALIDAÇÃO CRUZADA DO MODELO SVM

```
Após as Tratativas:
Matriz de Confusão - Treinamento:
[[950  22   3   0]
 [  0 970   3   2]
 [  0   0 975   0]
 [  0   0   0 975]]
Relatório de Classificação - Treinamento:
              precision    recall  f1-score   support

           0       1.00      0.97      0.99       975
           1       0.98      0.99      0.99       975
           2       0.99      1.00      1.00       975
           3       1.00      1.00      1.00       975

    accuracy                           0.99      3900
   macro avg       0.99      0.99      0.99      3900
weighted avg       0.99      0.99      0.99      3900

Matriz de Confusão - Teste:
[[235   0   0   0]
 [  0  76   7   0]
 [  0   0  10   1]
 [  0   0   0  17]]
Relatório de Classificação - Teste:
...
   macro avg       0.88      0.96      0.91       346
weighted avg       0.98      0.98      0.98       346

Acurácia média com Validação Cruzada: 98.95%
```

FONTE: O autor (2025)

```
percentual_diferenca = cross_val_scores.mean() * 100 -
cross_val_scores_before.mean() * 100
print(f"Com as devidas tratativas foi alcançado {percentual_diferenca:.2f}% de
melhoria da acurácia")
```

Com as devidas tratativas foi alcançado 6.12% de melhoria da acurácia

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

Classificação - Veículo

TABELA 2 – CLASSIFICAÇÃO VEÍCULOS

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry= 2	0.7545	Reference Prediction bus opel saab van bus 42 2 0 0 opel 0 23 16 0 saab 0 16 23 1 van 1 1 4 38 Overall Statistics Accuracy : 0.7545 95% CI : (0.682, 0.8177) No Information Rate : 0.2575 P-Value [Acc > NIR] : < 2.2e-16 Kappa : 0.6728 Mcnemar's Test P-Value : NA
RF – CV	mtry=2	0.7485	Reference Prediction bus opel saab van bus 42 2 0 0 opel 0 23 17 0 saab 0 16 22 1 van 1 1 4 38 Overall Statistics Accuracy : 0.7485 95% CI : (0.6756, 0.8123) No Information Rate : 0.2575 P-Value [Acc > NIR] : < 2.2e-16 Kappa : 0.6648 Mcnemar's Test P-Value : NA
RNA – Hold-out	size= 5 decay= 0.1	0.7365	Reference Prediction bus opel saab van bus 39 1 3 2 opel 1 25 15 1 saab 1 13 25 2 van 2 3 0 34 Overall Statistics Accuracy : 0.7365 95% CI : (0.6629, 0.8016) No Information Rate : 0.2575 P-Value [Acc > NIR] : <2e-16 Kappa : 0.6485 Mcnemar's Test P-Value : 0.6574

SVM – CV	C=1 Sigma=0.06883958	0.7365	Reference Prediction bus opel saab van bus 41 0 0 1 opel 0 23 15 3 saab 0 17 25 1 van 2 2 3 34 Overall Statistics Accuracy : 0.7365 95% CI : (0.6629, 0.8016) No Information Rate : 0.2575 P-Value [Acc > NIR] : < 2.2e-16 Kappa : 0.6486 Mcnemar's Test P-Value : NA
SVM – Hold-out	C=1 Sigma=0.06257508	0.7126	Reference Prediction bus opel saab van bus 41 0 1 1 opel 0 22 17 3 saab 0 18 22 1 van 2 2 3 34 Overall Statistics Accuracy : 0.7126 95% CI : (0.6376, 0.7799) No Information Rate : 0.2575 P-Value [Acc > NIR] : < 2.2e-16 Kappa : 0.6167 Mcnemar's Test P-Value : NA
KNN	k=1	0.6412	Reference Prediction bus opel saab van bus 43 2 1 2 opel 1 14 20 1 saab 5 20 13 1 van 2 3 3 39 Overall Statistics Accuracy : 0.6412 95% CI : (0.5642, 0.7132) No Information Rate : 0.3 P-Value [Acc > NIR] : <2e-16 Kappa : 0.5196 Mcnemar's Test P-Value : 0.5438
RNA – CV	size=5 decay=0.1	0.5329	Reference Prediction bus opel saab van bus 30 13 6 1 opel 4 13 22 1 saab 3 14 10 1 van 6 2 5 36 Overall Statistics Accuracy : 0.5329 95% CI : (0.4543, 0.6104) No Information Rate : 0.2575 P-Value [Acc > NIR] : 3.395e-14 Kappa : 0.3781 Mcnemar's Test P-Value : 0.02839

FONTE: O autor (2025)

Melhor Acurácia = Random Forest - Hold-out

Código e dados da técnica de melhor Acurácia

```
# Carregando os dados e retirando a coluna de sequencial
library("caret")
```

```
setwd("C:/Especialização - IA/Máteria 8 - Aprendizado de
Máquina/trabalho/praticas/06 - veiculos")
dados_veiculos <- read.csv("6 - veiculos - Dados.csv")
view("dados_veiculos")
dados_veiculos$a <- NULL
```

```
# Adicionada a semente (202419) e particionando as bases para treinamento e
teste
set.seed(202419)

indices <- createDataPartition(dados_veiculos$tipo, p=0.80, list=FALSE)
treino <- dados_veiculos[indices,]
teste <- dados_veiculos[-indices,]

# Executando a técnica
rf <- train(tipo~., data=treino, method="rf")
rf
```

FIGURA 63 – OUTPUT MODELO RANDOM FOREST TREINADO PARA CLASSIFICAÇÃO

```
> rf <- train(tipo~., data=treino, method="rf")
> rf
Random Forest

679 samples
 18 predictor
  4 classes: 'bus', 'opel', 'saab', 'van'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 679, 679, 679, 679, 679, 679, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.7410313  0.6548054
  10    0.7420783  0.6560449
  18    0.7373219  0.6497332

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 10.
```

FONTE: O autor (2025)

```
# Executando a Matrix de Confusão
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$tipo))
```

FIGURA 64 – OUTPUT MATRIZ DE CONFUSÃO DO MODELO RANDOM FOREST

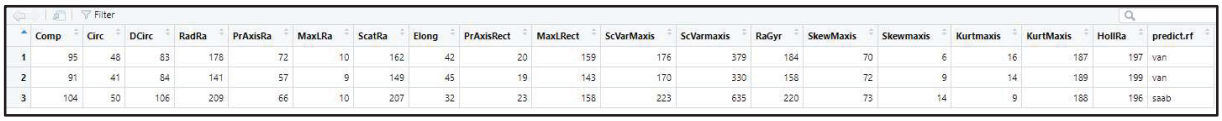
```
> predict.rf <- predict(rf, teste)
> confusionMatrix(predict.rf, as.factor(teste$tipo))
Confusion Matrix and Statistics

          Reference
Prediction bus opel saab van
      bus   41    0    0   0
      opel   0   24   18   0
      saab   0   16   21   1
      van    2    2    4  38

Overall Statistics

               Accuracy : 0.7425
                 95% CI : (0.6692, 0.807)
    No Information Rate : 0.2575
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.657

 Mcnemar's Test P-Value : NA


Statistics by Class:

                     Class: bus Class: opel Class: saab Class: van
Sensitivity              0.9535      0.5714      0.4884      0.9744
Specificity              1.0000      0.8560      0.8629      0.9375
Pos Pred Value           1.0000      0.5714      0.5526      0.8261
Neg Pred Value           0.9841      0.8560      0.8295      0.9917
Prevalence               0.2575      0.2515      0.2575      0.2335
Detection Rate           0.2455      0.1437      0.1257      0.2275
Detection Prevalence     0.2455      0.2515      0.2275      0.2754
Balanced Accuracy        0.9767      0.7137      0.6756      0.9559
```

FONTE: O autor (2025)

```
# Teste com novos casos
dados_novos_casos <- data.frame(
  Comp = c(95,91,104),
  Circ = c(48,41,50),
  DCirc = c(83,84,106),
  RadRa = c(178,141,209),
  PrAxisRa = c(72,57,66),
  MaxLRa = c(10,9,10),
  ScatRa = c(162,149,207),
  Elong = c(42,45,32),
  PrAxisRect = c(20,19,23),
  MaxLRect = c(159,143,158),
  ScVarMaxis = c(176,170,223),
  ScVarmaxis = c(379,330,635),
  RaGyr = c(184,158,220),
```

```
  SkewMaxis = c(70,72,73),
  Skewmaxis = c(6,9,14),
  Kurtmaxis = c(16,14,9),
  KurtMaxis = c(187,189,188),
  HollRa = c(197,199,196),
  tipo = factor(c("?","?","?"))
)

predict.rf <- predict(rf, dados_novos_casos)
dados_novos_casos$tipo <- NULL
resultado <- cbind(dados_novos_casos, predict.rf)
view(resultado)
```

FIGURA 65 – OUTPUT PREVISÕES DO MODELO RANDOM FOREST PARA NOVOS CASOS



	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	predict.rf
1	95	48	83	178	72	10	162	42	20	159	176	379	184	70	6	16	187	197	van
2	91	41	84	141	57	9	149	45	19	143	170	330	158	72	9	14	189	199	van
3	104	50	106	209	66	10	207	32	23	158	223	635	220	73	14	9	188	196	saab

FONTE: O autor (2025)

Classificação - Diabetes

TABELA 3 – CLASSIFICAÇÃO VEÍCULOS

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	C= 0.5 Sigma= 0.1203422	0.7582	Reference Prediction neg pos neg 80 17 pos 20 36 Accuracy : 0.7582 95% CI : (0.6824, 0.8237) No Information Rate : 0.6536 P-Value [Acc > NIR] : 0.003479 Kappa : 0.473 Mcnemar's Test P-Value : 0.742308
RNA – CV	size= 3 decay= 0.1	0.7516	Confusion Matrix and Statistics Reference Prediction neg pos neg 77 15 pos 23 38 Accuracy : 0.7516 95% CI : (0.6754, 0.8179) No Information Rate : 0.6536 P-Value [Acc > NIR] : 0.005891 Kappa : 0.4703 Mcnemar's Test P-Value : 0.256145

SVM – Hold-out	C=0.25 Sigma= 0.1289004	0.7386	Confusion Matrix and Statistics Reference Prediction neg pos neg 81 21 pos 19 32 Accuracy : 0.7386 95% CI : (0.6615, 0.8062) No Information Rate : 0.6536 P-Value [Acc > NIR] : 0.01536 Kappa : 0.4175 Mcnemar's Test P-Value : 0.87437
RF – CV	mtry=5	0.732	> confusionMatrix(predict.rf, as.factor(tes Confusion Matrix and Statistics Reference Prediction neg pos neg 79 20 pos 21 33 Accuracy : 0.732 95% CI : (0.6545, 0.8003) No Information Rate : 0.6536 P-Value [Acc > NIR] : 0.02367 Kappa : 0.4108 Mcnemar's Test P-Value : 1.00000
RF – Hold-out	mtry=2	0.7124	Reference Prediction neg pos neg 78 22 pos 22 31 Accuracy : 0.7124 95% CI : (0.6338, 0.7826) No Information Rate : 0.6536 P-Value [Acc > NIR] : 0.07283 Kappa : 0.3649 Mcnemar's Test P-Value : 1.00000
KNN	k=9	0.6623	Reference Prediction neg pos neg 72 31 pos 21 30 Accuracy : 0.6623 95% CI : (0.5818, 0.7365) No Information Rate : 0.6039 P-Value [Acc > NIR] : 0.07973 Kappa : 0.2737 Mcnemar's Test P-Value : 0.21200
RNA – Hold-out	size= 5 decay= 0.1	0.6471	Reference Prediction neg pos neg 78 32 pos 22 21 Accuracy : 0.6471 95% CI : (0.5658, 0.7225) No Information Rate : 0.6536 P-Value [Acc > NIR] : 0.6037 Kappa : 0.1844 Mcnemar's Test P-Value : 0.2207

FONTE: O autor (2025)

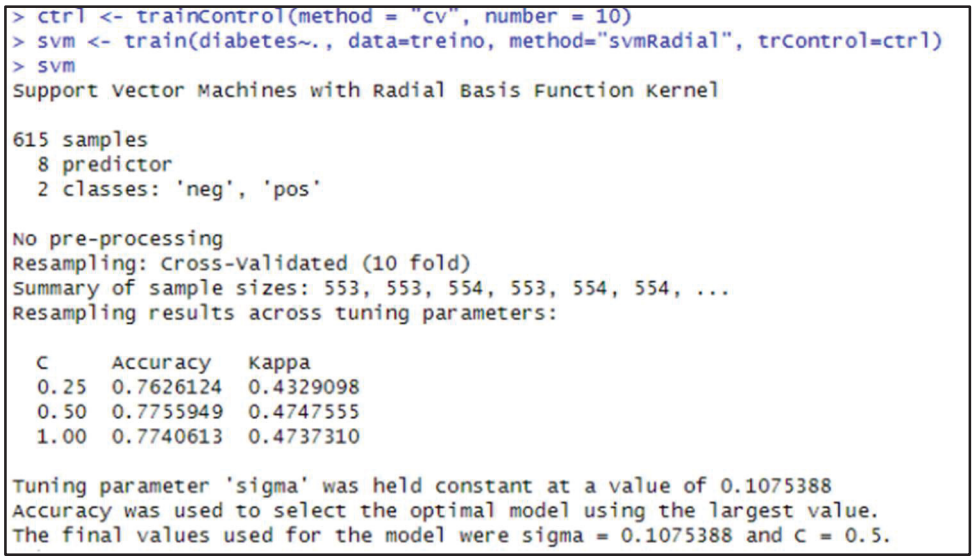
Melhor Acurácia = SVM - Hold-out

Código e dados da técnica de melhor Acurácia

```
# Carregando os dados e retirando a coluna de sequencial
library("caret")

setwd("C:/Especialização - IA/Máteria 8 - Aprendizado de Máquina/Trabalho/10 - Diabetes")
dados_diabete <- read.csv("10 - Diabetes - Dados.csv")
dados <- read.csv("10 - Diabetes - Dados.csv")
view(dados_diabete)
dados_diabete$num <- NULL
```

```
# Adicionada a semente (202419) e particionando as bases para treinamento e teste
set.seed(202419)

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

# Executando a técnica
ctrl <- trainControl(method = "cv", number = 10)

svm <- train(diabetes~., data=treino, method="svmRadial", trControl=ctrl)
svm
```

FIGURA 66 – OUTPUT PREDIÇÃO DE DIABETES COM VALIDAÇÃO CRUZADA



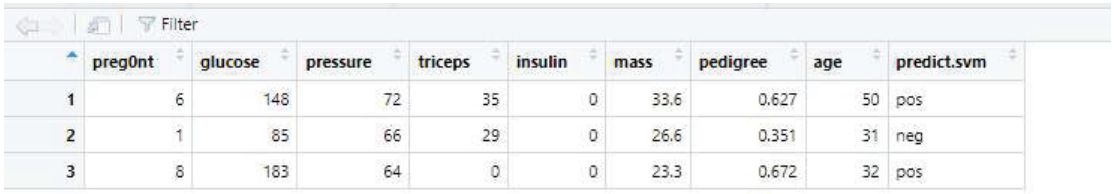
```
> ctrl <- trainControl(method = "cv", number = 10)
> svm <- train(diabetes~., data=treino, method="svmRadial", trControl=ctrl)
> svm
Support Vector Machines with Radial Basis Function Kernel

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

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 553, 553, 554, 553, 554, 554, ...
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.7626124  0.4329098
  0.50  0.7755949  0.4747555
  1.00  0.7740613  0.4737310

Tuning parameter 'sigma' was held constant at a value of 0.1075388
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1075388 and C = 0.5.
```

FONTE: O autor (2025)

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

Teste com novos casos

FIGURA 67 – OUTPUT MATRIX DE CONFUSÃO



	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	predict.svm
1	6	148	72	35	0	33.6	0.627	50	pos
2	1	85	66	29	0	26.6	0.351	31	neg
3	8	183	64	0	0	23.3	0.672	32	pos

FONTE: O autor (2025)

Regressão - Admissão

TABELA 4 – REGRESSÃO ADMISSÃO

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – Holdout	C=1 Sigma= 0.167689	0.8394713	0.05741512	0.916823	0.05682623	0.04210717
SVM – CV	C=1 Sigma= 0.1911262	0.8380984	0.05766011	0.9160269	0.05706871	0.04240472
RF – CV	mtry=2	0.8357089	0.05808405	0.9169273	0.0574883	0.04171365
RF – Holdout	mtry=2	0.8333638	0.05849713	0.9151878	0.05789715	0.0418858
RNA – Holdout	size=5 decay=0.1	0.8255816	0.05984752	0.9107471	0.05923368	0.04521655
RNA – CV	size=3 decay=0.1	0.781962	0.06691385	0.8862512	0.06622754	0.05117317
KNN	K=9	0.7525383	0.07128595	0.870006	0.0705548	0.05412698

FONTE: O autor (2025)

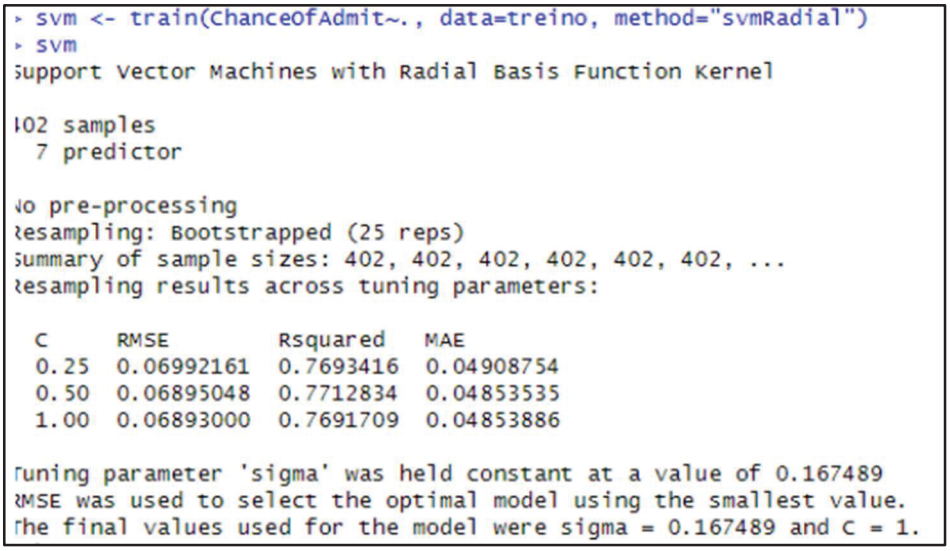
Código e dados da técnica de R2

```
# Carregando os dados, e retirando a coluna de sequencial adicionada a semente
# (202419) e particionando as bases para treinamento e teste
set.seed(202419)
```

```
## Leitura dos dados
setwd("C:/git/IAA008 - Trabalho/regressao/admissao")
dados <- read.csv("9 - Admissao - Dados.csv", header=T)
dados$num <- NULL

## Criar bases de Treino e Teste
indices <- createDataPartition(dados$ChanceofAdmit, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

```
# Executando a técnica
svm <- train(ChanceofAdmit~., data=treino, method="svmRadial")
svm

## Aplicar modelos treinados na base de Teste
predicoes.svm <- predict(svm, teste)
```

FIGURA 68 – OUTPUT PREVISÃO DO MODELO BASE TESTE



```
> svm <- train(ChanceofAdmit~., data=treino, method="svmRadial")
> svm
Support Vector Machines with Radial Basis Function Kernel

402 samples
  7 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 402, 402, 402, 402, 402, 402, ...
Resampling results across tuning parameters:

  C     RMSE        Rsquared   MAE
  0.25  0.06992161  0.7693416  0.04908754
  0.50  0.06895048  0.7712834  0.04853535
  1.00  0.06893000  0.7691709  0.04853886

Tuning parameter 'sigma' was held constant at a value of 0.167489
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.167489 and C = 1.
```

FONTE: O autor (2025)

Execução dos cálculos

```
#Calcula o Coeficiente de Determinação R2
R2 <- 1 - (sum((observado - predito)^2) / sum((observado - mean(observado))^2))
R2
```

[1] 0.8394713

```
# Calcula o Erro Padrão da Estimativa
Sxy <- sqrt(sum((observado - predito)^2) / (length(predito) - 2))
Sxy
```

[1] 0.05741512

```
# Calcula o Erro Médio Absoluto (MAE)
MAE <- mean(abs(observado - predito))
MAE
```

[1] 0.04210717

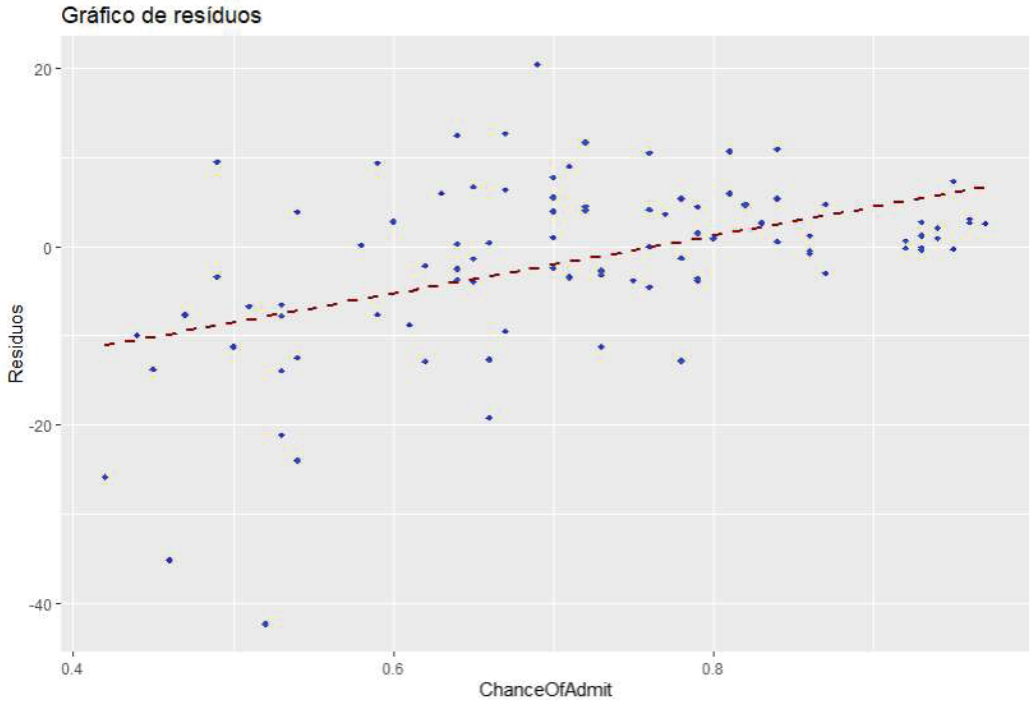
```
# Calcular o Coeficiente de Correlação de Pearson
Pearson <- cor(observado, predito)
Pearson
```

[1] 0.916823

```
# Calcula o Root Mean Square Error (RMSE)
RMSE <- sqrt(mean((observado - predito)^2))
RMSE
```

[1] 0.05682623

GRÁFICO X – GRÁFICO DE RESÍDUOS



FONTE: O autor (2025)

Teste com novos dados

FIGURA 69 – OUTPUT RESULTADO COM NOVOS DADOS

	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predict.svm
1	337	118	4	4.5	4.5	9.65	1	0.9395942
2	324	107	4	4.0	4.5	8.87	1	0.8119879
3	316	104	3	3.0	3.5	8.00	1	0.6610253

FONTE: O autor (2025)

Regressão - Biomassa

TABELA 5 – REGRESSÃO BIOMASSA

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV	C=1 Sigma= 0.7047669	0.9190381	173.8086	0.969495	170.8873	94.38992
KNN	K=1	0.8860589	206.1917	0.9505158	202.7261	89.81617
RF – CV	mtry=2	0.8595157	228.9523	0.9285254	225.1041	90.89986
SVM – Hold-out	C=1 Sigma= 0.8361377	0.8386846	245.3404	0.9436381	241.2167	128.0713
RF – Holdout	mtry=2	0.8464425	239.3683	0.9205069	235.345	94.69088
RNA – Hold-out	size=3 decay=0.1	0.4915887	435.5506	0.9636964	428.2299	262.1295
RNA – CV	size=3 decay=0.4	-2.984452	1219.313	0.3342387	1198.819	379.0955

FONTE: O autor (2025)

Código e dados da técnica de R2

```
# Carregando os dados, e retirando a coluna de sequencial adicionada a semente
# (202419) e particionando as bases para treinamento e teste
set.seed(202419)

## Leitura dos dados
setwd("C:/git/IAA008 - Trabalho/regressao/biomassa")
dados <- read.csv("5 - Biomassa - Dados.csv", header=T)

## Criar bases de Treino e Teste
indices <- createDataPartition(dados$biomassa, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

```
# Executando a técnica
ctrl <- trainControl(method = "cv", number = 10)
svm <- train(biomassa~., data=treino, method="svmRadial", trControl=ctrl)
svm
```

FIGURA 70 – OUTPUT MODELO DE VALIDAÇÃO CRUZADA

```
> ctrl <- trainControl(method = "cv", number = 10)
> svm <- train(biomassa~., data=treino, method="svmRadial", trControl=ctrl)
> svm
Support Vector Machines with Radial Basis Function Kernel

240 samples
  3 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 216, 216, 216, 216, 216, 216, ...
Resampling results across tuning parameters:

  C     RMSE      Rsquared   MAE
  0.25  948.2346  0.6883601  309.9163
  0.50  893.4691  0.7097847  283.9734
  1.00  846.3713  0.7188136  265.9347

Tuning parameter 'sigma' was held constant at a value of 1.341862
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 1.341862 and C = 1.
```

FONTE: O autor (2025)

Execução dos cálculos

```
#Calcula o Coeficiente de Determinação R2
R2 <- 1 - (sum((observado - predito)^2) / sum((observado - mean(observado))^2))
R2
```

[1] 0.9190381

```
# Calcula o Erro Padrão da Estimativa
Sxy <- sqrt(sum((observado - predito)^2) / (length(predito) - 2))
Sxy
```

[1] 173.8086

```
# Calcula o Erro Médio Absoluto (MAE)
MAE <- mean(abs(observado - predito))
MAE
```

[1] 94.38992

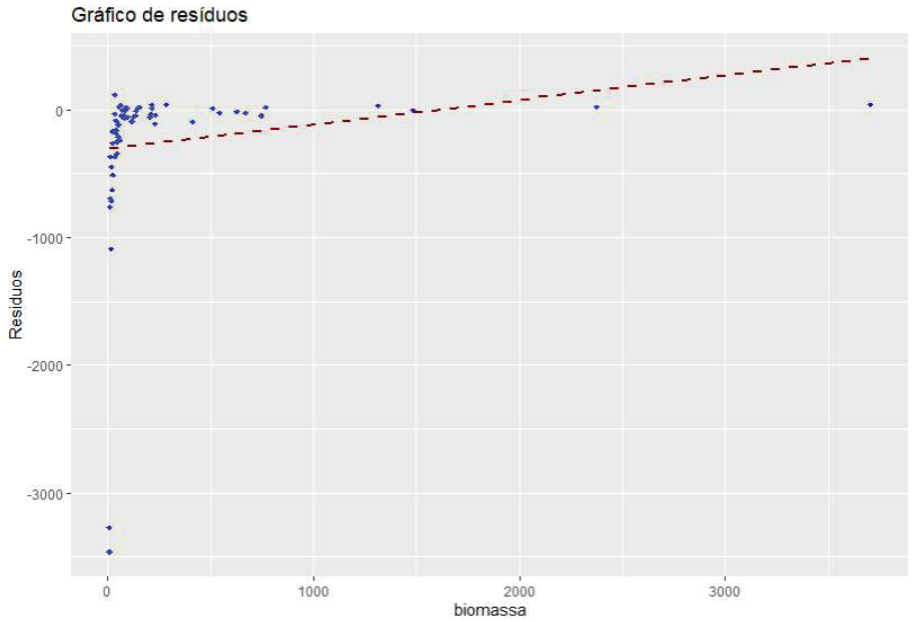
```
# Calcular o Coeficiente de Correlação de Pearson
Pearson <- cor(observado, predito)
Pearson
```

[1] 0.969495

```
# Calcula o Root Mean Square Error (RMSE)
RMSE <- sqrt(mean((observado - predito)^2))
RMSE
```

[1] 170.8873

GRÁFICO 11 – GRÁFICO DE RESÍDUOS



FONTE: O autor (2025)

Teste com novos dados

FIGURA 71 – OUTPUT RESULTADO DE TESTES DE NOVOS DADOS

	dap	h	Me	predict.svm
1	6.4	5.0	1.04	-145.9436
2	7.3	5.0	1.04	-119.5884
3	7.8	5.5	1.04	-114.7724

FONTE: O autor (2025)

Agrupamento - Veículo

```
# Separando em 10 grupos
cluster.results <- kmodes(dados_veiculos_agrupamento, 10, iter.max = 10,
weighted = FALSE )
cluster.results
```

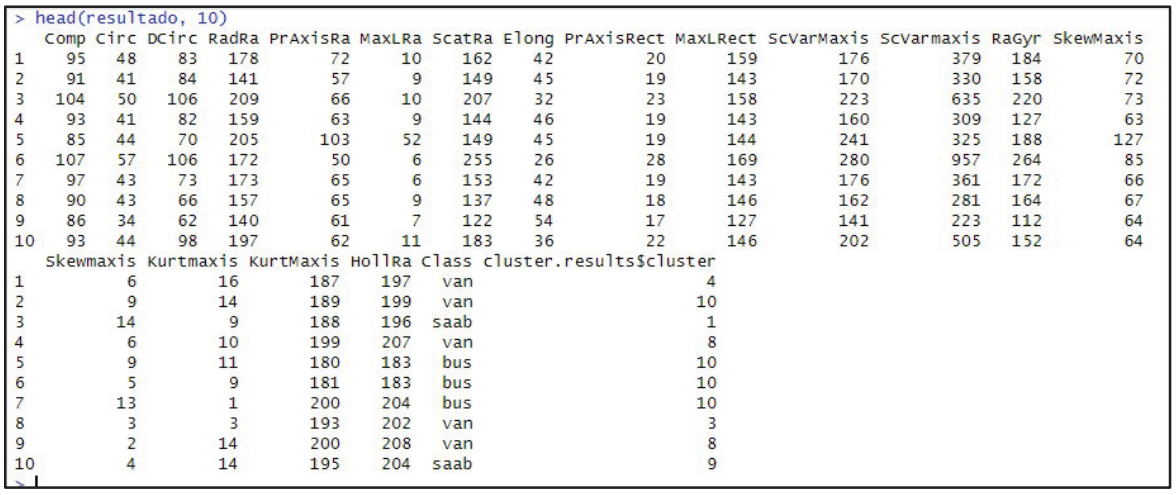
FIGURA 72 – OUTPUT RESULTADO DA SEPARAÇÃO EM GRUPOS DE 10

```
> cluster.results
K-modes clustering with 10 clusters of sizes 85, 71, 124, 92, 96, 62, 85, 81, 61, 89

Cluster modes:
   Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis ScVarmaxis RaGyr SkewMaxis
1   100   51   105   201       61     10    201    32         23      158        217        635   186        72
2    90   47    85   136       64     11    157    43         20      160        173        354   186        75
3    93   37    66   150       59      7    133    50         18      128        159        259   123        62
4   104   54   101   177       57     10    213    31         24      170        219        669   218        74
5    82   43    68   169       64      7    151    44         19      145        175        341   171        80
6    91   39    77   180       60      7    177    37         21      134        197        331   141        72
7   108   53   103   194       63     11    217    31         24      168        229        709   214        71
8    89   45    66   162       54      6    146    46         19      144        168        314   174        64
9    95   45    96   197       62      9    186    35         22      161        202        367   178        66
10   85   42    70   120       56      7    149    45         19      143        170        327   172        85

   Skewmaxis Kurtmaxis KurtMaxis HollRa Class
1          0        19       189    196  saab
2          1         7       183    192   van
3          1         6       201    183   van
4          0        21       187    195  opel
5          2        11       181    184   bus
6          0         7       192    199  saab
7          0        11       188    197  opel
8          2        14       188    196   van
9          1         7       193    198  opel
10         4         1       180    183   bus

Clustering vector:
  [1]  4 10  1  8 10 10 10  3  8  9  3  8  5  8  6  1  8  6  4  4  2  3  5  6  1  8  3  6  9  8  8  7  3  4  4  2
 [37]  5  5  4  5  1  3  2  6  4  2  3  8  2  3  5  1  7 10  5  3  9  2  7  3  9  3  3 10  9  8  5  4  8  4  4  7
 [73]  6  3  5  7  6 10  4 10  5  7  8  5  5  5  8  3 10  3  7  4  1  2  8  4  8  5  1 10  8  8  3  2  9  7  1  6
[109] 10  9  8  8  8  4  8  2  1  7  8  6  3  2  8  4  8  2 10 10  2  1  1  6  9  3  4  2  2  3  3  5  2 10  7 10
[145]  9  4  1 10  8  3  4  5  3  1  8 10 10  3  5  9  3  4  1  3  4  4  1  4  5  3  7  5  8  7  2  7  6  8  5  8
[181]  1  9  5  2  1  1 10  2  1  9  7  8  5  5  1  5  7  5 10 10  2  6  7  7  8  8  9 10 10  4  9  3  5  4  4  9
[217]  5  7 10  8  4  2  9  5  8  8  7  3  1  1  2  4 10  1  3  2  3  5  4  2  9  3 10  7  2  3  5  6  7  6  3  7
[253]  2  5  2  7 10 10  1  4  3  6  8  3  9  5 10  1  6  3  5  6 10  8  5 10 10  3  7  5  9  5  5  7  2  1  8  2
[289]  4  3  3  2 10  3  5  5  9  3  6  6  4  5 10  3  5 10  1  7  9  1  1  2 10  1 10  3  3  9  3  7  4 10 10  1
[325]  6 10  1  8  8  6  3  9  7 10  5  7  8  1  8  5  8  9  3  4  4  4  4  5  8  3  9 10  8  5  6  4 10  1  8  9
[361]  7  1  2  2  8  1  2  3  8  4  6  2  2  2  3  1  7  3 10  7  6  5  5  1  7  2  9  5  2  1 10  3  3 10  6  2
[397]  4  5  7  3  1  3  4  3  9  5  3 10  8  6  8  9 10  9  2  7  9 10  5  3  3  1  3 10  9  9  7  9  1  3  7  1
[433] 10 10  1  3  3 10  6  1  1  2  8  9  5  8  9  4  7  8  6  3  9  5  1  5  2  8  4  6  3  9  9 10  6  5  5  4
[469]  1  3  3  4  5  1  5  1  4  2  9 10  4  4  9  3  2  5  4  6  7  4  7  6 10  4  5  8  7  2  3  4  7  4  3  8
[505]  7  7  1  9  2  4  5  9  4  2  3  3 10  4  5  3  6  3  8  8  7  4  9  4  2  8  6  7  1  3  3  2  4  3  1  5
[541] 10  3  6  6  5  2  3 10  5  7  8  5  6  8  1  6  7  3  5  4  4  4  7  8 10  6  7  6 10  3  7  5  8  2  1  7
[577]  7  9  7  9 10  8  7  4  1  3  7  4  8  4  5  4  5  8  3  7  6  8  9  6  5  3  1  2  9  7  3  3  9  8  1  2
[613]  4  3  6  3  4  3  3  4  7  8  6  7  1  4 10 10  6  3  3  4 10  1  3  3  3 10  6  7  9 10  9  5  5  4  5  8
[649]  7 10  1  8  3  6  6  3 10  2  3  9  3  9  8  1  3  2  9  1  2 10  9  7  6  3  6  1  5 10 10  4  2  5  2 10
[685] 10 10  3 10  5  1  9 10  1  2  1  3  2  5 10  2  7  1  4  3  4  4  5  2  2  9  1  1  1  7  5  4  6 10  4  4
[721]  7  7  4  9  2  3  1  5  3  1  4  2  8  5  3  7  1  7  7  8  6 10  7  1  2  5  3  5  5  1  6  5  1  6  3  6
[757]  1 10 10  4  9  1  3  4  7  5  6  5  4  8 10  1  9 10  3  6  1  5  7  3  1  3  4  4  3  5  7  3  7  6  8  3
[793]  1  7 10  3  7  1  7  6  8  3  7  3  4  5  8  1  5 10  7  4  2  2 10 10  2  7  7  7  5  4  8  4  7  4  9  4
[829]  8  2  5 10  7  6  3  5  8  6  5  3  3  6  2  4  3  3

within cluster simple-matching distance by cluster:
 [1] 1297 1068 1978 1396 1500  983 1273 1257  967 1358
```

FONTE: O autor (2025)

```
# Cria um arquivo com todos os registros e mais os clusters de cada um
resultado <- cbind(dados_veiculos_agrupamento, cluster.results$cluster)
head(resultado, 10)
```

FIGURA 73 – OUTPUT 10 PRIMEIRAS LINHAS DO RESULTADO



```
> head(resultado, 10)
   Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis ScVarmaxis RaGyr SkewMaxis
1    95   48    83   178       72     10    162    42         20      159        176         379   184        70
2    91   41    84   141       57      9    149    45         19      143        170         330   158        72
3   104   50   106   209       66     10    207    32         23      158        223         635   220        73
4    93   41    82   159       63      9    144    46         19      143        160         309   127        63
5    85   44    70   205      103     52    149    45         19      144        241         325   188       127
6   107   57   106   172       50      6    255    26         28      169        280         957   264        85
7    97   43    73   173       65      6    153    42         19      143        176         361   172        66
8    90   43    66   157       65      9    137    48         18      146        162         281   164        67
9    86   34    62   140       61      7    122    54         17      127        141         223   112        64
10   93   44    98   197       62     11    183    36         22      146        202         505   152        64
   Skewmaxis Kurtmaxis KurtMaxis HollRa Class cluster.results$cluster
1          6        16       187    197   van                       4
2          9        14       189    199   van                      10
3         14         9       188    196  saab                       1
4          6        10       199    207   van                       8
5          9        11       180    183   bus                      10
6          5         9       181    183   bus                      10
7         13         1       200    204   bus                      10
8          3         3       193    202   van                       3
9          2        14       200    208   van                       8
10         4        14       195    204  saab                       9
>
```

FONTE: O autor (2025)

Lista de comandos utilizados no RStudio

```
library("caret")
library(klaR)

setwd("C:/Especialização - IA/Matéria 8 - Aprendizado de
Máquina/Trabalho/praticas/11 - Agrupamento - Praticas/11 - Agrupamento -
Praticas - 4 - Veiculos")
dados_veiculos_agrupamento <- read.csv("4 - Veiculos - Dados.csv")
dados <- read.csv("4 - Veiculos - Dados.csv")

View(dados_veiculos_agrupamento)

dados_veiculos_agrupamento$a <- NULL

set.seed(202419)

# Separando em 10 grupos
cluster.results <- kmodes(dados_veiculos_agrupamento, 10, iter.max = 10,
weighted = FALSE )
cluster.results

# Cria um arquivo com todos os registros e mais os clusters de cada um
resultado <- cbind(dados_veiculos_agrupamento, cluster.results$cluster)
head(resultado, 10)
```

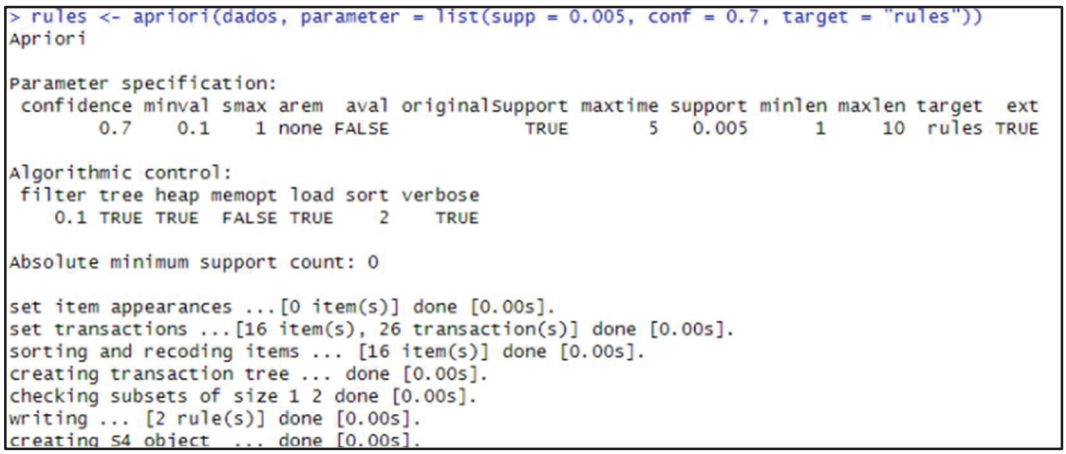
Regras de Associação - Musculação

```
library("caret")
library(arules)
```

```
library(datasets)

setwd("C:/Users/junio/Desktop/Memorial - UFPR/IAA008 - ESPERANDO CODIGO/regras de associacao/musculacao")

dados <- read.transactions("2 - Musculacao - Dados.csv")
```

```
## Definindo regras com o suporte - 0.05 e confiança = 0.7
rules <- apriori(dados, parameter = list(supp = 0.5, conf = 0.9, target = "rules"))
```

FIGURA 74 – OUTPUT DEFINIÇÃO DE REGRAS DE SUPORTE

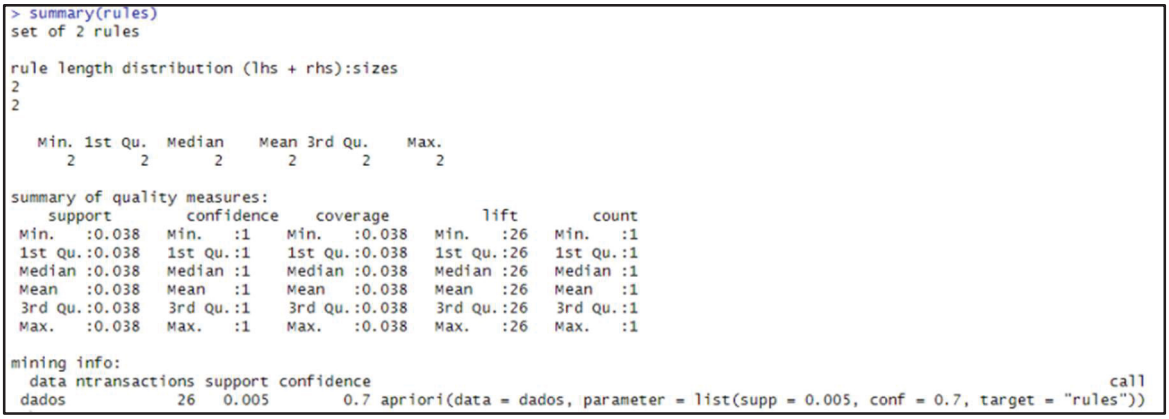


```
> rules <- apriori(dados, parameter = list(supp = 0.005, conf = 0.7, target = "rules"))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.7    0.1    1 none FALSE            TRUE       5   0.005      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 0

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[16 item(s), 26 transaction(s)] done [0.00s].
sorting and recoding items ... [16 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 done [0.00s].
writing ... [2 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

FONTE: O autor (2025)

```
# Resumo de separação das regras
summary(rules)
```

FIGURA 75 – OUTPUT SEPARAÇÃO DE REGRAS



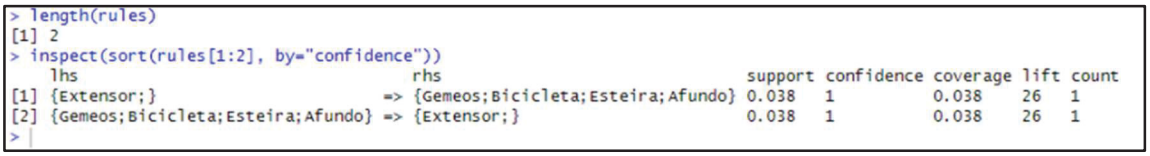
```
> summary(rules)
set of 2 rules

rule length distribution (lhs + rhs):sizes
2
2

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      2       2       2       2       2       2

summary of quality measures:
    support         confidence    coverage          lift       count
 Min.   :0.038   Min.   :1    Min.   :0.038   Min.   :26   Min.   :1
 1st Qu.:0.038   1st Qu.:1    1st Qu.:0.038   1st Qu.:26   1st Qu.:1
 Median :0.038   Median :1    Median :0.038   Median :26   Median :1
 Mean   :0.038   Mean   :1    Mean   :0.038   Mean   :26   Mean   :1
 3rd Qu.:0.038   3rd Qu.:1    3rd Qu.:0.038   3rd Qu.:26   3rd Qu.:1
 Max.   :0.038   Max.   :1    Max.   :0.038   Max.   :26   Max.   :1

mining info:
  data ntransactions support confidence                                                                                call
 dados            26   0.005        0.7 apriori(data = dados, parameter = list(supp = 0.005, conf = 0.7, target = "rules"))
```

FONTE: O autor (2025)

```
# Extraindo as regras
```

```
inspect(sort(rules[1:2], by="confidence"))
```

FIGURA 76 – OUTPUT EXTRAÇÃO DE REGRAS



FONTE: O autor (2025)

APÊNDICE 8 – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

1 Classificação de Imagens (CNN)

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

```
# Carga da base
cifar10 = tf.keras.datasets.cifar10
# Já está separado em dados de treino e teste
```

```
# Não precisa separar
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
# Normalização os dados
# Imagens em pixels de 0 - 255
# / 255.0 transforma em 0 - 1
x_train, x_test = x_train / 255.0, x_test / 255.0
# O dado y é a classe a qual faz parte
# O flattem torna os dados vetorizados
y_train, y_test = y_train.flatten(), y_test.flatten()
# Dimensão dos dados
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)
```

```
# Criando a rede Convulacional
K = len(set(y_train))

# Estágio 1
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
x = Flatten()(x)

# Estágio 2
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)

# Modelo
model = Model(i, x)

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

FIGURA 77 – OUTPUT ARQUITETURA DA REDE

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 32, 32, 3)	0
conv2d_3 (Conv2D)	(None, 15, 15, 32)	896
conv2d_4 (Conv2D)	(None, 7, 7, 64)	18,496
conv2d_5 (Conv2D)	(None, 3, 3, 128)	73,856
flatten_1 (Flatten)	(None, 1152)	0
dropout_2 (Dropout)	(None, 1152)	0
dense_2 (Dense)	(None, 1024)	1,180,672
dropout_3 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 10)	10,250

Total params: 1,284,170 (4.90 MB)
Trainable params: 1,284,170 (4.90 MB)
Non-trainable params: 0 (0.00 B)

FONTE: O autor (2025)

```
# Compilar o modelo
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy",
metrics=["accuracy"])

# Treinar o modelo
r = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=15)
```

FIGURA 78 – OUTPUT TREINO DO MODELO

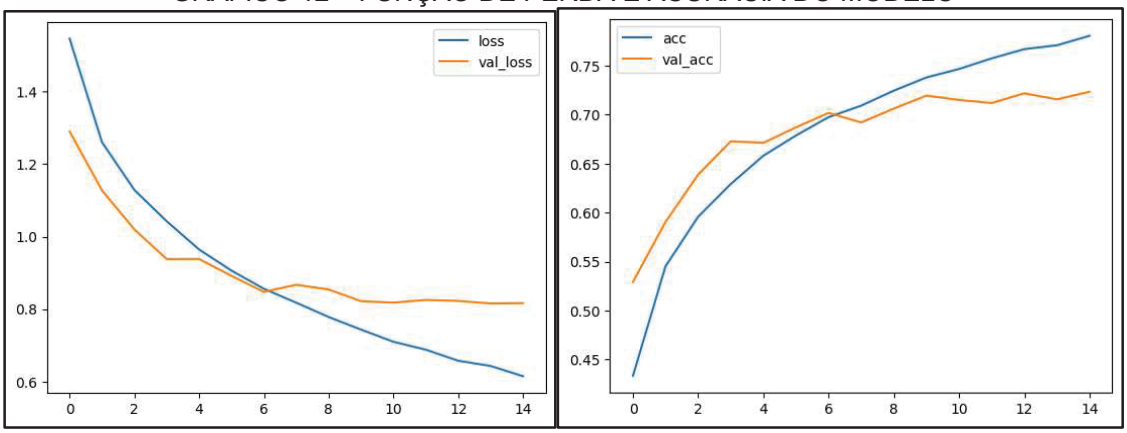
```
Epoch 1/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 7ms/step - accuracy: 0.3475 - loss: 1.7664 - val_accuracy: 0.5314 - val_loss: 1.3173
Epoch 2/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 3ms/step - accuracy: 0.5228 - loss: 1.3180 - val_accuracy: 0.5605 - val_loss: 1.2211
Epoch 3/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.5794 - loss: 1.1725 - val_accuracy: 0.6359 - val_loss: 1.0427
Epoch 4/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6130 - loss: 1.0728 - val_accuracy: 0.6428 - val_loss: 1.0031
Epoch 5/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6468 - loss: 0.9825 - val_accuracy: 0.6644 - val_loss: 0.9497
Epoch 6/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.6709 - loss: 0.9322 - val_accuracy: 0.6803 - val_loss: 0.9207
Epoch 7/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.6966 - loss: 0.8516 - val_accuracy: 0.6781 - val_loss: 0.9183
Epoch 8/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7118 - loss: 0.8140 - val_accuracy: 0.6966 - val_loss: 0.8568
Epoch 9/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7238 - loss: 0.7704 - val_accuracy: 0.7068 - val_loss: 0.8428
Epoch 10/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7382 - loss: 0.7342 - val_accuracy: 0.7139 - val_loss: 0.8434
Epoch 11/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7464 - loss: 0.7142 - val_accuracy: 0.7054 - val_loss: 0.8462
Epoch 12/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7545 - loss: 0.6912 - val_accuracy: 0.7133 - val_loss: 0.8277
Epoch 13/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7690 - loss: 0.6499 - val_accuracy: 0.7202 - val_loss: 0.8091
Epoch 14/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.7763 - loss: 0.6310 - val_accuracy: 0.7105 - val_loss: 0.8310
Epoch 15/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7855 - loss: 0.6063 - val_accuracy: 0.7143 - val_loss: 0.8184
```

FONTE: O autor (2025)

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

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

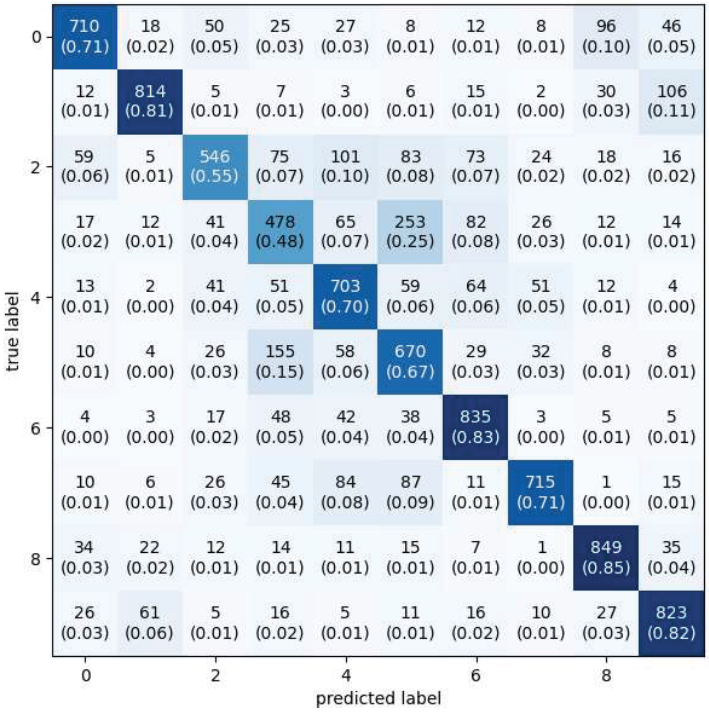
GRÁFICO 12 – FUNÇÃO DE PERDA E ACURÁCIA DO MODELO



FONTE: O autor (2025)

```
# Efetuar predições na base de teste
y_pred = model.predict(x_test).argmax(axis=1)

# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
show_normed=True)
```

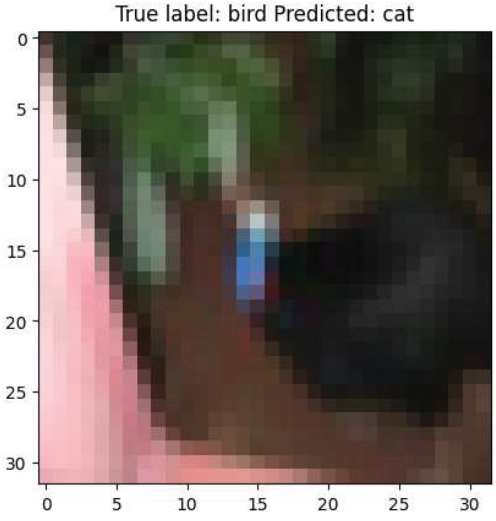
GRÁFICO 13 – MATRIZ DE CONFUSÃO



FONTE: O autor (2025)

```
# Mostrar algumas classificações erradas
labels= ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog",
"horse", "ship", "truck"]
misclassified = np.where(y_pred != y_test)[0]
i = np.random.choice(misclassified)
plt.imshow(x_test[i], cmap="gray")
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]],
labels[y_pred[i]]))
```

IMAGEM 2 – CLASSIFICAÇÃO ERRADA



FONTE: O autor (2025)

2 Detector de SPAM (RNN)

```
# Importação das Bibliotecas
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
```

```
df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values
```

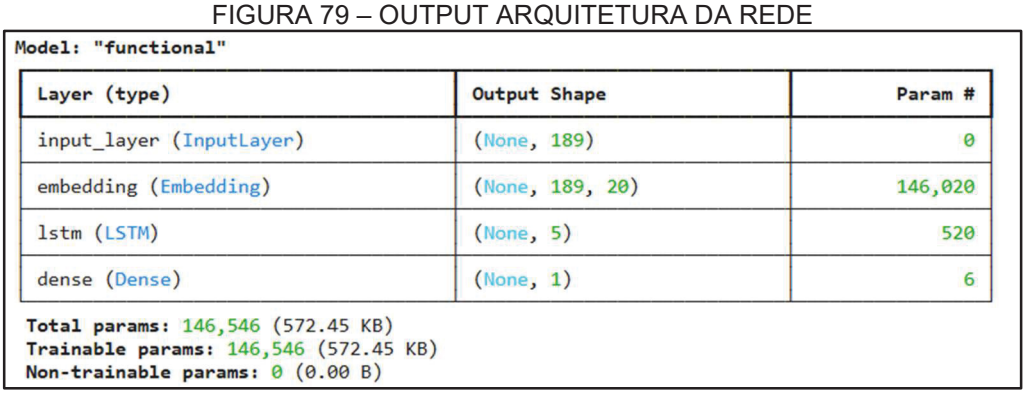
```
# Separa a base em treino e teste
x_train, x_test, y_train, y_test = train_test_split(df["data"], y,
test_size=0.33)
```

```
# Número máximo de palavras para considerar
# São consideradas as mais frequentes, as demais são ignoradas
num_words = 20000 # número de palavras

tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train)
sequences_test = tokenizer.texts_to_sequences(x_test)
word2index = tokenizer.word_index
V = len(word2index)
print("%s tokens" % V)
```

```
# Acerta o tamanho das sequências (padding)
data_train = pad_sequences(sequences_train) # sequências de tamanho padronizado

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

data_test = pad_sequences(sequences_test, maxlen=T)

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

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

model.summary()
```

FIGURA 79 – OUTPUT ARQUITETURA DA REDE



Model: "functional"

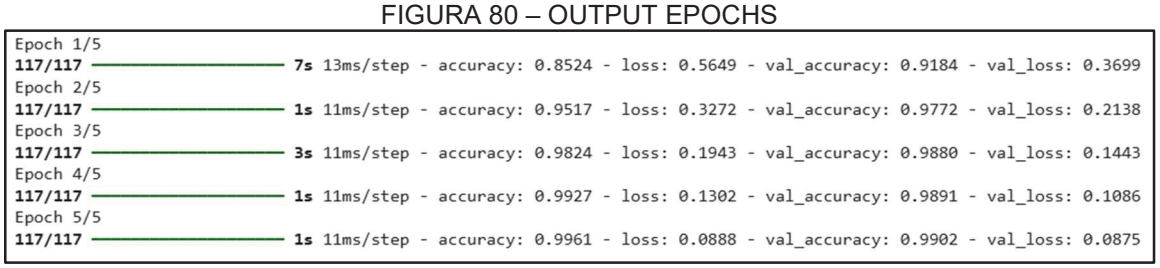
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 189)	0
embedding (Embedding)	(None, 189, 20)	146,020
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6

Total params: 146,546 (572.45 KB)
Trainable params: 146,546 (572.45 KB)
Non-trainable params: 0 (0.00 B)

FONTE: O autor (2025)

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

FIGURA 80 – OUTPUT EPOCHS



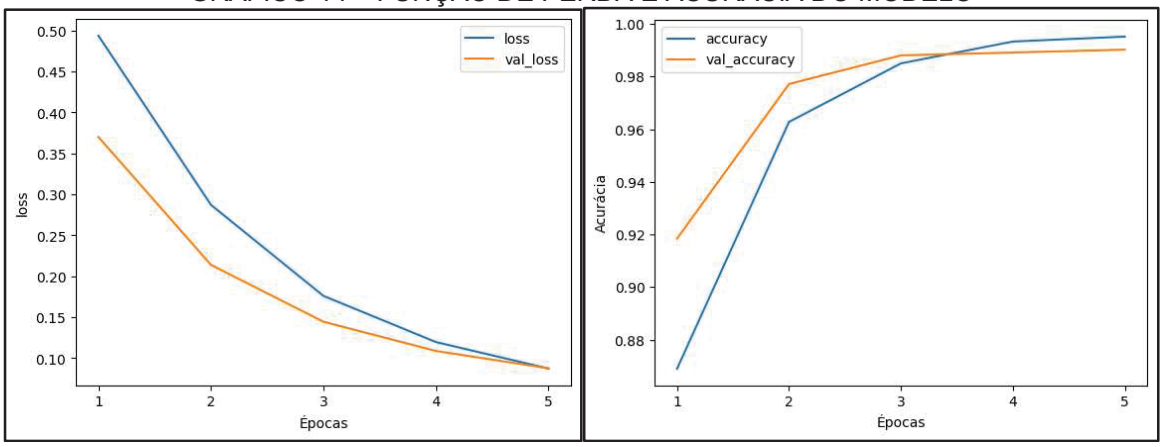
```
Epoch 1/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 7s 13ms/step - accuracy: 0.8524 - loss: 0.5649 - val_accuracy: 0.9184 - val_loss: 0.3699
Epoch 2/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.9517 - loss: 0.3272 - val_accuracy: 0.9772 - val_loss: 0.2138
Epoch 3/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - accuracy: 0.9824 - loss: 0.1943 - val_accuracy: 0.9880 - val_loss: 0.1443
Epoch 4/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.9927 - loss: 0.1302 - val_accuracy: 0.9891 - val_loss: 0.1086
Epoch 5/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.9961 - loss: 0.0888 - val_accuracy: 0.9902 - val_loss: 0.0875
```

FONTE: O autor (2025)

```
# Plota função de perda e acurácia
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.xlabel("Épocas")
plt.ylabel("loss")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
```

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

GRÁFICO 14 – FUNÇÃO DE PERDA E ACURÁCIA DO MODELO



FONTE: O autor (2025)

```
# Efetua a predição de um texto novo
#texto = "Hi, my name is Razer and want to tell you something."
texto = "Is your car dirty? Discover our new product. Free for all. Click the
link."

seq_texto = tokenizer.texts_to_sequences([texto]) # Tokeniza
data_texto = pad_sequences(seq_texto, maxlen=T) # Padding
pred = model.predict(data_texto) # Predição
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")
```

3 Gerador de Dígitos Fake (GAN)

```
# Para Gerar os GIFs
!pip install imageio
!pip install git+https://github.com/tensorflow/docs
```

```
# Importações
import tensorflow as tf
import glob
```

```
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
from tensorflow.keras import layers
import time
from IPython import display
```

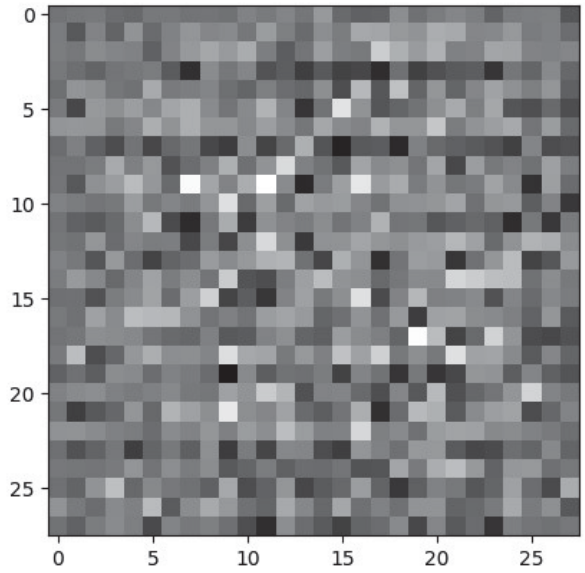
```
# Carregar a base de dados
(train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()

# Normalização
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5 # Normaliza entre [-1, 1]

# Gera o banco em partes e randomiza
BUFFER_SIZE = 60000
BATCH_SIZE = 256

# Cria o dataset (from_tensor_slices)
# Randomiza (shuffle)
# Combina elementos consecutivos em lotes (batch)
train_dataset =
tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).batch(BATC
H_SIZE)
```

```
# Cria o GERADOR
def make_generator_model():
  model = tf.keras.Sequential()

  model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
  model.add(layers.BatchNormalization())
  model.add(layers.LeakyReLU())
  model.add(layers.Reshape((7, 7, 256)))
  assert model.output_shape == (None, 7, 7, 256)

  # Note: None is the batch size
  model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same',
use_bias=False))
  assert model.output_shape == (None, 7, 7, 128)
  model.add(layers.BatchNormalization())
  model.add(layers.LeakyReLU())

  model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same',
use_bias=False))
  assert model.output_shape == (None, 14, 14, 64)
  model.add(layers.BatchNormalization())
```

```
  model.add(layers.LeakyReLU())

  model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same',
use_bias=False, activation='tanh'))
  assert model.output_shape == (None, 28, 28, 1)

  return model
```

```
# Teste do GERADOR, ainda não treinado
generator = make_generator_model()

noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)

plt.imshow(generated_image[0, :, :, 0], cmap='gray')
```

IMAGEM 3 – OUTPUT GERADOR AINDA NÃO TREINADO



FONTE: O autor (2025)

```
# Cria o DISCRIMINADOR
def make_discriminator_model():
  model = tf.keras.Sequential()

  model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
input_shape=[28, 28, 1]))
  model.add(layers.LeakyReLU())

  model.add(layers.Dropout(0.3))

  model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
  model.add(layers.LeakyReLU())
```

```
  model.add(layers.Dropout(0.3))

  model.add(layers.Flatten())

  model.add(layers.Dense(1))

  return model
```

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

```
# Define as funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

# Perda do DISCRIMINADOR
def discriminator_loss(real_output, fake_output):
  real_loss = cross_entropy(tf.ones_like(real_output), real_output)
  fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
  total_loss = real_loss + fake_loss
  return total_loss

# Perda do GERADOR
def generator_loss(fake_output):
  return cross_entropy(tf.ones_like(fake_output), fake_output)
```

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

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

```
# Configura o Loop de treinamento
EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16

# You will reuse this seed overtime (so it's easier)
# to visualize progress in the animated GIF)
```

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

```
# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função
@tf.function
def train_step(images):
  noise = tf.random.normal([BATCH_SIZE, noise_dim])

  with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
    generated_images = generator(noise, training=True)

    real_output = discriminator(images, training=True)
    fake_output = discriminator(generated_images, training=True)

    gen_loss = generator_loss(fake_output)
    disc_loss = discriminator_loss(real_output, fake_output)

  gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
  gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)

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

```
# Treinamento completo/laço
def train(dataset, epochs):
  for epoch in range(epochs):
    start = time.time()
    for image_batch in dataset:
      train_step(image_batch)

    # Produce images for the GIF as you go
    display.clear_output(wait=True)
    generate_and_save_images(generator, epoch + 1, seed)

    # Save the model every 15 epochs
    if (epoch + 1) % 15 == 0:
      checkpoint.save(file_prefix = checkpoint_prefix)

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

  # Generate after the final epoch
  display.clear_output(wait=True)
  generate_and_save_images(generator, epochs, seed)
```

```
# Gerar e salvar imagens
def generate_and_save_images(model, epoch, test_input):
  # Notice `training` is set to False.
  # This is so all layers run in inference mode (batchnorm).
  predictions = model(test_input, training=False)
  fig = plt.figure(figsize=(4, 4))
  for i in range(predictions.shape[0]):
    plt.subplot(4, 4, i+1)
    plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
    plt.axis('off')

  plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
  plt.show()
```

```
# Treinar o modelo e restaurar o último ponto de verificação
train(train_dataset, EPOCHS)

checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))
```

IMAGEM 4 – OUTPUT MODELO TREINADO

FONTE: O autor (2025)

```
# Criar um GIF
# Display a single image using the epoch number
def display_image(epoch_no):
  return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))

display_image(EPOCHS)

anim_file = 'dcgan.gif'

with imageio.get_writer(anim_file, mode='I') as writer:
  filenames = glob.glob('image*.png')
```

```
  filenames = sorted(filenames)
  for filename in filenames:
    image = imageio.imread(filename)
    writer.append_data(image)
  image = imageio.imread(filename)
  writer.append_data(image)

import tensorflow_docs.vis.embed as embed
embed.embed_file(anim_file)
```

4 Tradutor de Textos (Transformer)

```
# Instalação e importação
!pip uninstall tensorflow
!pip install tensorflow==2.15.0x
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0
```

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

```
# Carregar a base de dados
examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en', with_info=True,
as_supervised=True)

train_examples, val_examples = examples['train'], examples['validation']
```

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

  print()
```

```
  for en in en_examples.numpy():
    print(en.decode('utf-8'))
```

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

tf.keras.utils.get_file(f"{model_name}.zip",
f"https://storage.googleapis.com/download.tensorflow.org/models/{model_name}.zip
", cache_dir='.', cache_subdir='', extract=True)

# Tem 2 tokenizers: um pt outro em en
# tokenizers.en tokeniza e detokeniza
tokenizers = tf.saved_model.load(model_name)
```

```
# PIPELINE DE ENTRADA
# Codificar/tokenizar lotes de texto puro
def tokenize_pairs(pt, en):
  pt = tokenizers.pt.tokenize(pt)

  # Converte ragged (irregular, tam variável) para dense
  # Faz padding com zeros.
  pt = pt.to_tensor()

  en = tokenizers.en.tokenize(en)

  # ragged -> dense
  en = en.to_tensor()
  return pt, en
```

```
# Pipeline simpes: processa, embaralha, agrupa os dados, prefetch
# Datasets de entrada terminam com prefetch
BUFFER_SIZE = 20000
BATCH_SIZE = 64

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

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

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

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

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

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

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

```
# CODIFICAÇÃO POSICIONAL
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]

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

```
# Cria uma máscara de 0 e 1, 0 para quando há valor e 1 quando não há
def create_padding_mask(seq):
  seq = tf.cast(tf.math.equal(seq, 0), tf.float32)

  # add extra dimensions to add the padding
  # to the attention logits.
  return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_l)

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

```
# Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):

  # Q K^T
  matmul_qk = tf.matmul(q, k, transpose_b=True) # (..., seq_len_q, seq_len_k)

  # converte matmul_qk para float32
  dk = tf.cast(tf.shape(k)[-1], tf.float32)
```

```
  # divide por sqrt(d_k)
  scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
  if mask is not None:
    # Soma a máscara, e os valores faltantes serão um número próximo a -inf if
mask is not None:
    scaled_attention_logits += (mask * -1e9)

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

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

  return output, attention_weights
```

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

    assert d_model % self.num_heads == 0

    self.depth = d_model // self.num_heads

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

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

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

  def call(self, v, k, q, mask):
    batch_size = tf.shape(q)[0]

    q = self.wq(q) # (batch_size, seq_len, d_model)
    k = self.wk(k) # (batch_size, seq_len, d_model)
    v = self.wv(v) # (batch_size, seq_len, d_model)

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

```
    v = self.split_heads(v, batch_size) # (batch_size, num_heads, seq_len_v,
depth)

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

    # Troca a dimensão 2 com 1, para acertar o num_heads
    # (batch_size, seq_len_q, num_heads, depth)
    scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])

    # Concatena os valores em: (batch_size, seq_len_q, d_model)
    concat_attention = tf.reshape(scaled_attention, (batch_size, -1,
self.d_model))

    output = self.dense(concat_attention) # (batch_size, seq_len_q, d_model)

    return output, attention_weights
```

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

```
class EncoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate=0.1):
    super(EncoderLayer, self).__init__()

    self.mha = MultiHeadAttention(d_model, num_heads)
    self.ffn = point_wise_feed_forward_network(d_model, dff)

    self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)

    self.dropout1 = tf.keras.layers.Dropout(rate)
    self.dropout2 = tf.keras.layers.Dropout(rate)

  def call(self, x, training, mask):
    attn_output, _ = self.mha(x, x, x, mask) # (batch_size, input_seq_len,
d_model)
    attn_output = self.dropout1(attn_output, training=training)
    out1 = self.layernorm1(x + attn_output) # (batch_size, input_seq_len,
d_model)

    ffn_output = self.ffn(out1) # (batch_size, input_seq_len, d_model)
    ffn_output = self.dropout2(ffn_output, training=training)
```

```
    out2 = self.layernorm2(out1 + ffn_output) # (batch_size, input_seq_len,
d_model)

    return out2
```

```
class DecoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate=0.1):
    super(DecoderLayer, self).__init__()

    self.mha1 = MultiHeadAttention(d_model, num_heads)
    self.mha2 = MultiHeadAttention(d_model, num_heads)

    self.ffn = point_wise_feed_forward_network(d_model, dff)

    self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)

    self.dropout1 = tf.keras.layers.Dropout(rate)
    self.dropout2 = tf.keras.layers.Dropout(rate)
    self.dropout3 = tf.keras.layers.Dropout(rate)

  def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    # enc_output.shape == (batch_size, input_seq_len, d_model)

    # (batch_size, target_seq_len, d_model)
    attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
    attn1 = self.dropout1(attn1, training=training)
    out1 = self.layernorm1(attn1 + x)

    # (batch_size, target_seq_len, d_model)
    attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1,
padding_mask)
    attn2 = self.dropout2(attn2, training=training)
    out2 = self.layernorm2(attn2 + out1) # (batch_size, target_seq_len, d_model)

    ffn_output = self.ffn(out2) # (batch_size, target_seq_len, d_model)
    ffn_output = self.dropout3(ffn_output, training=training)
    out3 = self.layernorm3(ffn_output + out2) # (batch_size, target_seq_len,
d_model)

    return out3, attn_weights_block1, attn_weights_block2
```

```
class Encoder(tf.keras.layers.Layer):
  def __init__(self, num_layers, d_model, num_heads, dff, input_vocab_size,
maximum_position_encoding, rate=0.1):
    super(Encoder, self).__init__()
    self.d_model = d_model
    self.num_layers = num_layers
```

```
    self.embedding = tf.keras.layers.Embedding(input_vocab_size, d_model)
    self.pos_encoding = positional_encoding(maximum_position_encoding,
self.d_model)
    self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate) for _ in
range(num_layers)]
    self.dropout = tf.keras.layers.Dropout(rate)

  def call(self, x, training, mask):
    seq_len = tf.shape(x)[1]
    # adding embedding and position encoding.
    x = self.embedding(x) # (batch_size, input_seq_len, d_model)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]
    x = self.dropout(x, training=training)
    for i in range(self.num_layers):
      x = self.enc_layers[i](x, training, mask)

    return x # (batch_size, input_seq_len, d_model)
```

```
class Decoder(tf.keras.layers.Layer):
  def __init__(self, num_layers, d_model, num_heads, dff, target_vocab_size,
maximum_position_encoding, rate=0.1):

    super(Decoder, self).__init__()

    self.d_model = d_model
    self.num_layers = num_layers

    self.embedding = tf.keras.layers.Embedding(target_vocab_size, d_model)
    self.pos_encoding = positional_encoding(maximum_position_encoding, d_model)

    self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate) for _ in
range(num_layers)]
    self.dropout = tf.keras.layers.Dropout(rate)

  def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    seq_len = tf.shape(x)[1]
    attention_weights = {}

    x = self.embedding(x) # (batch_size, target_seq_len, d_model)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]

    x = self.dropout(x, training=training)

    for i in range(self.num_layers):
      x, block1, block2 = self.dec_layers[i](x, enc_output, training,
look_ahead_mask, padding_mask)

      attention_weights[f'decoder_layer{i+1}_block1'] = block1
```

```
      attention_weights[f'decoder_layer{i+1}_block2'] = block2

    # x.shape == (batch_size, target_seq_len, d_model)
    return x, attention_weights
```

```
class Transformer(tf.keras.Model):
  def __init__(self, num_layers, d_model, num_heads, dff, input_vocab_size,
target_vocab_size, pe_input, pe_target, rate=0.1):
    super().__init__()
    self.encoder = Encoder(num_layers, d_model, num_heads, dff,
input_vocab_size, pe_input, rate)

    self.decoder = Decoder(num_layers, d_model, num_heads, dff,
target_vocab_size, pe_target, rate)

    self.final_layer = tf.keras.layers.Dense(target_vocab_size)

  def call(self, inputs, training):
    # Keras models prefer if you pass all your inputs in the first argument
    inp, tar = inputs

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

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

    # dec_output.shape == (batch_size, tar_seq_len, d_model)
    dec_output, attention_weights = self.decoder(tar, enc_output, training,
look_ahead_mask, dec_padding_mask)

    # (batch_size, tar_seq_len, target_vocab_size)
    final_output = self.final_layer(dec_output)

    return final_output, attention_weights

  def create_masks(self, inp, tar):
    # Encoder padding mask
    enc_padding_mask = create_padding_mask(inp)

    # Used in the 2nd attention block in the decoder.
    # This padding mask is used to mask the encoder outputs.
    dec_padding_mask = create_padding_mask(inp)

    # Used in the 1st attention block in the decoder.
    # It is used to pad and mask future tokens in the input received by
    # the decoder.
    look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])
    dec_target_padding_mask = create_padding_mask(tar)
    look_ahead_mask = tf.maximum(dec_target_padding_mask, look_ahead_mask)
```

```
    return enc_padding_mask, look_ahead_mask, dec_padding_mask
```

```
# Hiperparâmetros
num_layers = 4
d_model = 128
dff = 512
num_heads = 8
dropout_rate = 0.1
```

```
class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
  def __init__(self, d_model, warmup_steps=4000):
    super(CustomSchedule, self).__init__()
    self.d_model = d_model
    self.d_model = tf.cast(self.d_model, tf.float32)
    self.warmup_steps = warmup_steps

  def __call__(self, step):
    step = tf.cast(step, tf.float32) # Adicionado para evitar ERRO
    arg1 = tf.math.rsqrt(step)
    arg2 = step * (self.warmup_steps ** -1.5)
    return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

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

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

def accuracy_function(real, pred):
  accuracies = tf.equal(real, tf.argmax(pred, axis=2))
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  accuracies = tf.math.logical_and(mask, accuracies)
  accuracies = tf.cast(accuracies, dtype=tf.float32)
  mask = tf.cast(mask, dtype=tf.float32)
  return tf.reduce_sum(accuracies)/tf.reduce_sum(mask)

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

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

```
# Checkpoint
checkpoint_path = "./checkpoints/train"

ckpt = tf.train.Checkpoint(transformer=transformer, optimizer=optimizer)

ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path, max_to_keep=5)

# if a checkpoint exists, restore the latest checkpoint.
if ckpt_manager.latest_checkpoint:
  ckpt.restore(ckpt_manager.latest_checkpoint)
  print('Latest checkpoint restored!!')
```

```
EPOCHS = 20
train_step_signature = [
  tf.TensorSpec(shape=(None, None), dtype=tf.int64),
  tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]

@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
  tar_inp = tar[:, :-1]
  tar_real = tar[:, 1:]
  with tf.GradientTape() as tape:
    predictions, _ = transformer([inp, tar_inp], training = True)
    loss = loss_function(tar_real, predictions)
  gradients = tape.gradient(loss, transformer.trainable_variables)
  optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))

  train_loss(loss)
  train_accuracy(accuracy_function(tar_real, predictions))
```

```
for epoch in range(EPOCHS):
  start = time.time()
  train_loss.reset_states()
  train_accuracy.reset_states()

  for (batch, (inp, tar)) in enumerate(train_batches):
```

```
    train_step(inp, tar)
    if batch % 50 == 0:
      print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f}
Accuracy {train_accuracy.result():.4f}')

  if (epoch + 1) % 5 == 0:
    ckpt_save_path = ckpt_manager.save()
    print(f'Saving checkpoint for epoch {epoch+1} at {ckpt_save_path}')

    print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}')
    print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')
```

FIGURA 81 – OUTPUT TREINO DO MODELO EPOCHS

```
Epoch 19 Batch 600 Loss 1.4654 Accuracy 0.6769
Epoch 19 Batch 650 Loss 1.4696 Accuracy 0.6763
Epoch 19 Batch 700 Loss 1.4708 Accuracy 0.6762
Epoch 19 Batch 750 Loss 1.4734 Accuracy 0.6759
Epoch 19 Batch 800 Loss 1.4782 Accuracy 0.6751
Epoch 20 Batch 0 Loss 1.4658 Accuracy 0.6825
Epoch 20 Batch 50 Loss 1.3844 Accuracy 0.6905
Epoch 20 Batch 100 Loss 1.3893 Accuracy 0.6890
Epoch 20 Batch 150 Loss 1.3909 Accuracy 0.6886
Epoch 20 Batch 200 Loss 1.4004 Accuracy 0.6872
Epoch 20 Batch 250 Loss 1.4058 Accuracy 0.6868
Epoch 20 Batch 300 Loss 1.4145 Accuracy 0.6853
Epoch 20 Batch 350 Loss 1.4196 Accuracy 0.6847
Epoch 20 Batch 400 Loss 1.4203 Accuracy 0.6845
Epoch 20 Batch 450 Loss 1.4228 Accuracy 0.6840
Epoch 20 Batch 500 Loss 1.4268 Accuracy 0.6831
Epoch 20 Batch 550 Loss 1.4287 Accuracy 0.6827
Epoch 20 Batch 600 Loss 1.4296 Accuracy 0.6827
Epoch 20 Batch 650 Loss 1.4319 Accuracy 0.6824
Epoch 20 Batch 700 Loss 1.4347 Accuracy 0.6818
Epoch 20 Batch 750 Loss 1.4378 Accuracy 0.6811
Epoch 20 Batch 800 Loss 1.4413 Accuracy 0.6806
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-4
Epoch 20 Loss 1.4427 Accuracy 0.6804
Time taken for 1 epoch: 97.46 secs
```

FONTE: O autor (2025)

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

  def __call__(self, sentence, max_length=20):
    # input sentence is portuguese, hence adding the start and end token
    assert isinstance(sentence, tf.Tensor)
    if len(sentence.shape) == 0:
      sentence = sentence[tf.newaxis]
    sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
    encoder_input = sentence

    # as the target is english, the first token to the transformer should be the
    # english start token.
```

```
    start_end = self.tokenizers.en.tokenize([''])[0]
    start = start_end[0][tf.newaxis]
    end = start_end[1][tf.newaxis]
    output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
    output_array = output_array.write(0, start)

    for i in tf.range(max_length):
      output = tf.transpose(output_array.stack())
      predictions, _ = self.transformer([encoder_input, output], training=False)
      predictions = predictions[:, -1:, :] # (batch_size, 1, vocab_size)
      predicted_id = tf.argmax(predictions, axis=-1)
      output_array = output_array.write(i+1, predicted_id[0])
      if predicted_id == end:
        break
    output = tf.transpose(output_array.stack())
    # output.shape (1, tokens)
    text = tokenizers.en.detokenize(output)[0]
    tokens = tokenizers.en.lookup(output)[0]
    _, attention_weights = self.transformer([encoder_input, output[:,:-1]],
training=False)

    return text, tokens, attention_weights
```

```
translator = Translator(tokenizers, transformer)

sentence = "Eu li sobre triceratops na enciclopédia."

translated_text, translated_tokens, attention_weights =
translator(tf.constant(sentence))

print(f'{"Prediction":15s}: {translated_text}')
```

APÊNDICE 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

Estudo de Caso: Implementação de Arquitetura Big Data para Integração de Múltiplas Fontes de Dados

O estudo de caso aborda um projeto real que resolveu problemas de lentidão no processamento de dados, focando em integrar diversas fontes de dados em um ambiente de Big Data.

A implementação priorizou respostas em Near Real-Time para melhorar a entrega de informações. As fontes incluem um banco de dados relacional Oracle, um banco de dados NoSQL MongoDB, planilhas do Google Sheets e arquivos JSON. O uso do Apache Kafka para orquestração da extração desses dados é fundamental para garantir a escalabilidade e a eficiência no transporte de grandes volumes de dados o mais próximo do tempo real.

Caracterização dos Dados e os "Vs" do Big Data

Os dados extraídos de diversas fontes possuem características próprias que se enquadram nas dimensões dos "Vs" do Big Data:

- **Volume:** A integração de dados de diferentes fontes, especialmente de um banco de dados Oracle (estruturado) e MongoDB (não estruturado), além de planilhas e arquivos JSON, implica um grande volume de dados, que precisa ser transportado, armazenado e processado eficientemente.
- **Velocidade:** Como o processamento Near Real-Time é um requisito, a velocidade com que os dados são transportados e integrados é crítica. O Apache Kafka entra como o intermediário para garantir que as diversas fontes de dados sejam consumidas continuamente, e sem interrupções.
- **Variedade:** As fontes de dados variam significativamente em seus formatos, desde bases estruturadas (Oracle), até bases semi-estruturadas (MongoDB) e não estruturadas (JSON). Essa heterogeneidade de dados aumenta a complexidade do processamento e exige um sistema de armazenamento e consulta capaz de lidar com essas variações, como o TrinoDB.
- **Veracidade:** Garantir a qualidade e a confiabilidade dos dados é crucial. Neste contexto, o uso do DBT (Data Build Tool) para transformações dentro do TrinoDB ajuda a normalizar e validar os dados, mantendo sua integridade durante o processo de ETL.

- **Valor:** A combinação de dados extraídos de várias fontes cria uma base rica para insights de negócio, que podem ser explorados por ferramentas de visualização de dados (DataViz), como o PowerBI e o Metabase.

Arquitetura e Ferramentas Empregadas

1. **Apache Kafka para Integração de Dados:**
   O Apache Kafka é usado como sistema de mensagens distribuído para integrar as diversas fontes de dados em Near Real-Time. Cada fonte (Oracle, MongoDB, GSheets, JSON) possui um conector específico no Kafka, que garante que os dados sejam extraídos continuamente e transmitidos para o próximo estágio do pipeline.
2. **Minio para DeepStorage e Apache Iceberg para Metadados:**
   Os dados consumidos pelo Kafka são armazenados em um formato otimizado (Parquet) no Minio, uma solução de armazenamento de objetos compatível com AWS S3. O uso do formato Parquet é estratégico para otimizar o armazenamento e o desempenho na consulta de grandes volumes de dados. O Apache Iceberg atua como um catálogo de metadados, permitindo a gestão eficiente e a rastreabilidade das tabelas de dados no Minio.
3. **TrinoDB como Camada de Consulta:**
   O TrinoDB é utilizado como o motor de consulta para acessar diretamente os dados armazenados no Minio. Esta abordagem permite que os dados sejam processados sem necessidade de movimentação adicional, preservando a eficiência e reduzindo latência. O TrinoDB suporta consultas SQL em dados distribuídos e em diferentes formatos, como Parquet, o que é essencial para garantir a flexibilidade e a performance necessária em um ambiente de Big Data.
4. **DBTpara Transformações e Limpeza de Dados:**
   O DBT é utilizado para as transformações de dados diretamente no TrinoDB, permitindo a criação de pipelines de transformação que normalizam, agregam e limpam os dados. Esta etapa garante que os dados estejam prontos para serem consumidos por diferentes ferramentas de DataViz.
5. **Apache Airflow para Orquestração:**
   A orquestração de todas as etapas, desde a extração de dados via Kafka até o armazenamento e processamento no TrinoDB, é realizada pelo Apache Airflow. Ele permite a automação dos pipelines de ETL, garantindo que as etapas sejam executadas de maneira ordenada e em conformidade com os requisitos de Near Real-Time.

Modelagem de Dados

A modelagem dos dados nesse caso deve ser flexível para lidar com a diversidade de formatos de entrada. No caso do Oracle, um esquema relacional tradicional é usado, com tabelas normalizadas. No MongoDB, os dados seguem uma estrutura de documentos JSON, muitas vezes aninhados.

Durante o processo de transformação (com DBT), esses dados são convertidos para um modelo de dados estruturado que pode ser consumido eficientemente por sistemas de consulta SQL no TrinoDB.

O uso do Apache Iceberg facilita o controle de versões e a evolução do esquema dos dados, essencial quando se lida com fontes de dados dinâmicas e mutáveis, como MongoDB. Já o formato Parquet permite uma eficiente compactação e leitura otimizada, sendo ideal para ambientes de Big Data.

Consumo e Visualização de Dados

Uma vez que os dados são processados e armazenados no Minio, eles podem ser consumidos por ferramentas de visualização de dados como o PowerBI e o Metabase. Essas ferramentas se conectam diretamente ao TrinoDB, que por sua vez acessa os dados no DeepStorage. Esse fluxo permite que dashboards em Near Real-Time sejam criados e atualizados conforme novos dados chegam ao sistema.

Conclusão

Esse estudo de caso demonstra a aplicação de uma arquitetura de Big Data moderna e escalável, integrando múltiplas fontes de dados em um ambiente heterogêneo. O uso de ferramentas como Apache Kafka, TrinoDB, DBT e Apache Airflow permite a criação de um pipeline de dados flexível, eficiente e com suporte a Near Real-Time, possibilitando a entrega rápida de insights valiosos para o negócio através de ferramentas de DataViz.

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) Extração de Características

a) Carregue a base de dados de Treino.

```
# Conectando ao Google Drive
drive.mount('/content/drive')
```

```
# Diretório da origem do arquivo zip no Google Drive
caminho_origem_imagem_treino =
'/content/drive/MyDrive/IAA011/base/Train_Warwick.zip'

# Diretório destino para extração do arquivo
caminho_imagens_treino = '/content/drive/MyDrive/IAA011/imagens/treino'

# Descompactando o arquivo zip
with zipfile.ZipFile(caminho_origem_imagem_treino, 'r') as zip_ref:
    zip_ref.extractall(caminho_imagens_treino)

print("Descompactação concluída!")
```

```
# Separação de pacientes juntamente com os nomes dos arquivos e classes
def separar_pacientes(caminho_imagens):
    imagens = []
    labels = []
    pacientes = []
    nomes_arquivos = []
    for label in os.listdir(caminho_imagens):
        sub_pasta = os.path.join(caminho_imagens, label)
        if os.path.isdir(sub_pasta):
            for nome_arquivo in os.listdir(sub_pasta):
                imagem_caminho = os.path.join(sub_pasta, nome_arquivo)
                imagem = cv2.imread(imagem_caminho)
                if imagem is not None:
```

```
                    imagens.append(imagem)
                    labels.append(label)
                    pacientes.append(nome_arquivo.split('_')[0])
                    nomes_arquivos.append(nome_arquivo)
    return np.array(imagens), np.array(labels), np.array(pacientes),
np.array(nomes_arquivos)
```

```
# Carregando dados de treino
caminho_imagens_treino =
'/content/drive/MyDrive/IAA011/imagens/treino/Train_4cls_amostra'
X, y, pacientes, nomes_arquivos = separar_pacientes(caminho_imagens_treino)
```

b) Crie partições contendo 80% para treino e 20% para validação (atenção aos pacientes).

```
print("Iniciando a separação Estratificada por Grupos (Pacientes)...")

# Definir a estratégia de separação
# Usaremos n_splits=5 para criar uma divisão 80/20 (1/5 = 20% para validação)
sgkf = StratifiedGroupKFold(n_splits=5, shuffle=True, random_state=2)

train_idx, val_idx = next(sgkf.split(X, y, groups=pacientes))

# Aplicação os índices para separar todos os nossos dados
X_train, X_val = X[train_idx], X[val_idx]
y_train, y_val = y[train_idx], y[val_idx]

# É uma boa prática manter o rastreio dos pacientes em cada conjunto também
pacientes_train, pacientes_val = pacientes[train_idx], pacientes[val_idx]

# Verificação (Respondendo diretamente ao feedback do professor)
print("\n--- Verificação da Separação ---")

# Verificação dos números de amostras
print(f"Total de imagens para treino: {len(X_train)}")
print(f"Total de imagens para validação: {len(X_val)}")

# Verificação dos pacientes únicos para garantir que não há sobreposição
treino_pacientes_unicos = np.unique(pacientes_train)
validacao_pacientes_unicos = np.unique(pacientes_val)
print(f"\nNúmero de pacientes para treino: {len(treino_pacientes_unicos)}")
print(f"Número de pacientes para validação: {len(validacao_pacientes_unicos)}")
print(f"Há sobreposição de pacientes? {'Sim' if
np.intersect1d(treino_pacientes_unicos, validacao_pacientes_unicos).size > 0
else 'Não'}")

# Verificação da distribuição de classes (ponto principal da crítica do
professor)
```

```
print("\nDistribuição de classes no conjunto de TREINO:")
train_distribution = pd.Series(y_train).value_counts().sort_index()
print(train_distribution)

print("\nDistribuição de classes no conjunto de VALIDAÇÃO:")
val_distribution = pd.Series(y_val).value_counts().sort_index()
print(val_distribution)

print("\nProporção de cada classe no conjunto de VALIDAÇÃO:")
print(val_distribution / val_distribution.sum())
```

```
# Resumo da Distribuição
nomes_arquivos_train = nomes_arquivos[train_idx]
nomes_arquivos_val = nomes_arquivos[val_idx]

def resumo_distribuicao(labels_treino, labels_val):
    print("Distribuição Treino:", Counter(labels_treino))
    print("Distribuição Validação:", Counter(labels_val))
    df_treino = pd.DataFrame.from_dict(Counter(labels_treino), orient='index',
columns=['Treino'])
    df_val = pd.DataFrame.from_dict(Counter(labels_val), orient='index',
columns=['Validação'])
    display(pd.concat([df_treino, df_val], axis=1).fillna(0))

resumo_distribuicao(y_train, y_val)
```

c) Extraia características utilizando LBP e a CNN VGG16 (gerando um csv para cada extrator).

```
# Função para extração de características utilizando LBP e a CNN VGG16
def extracao_caracteristica_vgg16_eficiente(imagens, model):
    features_vgg = []
    for img in imagens:
        # Redimensionando para o tamanho esperado pelo modelo
        img_resized = cv2.resize(img, (224, 224))
        # Convertendo para array e expandir dimensões para (1, 224, 224, 3)
        img_array = np.expand_dims(img_to_array(img_resized), axis=0)
        # Pré-processamento específico da VGG16 (importante!)
        img_preprocessed = preprocess_input(img_array)
        # Extração das características
        feature = model.predict(img_preprocessed)
        features_vgg.append(feature.flatten())
    return np.array(features_vgg)

def extracao_caracteristica_lbp(imagens):
    lbp_features = []
    for img in imagens:
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        lbp = feature.local_binary_pattern(gray, P=8, R=1, method='uniform')
```

```
        hist, _ = np.histogram(lbp, bins=np.arange(257), density=True)
        lbp_features.append(hist)
    return np.array(lbp_features)
```

```
# Extração de características utilizando LBP e a CNN VGG16 - TREINO
# Carregando o modelo
print("Carregando modelo VGG16 pré-treinado...")
vgg_model_extractor = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3), pooling='avg')
print("Modelo carregado.")

# --- Processando o conjunto de TREINO ---
print("\nProcessando conjunto de TREINO...")

# Extração das características para os dados de treino
lbp_treino_features = extracao_caracteristica_lbp(X_train)
vgg_treino_features = extracao_caracteristica_vgg16_eficiente(X_train,
vgg_model_extractor)

# --- Processando o conjunto de VALIDAÇÃO ---
print("\nProcessando conjunto de VALIDAÇÃO...")

# Extração das características para os dados de validação
lbp_val_features = extracao_caracteristica_lbp(X_val)
vgg_val_features = extracao_caracteristica_vgg16_eficiente(X_val,
vgg_model_extractor)
```

```
# Criação de DataFrames com METADADOS para rastreabilidade - TREINO
df_treino_meta = pd.DataFrame({
    'arquivo': nomes_arquivos_train,
    'paciente': pacientes_train,
    'classe': y_train
})

# Combinação dos metadados e features e salvar em CSV
# LBP
df_treino_lbp_features = pd.DataFrame(lbp_treino_features, columns=[f'LBP_{i}'
for i in range(lbp_treino_features.shape[1])])
df_treino_lbp_final = pd.concat([df_treino_meta, df_treino_lbp_features],
axis=1)
df_treino_lbp_final.to_csv('/content/drive/MyDrive/IAA011/lbp_treino.csv',
index=False)
print("CSV 'lbp_treino.csv' gerado com sucesso.")

# VGG16
df_treino_vgg_features = pd.DataFrame(vgg_treino_features, columns=[f'VGG_{i}'
for i in range(vgg_treino_features.shape[1])])
df_treino_vgg_final = pd.concat([df_treino_meta, df_treino_vgg_features],
axis=1)
```

```
df_treino_vgg_final.to_csv('/content/drive/MyDrive/IAA011/vgg_treino.csv',
index=False)
print("CSV 'vgg_treino.csv' gerado com sucesso.")

# Visualização dos resultados para confirmar
print("\nAmostra do arquivo 'lbp_treino.csv':")
print(df_treino_lbp_final.head())
d) Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos.
```

```
# Criação de DataFrames com METADADOS para rastreabilidade - VALIDACAO
df_val_meta = pd.DataFrame({
    'arquivo': nomes_arquivos_val,
    'paciente': pacientes_val,
    'classe': y_val
})

# Combinações de metadados e features e salvar em CSV
# LBP
df_val_lbp_features = pd.DataFrame(lbp_val_features, columns=[f'LBP_{i}' for i
in range(lbp_val_features.shape[1])])
df_val_lbp_final = pd.concat([df_val_meta, df_val_lbp_features], axis=1)
df_val_lbp_final.to_csv('/content/drive/MyDrive/IAA011/lbp_validacao.csv',
index=False)
print("CSV 'lbp_validacao.csv' gerado com sucesso.")

# VGG16
df_val_vgg_features = pd.DataFrame(vgg_val_features, columns=[f'VGG_{i}' for i
in range(vgg_val_features.shape[1])])
df_val_vgg_final = pd.concat([df_val_meta, df_val_vgg_features], axis=1)
df_val_vgg_final.to_csv('/content/drive/MyDrive/IAA011/vgg_validacao.csv',
index=False)
print("CSV 'vgg_validacao.csv' gerado com sucesso.")

# Visualizando os resultados para confirmar
print("\nAmostra do arquivo 'lbp_validacao.csv':")
print(df_val_lbp_final.head())
```

d) Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos.

```
# Preparação dos dados para o treinamento
X_train_lbp = df_treino_lbp_final.drop(columns=['arquivo', 'paciente',
'classe'])
y_train_labels = df_treino_lbp_final['classe']

X_train_vgg = df_treino_vgg_final.drop(columns=['arquivo', 'paciente',
'classe'])
# y_train_labels é o mesmo para ambos
```

```
# Modelos para features LBP
print("Iniciando o treinamento dos modelos clássicos...")

lbp_rf_model = RandomForestClassifier(random_state=42).fit(X_train_lbp,
y_train_labels)
lbp_svm_model = svm.SVC(random_state=42).fit(X_train_lbp, y_train_labels)
lbp_nn_model = MLPClassifier(max_iter=500, random_state=42).fit(X_train_lbp,
y_train_labels)

print("Treinamento concluído.")
```

```
# Modelos para features VGG16
print("Iniciando o treinamento dos modelos clássicos...")

vgg_rf_model = RandomForestClassifier(random_state=42).fit(X_train_vgg,
y_train_labels)
vgg_svm_model = svm.SVC(random_state=42).fit(X_train_vgg, y_train_labels)
vgg_nn_model = MLPClassifier(max_iter=500, random_state=42).fit(X_train_vgg,
y_train_labels)

print("Treinamento concluído.")
```

e) Carregue a base de Teste e execute a tarefa 3 nesta base.

```
# Diretório da origem do arquivo zip no Google Drive
caminho_origem_imagem_teste =
'/content/drive/MyDrive/IAA011/base/Test_Warwick.zip'

# Diretório destino para extração do arquivo
caminho_imagens_teste = '/content/drive/MyDrive/IAA011/imagens/teste'

# Descompactando o arquivo zip
with zipfile.ZipFile(caminho_origem_imagem_teste, 'r') as zip_ref:
    zip_ref.extractall(caminho_imagens_teste)

print("Descompactação concluída!")
```

```
# Carregando dados de teste
caminho_imagens_teste =
'/content/drive/MyDrive/IAA011/imagens/teste/Test_4cl_amostra'
X_test, y_test, test_pacientes, test_nomes_arquivos =
separar_pacientes(caminho_imagens_teste)
```

f) Aplique os modelos treinados nos dados de treino

```
# Extração de características utilizando LBP e a CNN VGG16 - TESTE
print("Carregando modelo VGG16 pré-treinado...")
```

```
vgg_model_extractor = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3), pooling='avg')
print("Modelo carregado.")

# --- Processando o conjunto de TESTE ---
print("\nProcessando conjunto de TESTE...")

# Extração das características para os dados de treino
lbp_teste_features = extracao_caracteristica_lbp(X_test)
vgg_teste_features = extracao_caracteristica_vgg16_eficiente(X_test,
vgg_model_extractor)
```

```
# Criação de DataFrames com METADADOS para rastreabilidade - TESTE
df_teste_meta = pd.DataFrame({
    'arquivo': test_nomes_arquivos,
    'paciente': test_pacientes,
    'classe': y_test
})

# Combinação de metadados e features e salvar em CSV
# LBP
df_teste_lbp_features = pd.DataFrame(lbp_teste_features, columns=[f'LBP_{i}' for
i in range(lbp_teste_features.shape[1])])
df_teste_lbp_final = pd.concat([df_teste_meta, df_teste_lbp_features], axis=1)
df_teste_lbp_final.to_csv('/content/drive/MyDrive/IAA011/lbp_teste.csv',
index=False)
print("CSV 'lbp_teste.csv' gerado com sucesso.")

# VGG16
df_teste_vgg_features = pd.DataFrame(vgg_teste_features, columns=[f'VGG_{i}' for
i in range(vgg_teste_features.shape[1])])
df_teste_vgg_final = pd.concat([df_teste_meta, df_teste_vgg_features], axis=1)
df_teste_vgg_final.to_csv('/content/drive/MyDrive/IAA011/vgg_teste.csv',
index=False)
print("CSV 'vgg_teste.csv' gerado com sucesso.")

# Visualização do resultado para confirmar
print("\nAmostra do arquivo 'lbp_teste.csv':")
print(df_teste_lbp_final.head())
```

g) Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.

```
# Preparação dos dados para o treinamento - TESTE
X_test_lbp = df_teste_lbp_final.drop(columns=['arquivo', 'paciente', 'classe'])
y_test_labels = df_teste_lbp_final['classe']

X_test_vgg = df_teste_vgg_final.drop(columns=['arquivo', 'paciente', 'classe'])
```

```
# y_test_labels é o mesmo para ambos
```

```
# Função de Avaliação Robusta para Multi-classe
def avaliar_modelo_multiclasse(y_true, y_pred, labels):
    """
    Calcula Sensibilidade (Recall), Especificidade, Precisão e F1-Score
    para cada classe em um problema de classificação multi-classe.
    """
    cm = confusion_matrix(y_true, y_pred, labels=labels)

    metricas = {}
    for i, label in enumerate(labels):
        TP = cm[i, i]
        FP = cm[:, i].sum() - TP
        FN = cm[i, :].sum() - TP
        TN = cm.sum() - (TP + FP + FN)

        # Sensibilidade (Recall ou True Positive Rate)
        sensibilidade = TP / (TP + FN) if (TP + FN) > 0 else 0.0

        # Especificidade (True Negative Rate)
        especificidade = TN / (TN + FP) if (TN + FP) > 0 else 0.0

        # Precisão (Positive Predictive Value)
        precisao = TP / (TP + FP) if (TP + FP) > 0 else 0.0

        # F1-Score
        f1 = 2 * (precisao * sensibilidade) / (precisao + sensibilidade) if
(precisao + sensibilidade) > 0 else 0.0

        metricas[label] = {
            'Sensibilidade': sensibilidade,
            'Especificidade': especificidade,
            'F1-Score': f1
        }

    return metricas, cm
```

```
# Avaliação de todos os modelos e apresentar os resultados
modelos = {
    "Random Forest (LBP)": (lbp_rf_model, X_test_lbp),
    "SVM (LBP)": (lbp_svm_model, X_test_lbp),
    "Rede Neural (LBP)": (lbp_nn_model, X_test_lbp),
    "Random Forest (VGG16)": (vgg_rf_model, X_test_vgg),
    "SVM (VGG16)": (vgg_svm_model, X_test_vgg),
    "Rede Neural (VGG16)": (vgg_nn_model, X_test_vgg),
}

# Obtendo a lista de todas as classes na ordem correta
```

```
class_labels = sorted(y_test_labels.unique())
resultados_finais = {}

print("\n--- Avaliação Detalhada dos Modelos ---")
for nome, (modelo, X_data) in modelos.items():
    y_pred = modelo.predict(X_data)
    metricas, matriz_confusao = avaliar_modelo_multiclasse(y_test_labels,
y_pred, labels=class_labels)

    print(f"\nResultados para: {nome}")
    print("Matriz de Confusão:")
    print(matriz_confusao)

    df_metricas = pd.DataFrame(metricas).T
    print("Métricas por Classe:")
    print(df_metricas)

    # Calculando e armazenando a média (macro) do F1-Score para comparação final
    f1_macro_avg = df_metricas['F1-Score'].mean()
    print(f"F1-Score (Média Macro): {f1_macro_avg:.4f}")
    resultados_finais[nome] = f1_macro_avg
```

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

```
# Indicação do melhor modelo com base na métrica F1-Score (média macro)
print("\n--- Conclusão do Exercício 1 ---")
melhor_modelo = max(resultados_finais, key=resultados_finais.get)
melhor_score = resultados_finais[melhor_modelo]

print(f"O melhor modelo foi '{melhor_modelo}' com um F1-Score (Média Macro) de
{melhor_score:.4f}.")
print("A métrica utilizada para a escolha foi o F1-Score (Média Macro), pois ela
oferece um balanço entre precisão e sensibilidade e considera o desempenho em
todas as classes de forma igualitária.")
```

Output: O melhor modelo foi '**Random Forest (VGG16)**' com um F1-Score (Média Macro) de **0.8075**. A métrica utilizada para a escolha foi o F1-Score (Média Macro), pois ela oferece um balanço entre precisão e sensibilidade e considera o desempenho em todas as classes de forma igualitária.

2) Redes Neurais

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

```
# Treinamento de Modelos VGG16 e Resnet50
print("Definindo Data Generators com pré-processamento correto e augmentation
```

```
conservador...")

# Treino COM Data Augmentation
train_datagen_vgg = ImageDataGenerator(
    preprocessing_function=vgg_preprocess_input,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)
train_datagen_resnet = ImageDataGenerator(
    preprocessing_function=resnet_preprocess_input,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Validação/Treino SEM Data Augmentation
val_datagen_vgg =
ImageDataGenerator(preprocessing_function=vgg_preprocess_input)
val_datagen_resnet =
ImageDataGenerator(preprocessing_function=resnet_preprocess_input)


# Função para construir o modelo
def build_model(base_model_class, num_classes, input_shape=(224, 224, 3)):
    base_model = base_model_class(weights='imagenet', include_top=False,
input_shape=input_shape)
    base_model.trainable = False

    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(1024, activation='relu')(x)
    x = Dropout(0.5)(x)
    predictions = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs=base_model.input, outputs=predictions)
    model.compile(optimizer=Adam(learning_rate=1e-4),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model

# Construção de 4 instancias de modelo
print("\nConstruindo 4 instâncias de modelo distintas...")
vgg_model_aug = build_model(VGG16, num_classes=4)
vgg_model_no_aug = build_model(VGG16, num_classes=4)
resnet_model_aug = build_model(ResNet50, num_classes=4)
resnet_model_no_aug = build_model(ResNet50, num_classes=4)
```

```
print("Modelos criados e compilados com sucesso.")

# Preparação dos dados
print("\nRedimensionando imagens para o formato 224x224...")
def resize_images(images, size=(224, 224)):
    resized_images = [cv2.resize(img, size) for img in images]
    return np.array(resized_images)

# Normalização
X_train_resized = resize_images(X_train)
X_val_resized = resize_images(X_val)
X_test_resized = resize_images(X_test)

print(f"Dimensões dos dados de treino após redimensionamento:
{X_train_resized.shape}")
print("Setup do Exercício 2 concluído e pronto para o treinamento.")
```

c) Aplique os modelos treinados nas imagens da base de Teste

```
# Preparação dos labels e parâmetros de treinamento
y_train_int = y_train.astype(np.int32)
y_val_int = y_val.astype(np.int32)
y_test_int = y_test.astype(np.int32)

EPOCHS = 10
BATCH_SIZE = 32

steps_per_epoch_train = math.ceil(len(X_train_resized) / BATCH_SIZE)
steps_per_epoch_val = math.ceil(len(X_val_resized) / BATCH_SIZE)

# Treinamento dos modelos
print("--- Iniciando treinamento dos modelos ---")

# VGG16 com Data Augmentation
print("\nTreinando VGG16 COM Data Augmentation...")
history_vgg_aug = vgg_model_aug.fit(
    train_datagen_vgg.flow(X_train_resized, y_train_int, batch_size=BATCH_SIZE),
    validation_data=val_datagen_vgg.flow(X_val_resized, y_val_int,
batch_size=BATCH_SIZE),
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch_train,
    validation_steps=steps_per_epoch_val
)

# VGG16 sem Data Augmentation
print("\nTreinando VGG16 SEM Data Augmentation...")
history_vgg_no_aug = vgg_model_no_aug.fit(
    val_datagen_vgg.flow(X_train_resized, y_train_int, batch_size=BATCH_SIZE),
    validation_data=val_datagen_vgg.flow(X_val_resized, y_val_int,
```

```
batch_size=BATCH_SIZE),
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch_train,
    validation_steps=steps_per_epoch_val
)

# ResNet50 com Data Augmentation
print("\Treinando ResNet50 COM Data Augmentation...")
history_resnet_aug = resnet_model_aug.fit(
    train_datagen_resnet.flow(X_train_resized, y_train_int,
batch_size=BATCH_SIZE),
    validation_data=val_datagen_resnet.flow(X_val_resized, y_val_int,
batch_size=BATCH_SIZE),
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch_train,
    validation_steps=steps_per_epoch_val
)

# ResNet50 sem Data Augmentation
print("\nTreinando ResNet50 SEM Data Augmentation...")
history_resnet_no_aug = resnet_model_no_aug.fit(
    val_datagen_resnet.flow(X_train_resized, y_train_int,
batch_size=BATCH_SIZE),
    validation_data=val_datagen_resnet.flow(X_val_resized, y_val_int,
batch_size=BATCH_SIZE),
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch_train,
    validation_steps=steps_per_epoch_val
)

print("\n--- Treinamento concluido! ---")
```

d) Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.

```
# Avaliação dos modelos no conjunto de TESTE
print("\n--- Avaliação final no conjunto de TESTE ---")

modelos_cnn = {
    "VGG16 com Augmentation": vgg_model_aug,
    "VGG16 sem Augmentation": vgg_model_no_aug,
    "ResNet50 com Augmentation": resnet_model_aug,
    "ResNet50 sem Augmentation": resnet_model_no_aug,
}

class_labels = sorted([str(l) for l in np.unique(y_test_int)])
resultados_cnn_finais = {}

for nome, modelo in modelos_cnn.items():
    pred_probs = modelo.predict(X_test_resized)
```

```
    y_pred = np.argmax(pred_probs, axis=1)

    report = classification_report(y_test_int, y_pred,
target_names=class_labels, output_dict=True)

    print(f"\nResultados para: {nome}")
    print(classification_report(y_test_int, y_pred, target_names=class_labels))

    f1_macro_avg = report['macro avg']['f1-score']
    resultados_cnn_finais[nome] = f1_macro_avg
    print(f"F1-Score (Média Macro): {f1_macro_avg:.4f}")

print("\n--- Conclusão do Exercício 2 ---")
```

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

```
# Indicação do melhor modelo
melhor_modelo_cnn = max(resultados_cnn_finais, key=resultados_cnn_finais.get)
melhor_score_cnn = resultados_cnn_finais[melhor_modelo_cnn]

print(f"O melhor modelo foi '{melhor_modelo_cnn}' com um F1-Score (Média Macro)
de {melhor_score_cnn:.4f}.")
print("\nA métrica utilizada para a escolha foi o F1-Score (Média Macro), pois
ela oferece um balanço justo entre precisão e sensibilidade para todas as
classes.")
```

Output: O melhor modelo foi '**VGG16 com Augmentation**' com um F1-Score (Média Macro) de **0.3975**.

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

RESUMO

Este artigo realiza um estudo de caso das implicações éticas do ChatGPT, um modelo de linguagem de inteligência artificial desenvolvido pela OpenAI, com capacidade de gerar textos e dialogar de forma humanizada. Apesar de seu potencial inovador em diversas áreas, o estudo identifica desafios significativos e desafiadores relacionados à disseminação de informações falsas, à privacidade dos usuários e à tomada de decisões automatizadas enviesadas. A falta de transparência nos algoritmos, a coleta massiva de dados e a potencialização de vieses existentes são pontos críticos analisados. A pesquisa inclui uma revisão de literatura e análise de casos específicos, propondo soluções como transparência algorítmica, proteção rigorosa de dados, regulamentação governamental, minimização de vieses, engajamento com stakeholders e educação sobre ética em IA. As considerações finais enfatizam a necessidade de uma abordagem proativa e conjunta para garantir que a IA seja utilizada de forma ética e responsável, para beneficiar a sociedade como um todo.

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

ABSTRACT

This article conducts a case study on the ethical implications of ChatGPT, an artificial intelligence language model developed by OpenAI, capable of generating human-like text and dialogue. Despite its innovative potential in various fields, the study identifies significant and challenging issues related to the dissemination of false information, user privacy, and biased automated decision -making. Critical points analyzed include the lack of transparency in algorithms, mass data collection, and the amplification of existing biases. The research incorporates a literature review and specific case analyses, proposing solutions such as algorithmic transparency, robust data protection, government regulation, bias minimization, stakeholder engagement, and education on AI ethics. The conclusions emphasize the need for a proactive and collaborative approach to ensure that AI is used ethically and responsibly, ultimately benefiting society as a whole.

Keywords: ChatGPT, artificial intelligence, ethics, misinformation, privacy, algorithmic bias.

INTRODUÇÃO

O ChatGPT, modelo de linguagem de inteligência artificial, lançado em 2022 pela OpenAI, revolucionou a comunicação com sua capacidade de conversar como um humano. Sua crescente utilização, embora promissora, levanta sérias preocupações éticas, reforçadas por estudos como o Stanford AI Report.

Nesse estudo, serão exploradas as implicações éticas do uso do ChatGPT, examinando questões críticas como a disseminação de informações, a privacidade dos usuários e a tomada de decisões automatizadas. Além disso, propõe possíveis soluções para lidar com esses dilemas, promovendo uma utilização ética e responsável.

1 REVISÃO DE LITERATURA

1.1 CHATGPT E A ÉTICA

O ChatGPT é um sistema interativo que permite que os usuários tenham conversas usando linguagem natural. De acordo com a OpenAI, a organização que o desenvolveu, o formato de diálogo empregado permite que ele responda a perguntas de acompanhamento, admita erros, desafie premissas incorretas e rejeite solicitações inapropriadas (OpenAI, 2022).

Em apenas dois meses após o lançamento, alcançou cerca de 100 milhões de usuário ativos, se tornando a plataforma com o crescimento de usuários mais rápido da história.

> A explosão que o chat está provocando, diante de todos os hypes tecnológico que o antecederam, é impressionante. Para alguns, isso se dá porque é a primeira vez que uma ferramenta tão poderosa é disponibilizada ao público em geral por meio de uma interface web gratuita e de fácil usabilidade. (SANTAELLA, 2023, p. 2).

Soares, revelou em seu estudo sobre o potencial valioso dessa plataforma para a sociedade, destacando várias vantagens, como: melhoria da comunicação interna, aumento da produtividade e suporte nas tomadas de decisões. Além, de auxiliar em atividades como, programação, criação de conteúdo, redação de textos, análise de dados, e entre diversas outras questões.

Entretanto, diante toda a ascensão da plataforma, e com investimentos altos para fins comerciais e sob o uso frequente de usuários, alguns dos seus pontos negativos começaram a ficar mais em evidência. “Esta tecnologia trouxe não só vantagens para a sociedade, mas também inseguranças e dúvidas” (SOARES, 2023, p. 2).

Velásquez, em sua revisão crítica, cita algumas limitações desse modelo.

Uma delas é a possibilidade de reproduzir preconceitos, já que foi treinada com dados da internet e pode acabar refletindo desigualdades e preconceitos já presentes nesses dados. Outro ponto, é a privacidade, pois informações pessoais e sensíveis dos usuários são coletadas e armazenadas durante a utilização da plataforma.

“As seríssimas questões da ética e da premente necessidade ponderada de regulamentação são temas obrigatórios quando se trata da IA, e não poderia ser diferente nesse Chat que quer se parecer com a gente (SANTAELLA, 2023, p.6) “.

1.2 DILEMAS E IMPLICAÇÕES ÉTICAS

O ChatGPT, utiliza de redes neurais artificiais, utilizando de uma linguagem de processamento natural e aprendizado profundo. No período de desenvolvimento, foi treinado com uma quantidade gigantesca de informações. Porém, isso indica uma limitação referente a quantidade, qualidade, e natureza de suas informações. (SANTAELLA, 2023, p.3)

Em uma avaliação de segurança realizada por pesquisadores internacionais, o ChatGPT e o GPT-4 foram identificados como as plataformas mais propensas a gerar respostas discriminatórias e ofensivas. O estudo também revelou que o ChatGPT possui um preconceito sistemático que favorece um lado político nos Estados Unidos, levantando preocupações sobre a influência nas opiniões e posições políticas dos usuários.

Diante isso, se torna, “crucial promover discussões contínuas e aprofundadas sobre a regulamentação da IA, a fim de garantir que os avanços tecnológicos sejam realizados de forma ética, segura e responsável.” (HELDWEIN, 2023, p.1)

1.2.1 DISSEMINAÇÃO DE INFORMAÇÕES

O ChatGPT baseia as suas respostas referentes aos dados utilizados em seu treinamento, para poder gerar textos de forma mais humanizada e convin cente. No entanto, a falta de transparência sobre os critérios usados em seu algoritmo levanta dúvidas sobre possíveis vieses não tratados, afetando a qualidade e confiabilidade das informações fornecidas. (SAMPAIO et. al, 2023). De acordo com Almeida, Mendonça e Filgueiras (2023), uma das limitações é que as respostas do ChatGPT nem sempre estão corretas, o que pode gerar incertezas para o usuário. Onde, não é possível identificar, se as informações são falsas ou verdadeiras.

> ”Ao não trazer as referências e fontes junto às suas respostas, o ChatGPT e outras IAs generativas dificultam o questionamento de suas respostas, mesmo quando erram ou inventam alguma coisa. Sem uma crítica ou regulamentação deste fato, corremos o risco de nos tornarmos em grande medida dependentes de um falso oráculo que, ao contrário das ciências, não nos permite consultar e questionar suas referências. (CATALANO et. al., 2023)

A própria OpenAI, empresa fundadora, divulgou um relatório extenso de 100 páginas, em 16 de março de 2023, explicando e relatando, que em testes realizados, foi descoberta mentiras nas respostas passadas pelo chat.

A NewsGuard, empresa que realiza a verificações de falsas notícias online, descreveu o chat como a plataforma que mais espalha desinformação na atualidade, após realizar um estudo com a ferramenta envolvendo teoria da conspiração e falsas narrativas.

O ChatGPT pode gerar consequências negativas, visto que, não indica as fontes das informações utilizadas, impedindo que os usuários possam atestar a confiabilidade da fonte e a veracidade dos dados – que são, em geral, extraídos de conteúdos disponíveis na web em seu treinamento. Com o risco de o usuário, gerar ou propagar informações falsas, desatualizadas ou, ainda, incorretas, em razão de serem retiradas de contexto, podendo levar à disseminação da desinformação. (MAAROUF, 2023)

1.2.2 PRIVACIDADE DOS USUÁRIOS

A privacidade dos dados dos usuários é uma das principais preocupações éticas quanto ao uso do ChatGPT, porém a coleta de dados é fundamental para o aprendizado contínuo proposto pelo modelo, porém, se não realizado de forma transparente e segura, pode comprometer informações confidenciais. Conforme afirmado por Soares (2023, p. 2), "como em qualquer outra tecnologia que lida com dados pessoais, há um risco de violação de privacidade dos usuários".

> ”Evidente que uma ausência de verificação adequada e diferenciação dos dados permite com que, dentre os dados inseridos junto à base de dados do ChatGPT, se encontrem também aqueles de cunho pessoal – inclusive, os obtidos sem qualquer consentimento do titular -, incidindo, portanto, reflexos no que diz respeito ao uso indevido destes por parte da IA.” (MAAROUF, 2023, p.61)

Diante disso, surgem preocupações legítimas sobre o armazenamento, processamento e compartilhamento desses dados, em um contexto em que as regulações de proteção de dados variam entre jurisdições.

1.2.3 TOMADA DE DECISÕES ÉTICAS

A aplicação do ChatGPT em processos decisórios críticos, como recrutamento ou concessão de crédito, entre outros, destaca a importância de evitar vieses algorítmicos automáticos. Modelos de IA treinados em dados históricos podem perpetuar e até amplificar discriminações já existentes.

Como exemplo do que ocorreu com a IA de recrutamento criada pela Amazon, que desfavoreceu as inscrições de mulheres, nos processos seletivos, isso ocorreu, pois, ela foi treinada com currículos atuais, onde em sua maior parte é masculina (REUTERS, 2018). Tem-se também o caso do algoritmo COMPAS, que foi desenvolvido com o objetivo de avaliar a probabilidade de reincidência criminal de ex-prisioneiros e auxiliar juízes em seus julgamentos. No entanto, após

investigações, constatou -se que o algoritmo apresentava viés racial, podendo induzir a condenações injustas (VIEIRA, 2019, p.1).

Estudos indicam que, sem uma supervisão adequada, essas tecnologias podem favorecer certos grupos em detrimento de outros, exacerbando desigualdades sociais.

2 METODOLOGIA

Este estudo de caso utilizou uma abordagem qualitativa para investigar as implicações éticas do uso do ChatGPT. A análise dos dados seguiu uma abordagem interpretativa, que incluiu a revisão de literatura relevante e a análise de casos específicos onde o ChatGPT foi implementado. A análise crítica focou em identificar preocupações éticas e propor soluções práticas para mitigar os riscos associados.

3 APRESENTAÇÃO DOS RESULTADOS

3.1 ANÁLISE CRÍTICA

A análise crítica das implicações éticas do ChatGPT revela um paradoxo: ao mesmo tempo que oferece um potencial inovador para diversas áreas, apresenta desafios significativos que exigem atenção. A facilidade de uso e a capacidade de gerar textos convincentes mascaram o risco inerente da desinformação, amplificado pela falta de transparência sobre as fontes utilizadas. A ausência de mecanismos rigorosos de verificação pode perpetuar vieses existentes nos dados de treinamento, resultando em decisões discriminatórias e perpetuando desigualdades sociais.

A coleta massiva de dados, crucial para o funcionamento do sistema, levanta preocupações legítimas sobre a privacidade dos usuários e a necessidade de garantir a proteção de informações sensíveis. Foi constatado que, sem políticas robustas de proteção de dados e técnicas de anonimização, há um risco considerável para a privacidade dos usuários.

Embora a promessa de avanços em áreas como educação, saúde e comunicação seja inegável, os resultados indicam que a implementação ética do ChatGPT exige medidas proativas. É essencial mitigar os riscos associados ao seu uso, promovendo a transparência nos processos algorítmicos e a supervisão humana para garantir que as decisões automatizadas sejam justas e equitativas.

3.2 PROPOSTA DE SOLUÇÕES

3.2.1 POLÍTICAS E REGULAMENTAÇÕES

Desenvolvedores devem estudar e adotar práticas que garantam a ***transparência dos processos algorítmicos***. Isso inclui a explicação clara de como os dados são coletados e utilizados, bem como a implementação de auditorias regulares. A transparência é referência das informações e

fundamental para construir a confiança dos usuários e garantir que os sistemas de IA sejam utilizados de maneira ética.

Sobre a ***proteção dos dados***, a implementação rigorosa de políticas de proteção de dados é essencial. Os desenvolvedores devem garantir que os dados dos usuários sejam anonimizados e que a coleta de informações pessoais seja minimizada, montando também na fase de desenvolvimento arquiteturas seguras e privadas. Seguindo as regulamentações de proteção de dados, como o GDPR na Europa e a LGPD no Brasil, é crucial para proteger a privacidade dos usuários.

Governos devem criar e reforçar ***regulamentações*** específicas para o uso da IA garantindo que essas tecnologias sejam usadas de maneira ética e responsável. Isso inclui a aplicação de penalidades para práticas que violem os direitos dos usuários. A colaboração entre governos, organizações internacionais e desenvolvedores é necessária para estabelecer normas globais de uso ético da IA.

3.2.2 PRÁTICAS DE DESIGN ÉTICO

Desenvolvedores devem trabalhar ativamente para identificar e ***mitigar vieses*** em seus modelos. Isso pode ser feito através de conjuntos de dados mais representativos e técnicas de mitigação de vieses algorítmicos. A diversidade na equipe de desenvolvimento e a inclusão de perspectivas variadas também são importantes para evitar vieses inadvertidos.

Uma opção pode ser também ***engajamento com stakeholders***, onde envolver diversas partes interessadas no processo de desenvolvimento pode ajudar a identificar potenciais problemas éticos e garantir que as soluções adotadas sejam inclusivas e justas. Isso inclui consultar especialistas em ética, representantes de grupos afetados e o público em geral.

Promover a ***educação sobre IA e ética*** tanto para desenvolvedores quanto para usuários pode ajudar a criar uma cultura de uso responsável da tecnologia. Programas de treinamento e campanhas de conscientização podem aumentar a compreensão sobre os riscos e benefícios da IA incentivando práticas éticas e responsáveis.

3.2.3 BALANCEAMENTO DE INTERESSES

As soluções propostas devem buscar um equilíbrio entre inovação tecnológica e proteção dos direitos humanos. A transparência, a justiça e a privacidade devem ser pilares fundamentais na construção e utilização de sistemas de IA. Isso requer uma abordagem multidisciplinar, envolvendo especialistas em tecnologia, ética, direito e políticas públicas.

4 CONSIDERAÇÕES FINAIS

O ChatGPT, como símbolo da rápida evolução da IA, exige uma abordagem ética proativa. A busca por soluções para os desafios aqui apresentados não pode ser postergada. A educação crítica sobre o uso responsável da ferramenta, o desenvolvimento de mecanismos de controle transparentes

e eficazes, a promoção da justiça algorítmica e a proteção robusta dos dados dos usuários são cruciais para garantir que a promessa de um futuro impulsionado pela IA seja, de fato, um futuro positivo para toda a humanidade.

Ignorar os dilemas éticos apresentados pelo ChatGPT seria negligenciar nossa responsabilidade em moldar um futuro tecnológico mais justo e equitativo. A sinergia entre a inovação tecnológica e a reflexão ética é o caminho para que a inteligência artificial cumpra seu potencial de transformar positivamente a sociedade. Promover discussões contínuas e aprofundadas sobre a regulamentação da IA é essencial para assegurar que os avanços tecnológicos ocorram de maneira ética, segura e responsável, beneficiando a sociedade como um todo.

REFERÊNCIAS

ALMEIDA, V.; MENDONÇA, R.; FILGUEIRAS, F. **ChatGPT: tecnologia, limitações e impactos**. Ciência Hoje, março 2023 [CH 396]. Disponível em: https://cienciahoje.org.br/artigo/chatgpt-tecnologia-limitacoes-e-impactos/. Acesso em: 1 jul. 2024.

CATALANO, José Victor Rodrigues; LORENZO, Bruno Rossi. **Sem referências: o ChatGPT sob a perspectiva latouriana do duplo clique.** Faz Ciência, v. 25, n. 41,

p. 38-58, jan./jun. 2023. Disponível em: https://e-revista.unioeste.br/index.php/fazciencia/article/view/30761/21917. Acesso em: 03 jul. 2024.

CENTRO DE ÉTICA vê ChatGPT como preconceituoso e enganoso e denuncia nos EUA. *Exame.com*, 2023. Disponível em: https://exame.com/future-of- money/centro-etica-chatgpt-preconceituoso-enganoso-denuncia-eua/. Acesso em: 02 jul. 2024.

CHATGPT já impacta na confiabilidade das notícias e nas fake news. *ContraPonto Digital*, 2024. Disponível em: https://contrapontodigital.pucsp.br/noticias/chatgpt-ja-impacta-na-confiabilidade-das-noticias-e-nas-fake-news#:~:text=Segundo%20uma%20pesquisa%20da%20organização,em%20aproximadamente%2080%25%20das%20vezes. Acesso em: 02 jul. 2024.

HELDWEIN, Flávio Lobo; ALMEIDA, Silvio Henrique Maia de. ChatGPT na publicação científica – A Era da IA chegou: oportunidades, desafios e ética. **Recet: Revista de Ensino e Treinamento da Sociedade Brasileira de Urologia**, v. 10, n. 1, p. 1-8, 2023. Disponível em: https://doi.org/10.55825/recet.sbu.0164. Acesso em: 20 jun.2024.

MAAROUF, Ana Clara Reolon. (2023). **Os reflexos causados pelo uso do ChatGPT** . Jornal da Universidade. Disponível em: https://www.ufrgs.br/jornal/os- reflexos-causados-pelo-uso-do-chatgpt/. Acesso em: 30 jun. 2024

MAAROUF, Ana Clara Reolon. **A responsabilidade civil pelo uso do ChatGPT: Uma análise dos reflexos jurídicos causados pela utilização da inteligência artificial.** p.80. Trabalho de Conclusão de Curso (Graduação em Direito) - Universidade Federal do do Rio Grande do Sul, Porto Alegre, 2023.

CATALANO, J. V.; ROSSI LORENZI, B. **Sem Referências: o ChatGPT sob a perspectiva latouriana e a armadilha do Duplo Clique**. Revista Faz Ciência, [S. l.], v. 25, n. 41, 2023. DOI: 10.48075/rfc.v25i41.30761. Disponível em: https://e-revista.unioeste.br/index.php/fazciencia/article/view/30761. Acesso em: 2 jul. 2024.

VELÁSQUEZ, Rodrigues. **O ChatGPT na pesquisa em Humanidades Digitais: Oportunidades, críticas e desafios**. TEKOA, [S. l.], v. 2, n. 2, 2023. Disponível em: https://revistas.unila.edu.br/tekoa/article/view/3711. Acesso em: 1 jul. 2024.

SAMPAIO, R. C., Nicolás, M. A., Junquilho, T. A., Silva, L. R. L., Freitas, C. S., Telles, M., & Teixeira, J. S. (2023). **ChatGPT and other AIs will change all scientific**

SANTOS, Lillian. **Tudo o que você precisa saber sobre o ChatGPT da OpenAI. Forbes Brasil.** São Paulo, 08 dez. 2022. Disponível em: https://forbes.com.br/forbes-tech/2023/01/o-que-e-chatgpt-e-como-ela-pode-ser-util- no-dia-a-dia/. Acesso em: jun. 2024.

SOARES, M. (2023). **Impacto do Chat GPT na sociedade**. The Trends Hub, (3). https://doi.org/10.34630/tth.vi3.5080. Acesso em: 29 jun. 2024.

Stanford University. **AI Index**. Disponível em: https://aiindex.stanford.edu/report/. Acesso em: 29 jun. 2024

OpenAI. (2022). **Introducing ChatGPT** . OpenAI. Disponível em: https://openai.com/index/chatgpt/. Acesso em: 30 jun. 2024

REUTERS, G1. (2018). **Amazon desiste de ferramenta secreta de recrutamento que mostrou viés contra mulheres**. Época Negócios, G1, 10 out. 2018. Disponível em: https://epocanegocios.globo.com/Empresa/noticia/2018/10/amazon -desiste-de- ferramenta-secreta-de-recrutamento-que-mostrou-vies-contra-mulheres.html. Acesso em: 2 jul. 2024

VIEIRA, Leonardo Marques. (2019). **A PROBLEMÁTICA DA INTELIGÊNCIA ARTIFICIAL E DOS VIESES ALGORÍTMICOS: CASO COMPAS**. Brazilian Technology Symposium, Campinas, São Paulo. ISSN 2447-8326. V.1. Disponível em: https://lcv.fee.unicamp.br/images/BTSym-19/Papers/090.pdf. Acessado em: 2 jul.2024

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Nome do artigo escolhido:

O pipeline para o desenvolvimento contínuo de modelos de inteligência artificial - Estado atual da pesquisa e prática.

Qual o **objetivo** do estudo descrito pelo artigo?	Qual o **problema/oportunidade/situação** que levou à necessidade de realização desse estudo?	Qual a **metodologia** que os autores usaram para obter e analisar as informações do estudo?	Quais os **principais resultados** obtidos pelo estudo?
O objetivo do estudo foi em entender as principais dificuldades envolvidas no desenvolvimento de modelos de Inteligência Artificial e propor soluções para reduzir a complexidades associadas a esse processo. A pesquisa apresenta uma proposta de implementação de práticas de DevOps aplicadas para IA, também abordando tópicos como Continuos Integration (CI), Continuos Delevery (CD) no contexto de IA, além de MLOps, gestão do ciclo de vida end-to-end, CD4ML (Continuos	As empresas enfrentam dificuldades significativas ao tentar implementar soluções baseadas em modelos de Inteligência Artificial (IA), devido ao grande volume de dados, as complexidades dos algoritmos e as exigências de qualidade e desempenho dos modelos. Uma solução viável para superar as dificuldades são a implementação dos pipelines para IA, que possibilitam a automação e a integração contínua de processos, como treinamento, validação e a implementação dos modelos. Essa abordagem oferece uma oportunidade para realizar entregas mais ágeis, com maior frequência e com mais qualidade, assim garantindo maior eficiência no ciclo de vida dos modelos de IA.	Na pesquisa foram três metodologias, como MLR (Meta-Literature Review), Estratégia de criação de taxinomia e Análise Qualitativa. MLR foi utilizada para representar uma visão abrangente e evidencias de perspectiva relevantes que são para o pipeline de desenvolvimento contínuo de IA. A taxinomia foi empregada para categorizar os dados e atribuir as informações a cada categoria, utilizando uma abordagem qualitativa com base nas fases do DevOps. Na estruturação foi	Os resultados apresentados e alinhados aos objetivos dos estudos foram uma abordagem de práticas como DevOps, Continuos Integration/Continous Delevery (CI/CD) para Inteligência Artificial (IA), MLOps (Machine Learning Operations) e o gerenciamento de ciclo de vida end-to-end, incluindo o conceito de CD4ML (Continuos Delevery for Machine Learning). Os gatilhos potencias foram fundamentais para fornecer feedback, gerar alertas, agendamento de serviços, atualizar repositórios e permitir os acionamentos manuais. Esses gatilhos exploraram os desafios enfrentados nos

Delevery for Machine Learning), a implementação de sistemas de feedback e alertas, serviços de orquestração, gatilhos automatizados e pipelines de desenvolvimento. O estudo destaca a importância de boas práticas em diversas etapas, como gerenciamento de repositórios, manipulação e processamento de dados, controle de qualidade dos dados, treinamento e aprendizado do modelo, além do gerenciamento de documentação e controle de versão. A pesquisa também discute estratégias de desenvolvimento de software para IA, planejamento da implementação da solução e monitoramento contínuo no ambiente produtivo, considerando a manipulação e a configuração da infraestrutura necessária para suportar a operação de forma eficiente.		adotado uma abordagem de faceta, por causa que os pipelines de desenvolvimento contínuo para soluções em IA são algo novos. Analise qualitativa foi utilizada para verificar se as informações da literatura são corretas e abrangentes o suficiente para descrever o conhecimento existente.	diferentes estágios de um pipeline, que incluem a manipulação de dados, o treinamento dos modelos, o desenvolvimento de software e a operação de sistemas. Cada um desses estágios envolve tarefas cruciais, como o pré-processamento de dados, controle de versão, o design de modelos, o treinamento e a implementação.

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 Classificação (RNA)

```
# Importar dados
data = tf.keras.datasets.fashion_mnist
```

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

```
# Visualização de como ficou a configuração  das variáveis
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", y_test.shape)
print("y_test.shape: ", y_test.shape)
```

```
# Visualizar variável de treino
display(x_train)
```

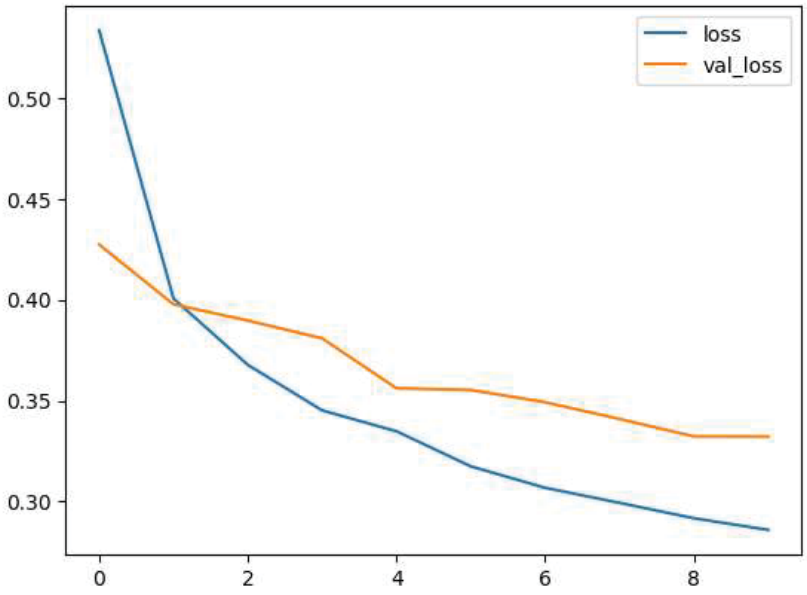
```
# Pré processamento
x_train, x_test = x_train/255.0, x_test/255.0
```

```
# Criando o modelo
i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)

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

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

r = model.fit(x_train,
              y_train,
              validation_data=(x_test, y_test),
              epochs=10)
```

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

GRÁFICO 15 – OUTPUT FUNÇÃO DE PERDA CLASSIFICAÇÃO

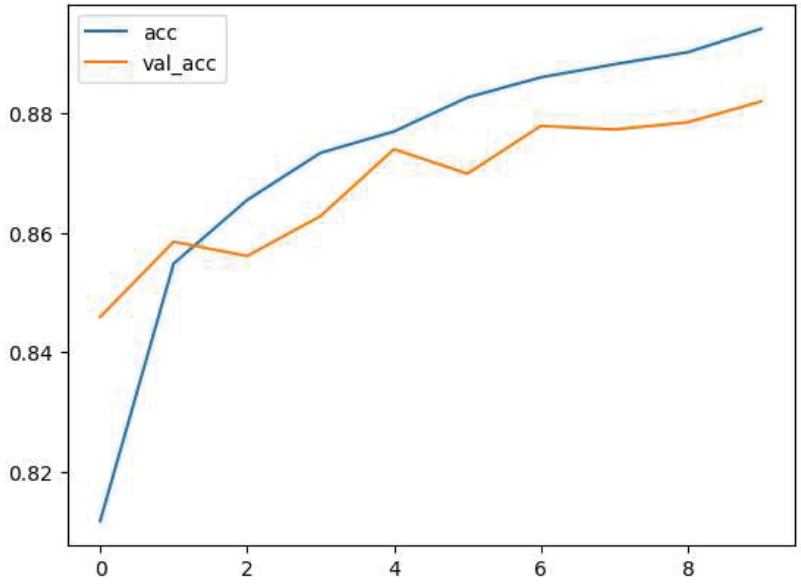


FONTE: O autor (ano)

Gráfico de Perda: Conforme o andamento das Epochs, foi abaixando o valor de 5 em 5. Os dados de teste, baixaram bem mais que os dados de validação. E, então, seria possível usar um pouco mais de Epochs para ficar ainda mais baixo.

```
# Plotar a acurácia
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
```

GRÁFICO 16 – OUTPUT ACURÁCIA



FONTE: O autor (ano)

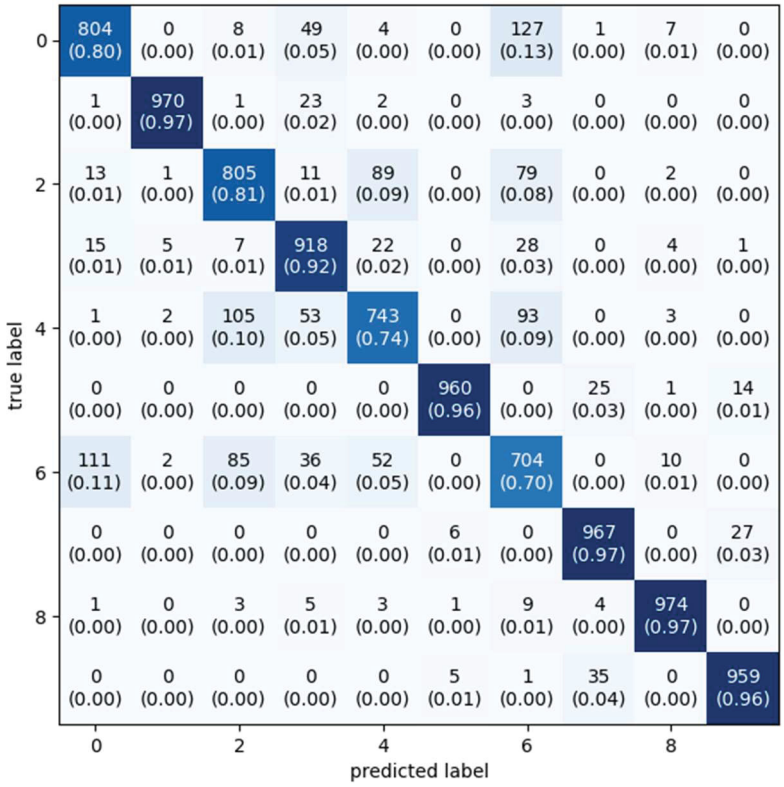
Gráfico de Acurácia: A Acurácia tende a aumentar, e chegou perto dos 90% de acurácia. Então, aumentando as Epochs para um valor que 10, poderia ser ainda melhor.

```
# Avaliação do modelo - base teste
print( model.evaluate(x_test, y_test) )
```

```
# Predições Avaliação do modelo - base teste
y_pred = model.predict(x_test).argmax(axis=1)
print(y_pred)
```

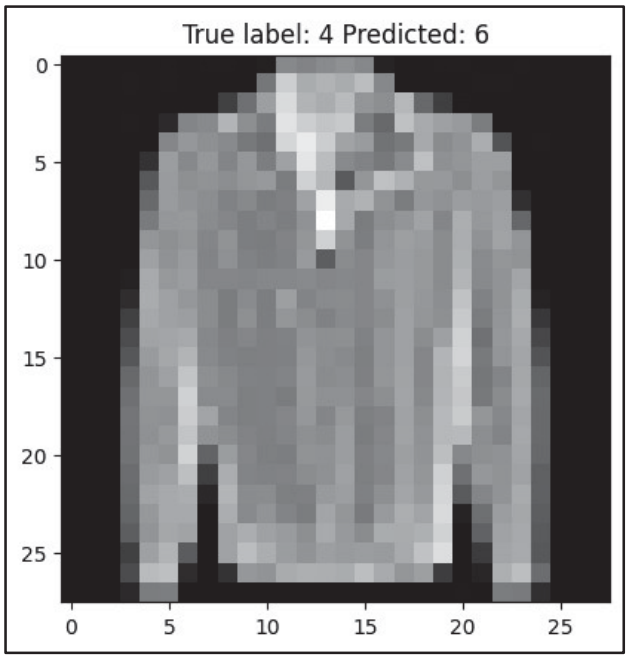
```
# Matriz  de confusão
cm = confusion_matrix(y_test, y_pred)
```

```
plot_confusion_matrix(conf_mat=cm,
                      figsize=(7, 7),
                      show_normed=True)
```




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

IMAGEM 5 – OUTPUT CLASSIFICAÇÕES ERRADAS



FONTE: O autor (ano)

Imagem gerada na seção “Mostrar algumas classificações erradas”. O valor predito é maior que o valor real. A imagem está fora do que deveria ser mostrado.

2 Regressão (RNA)

```
# Importação de dados
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-red.csv"
data = pd.read_csv(url, delimiter=';')
data.columns = ['acidez_fixa', 'acidez_volatil', 'acido_citrico',
'acucar_residual', 'cloretos', 'dioxido_de_enxofre_livre',
'dioxido_de_enxofre_total', 'densidade', 'pH', 'sulfatos', 'alcool',
'score_qualidade_vinho']
```

```
# Separação da base em treino e teste (75/25)
X = data.drop(columns=['score_qualidade_vinho']).astype(float)
Y = data['score_qualidade_vinho'].astype(float)
```

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

```
# Criando modelo Predições Avaliação do modelo - base teste
i = tf.keras.layers.Input(shape=(11,))
```

```
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)

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

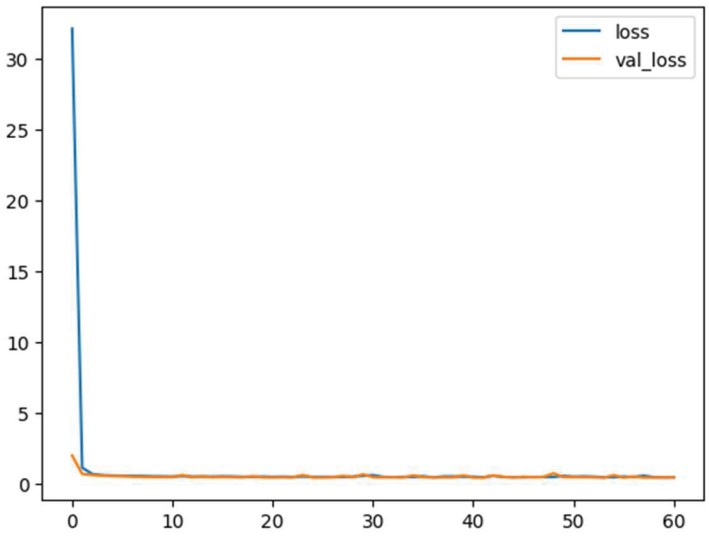
```
# Criação de funções para as métricas MAE, MSE e R2 serem inseridas no modelo
def mae(y_true, y_pred):
    absolute_error = tf.abs(y_true - y_pred)
    mae = tf.reduce_mean(absolute_error)
    return mae

def mse(y_true, y_pred):
    mse = tf.reduce_mean(tf.square(y_true - y_pred))
    return mse

def r2(y_true, y_pred):
  media = backend.mean(y_true)
  num   = backend.sum (backend.square(y_true - y_pred))
  den   = backend.sum (backend.square(y_true - media))
  return (1.0 - num/den)
```

```
# Compilação
optimizer=tf.keras.optimizers.Adam(learning_rate=0.05)
# optimizer=tf.keras.optimizers.SGD(learning_rate=0.2, momentum=0.5)
# optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
              loss="mse",
              metrics=[mae, mse, r2])
```

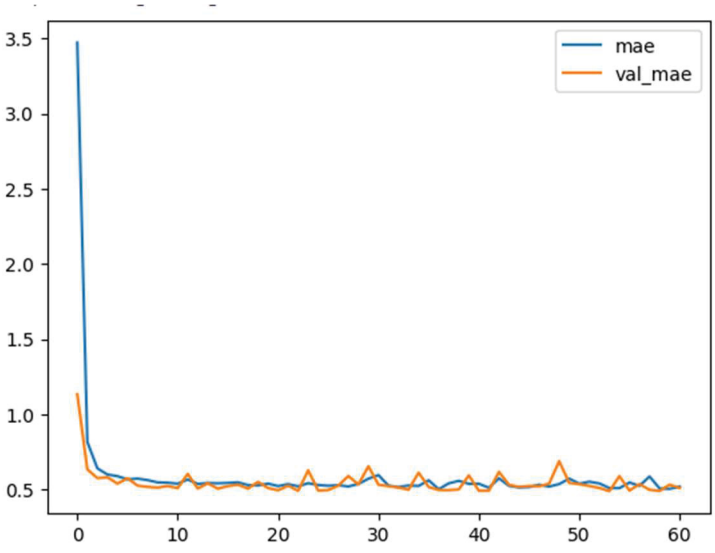
```
# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
                            monitor='val_loss',
                            patience=20,
                            restore_best_weights=True)
```

```
r = model.fit(x_train, y_train,
              epochs=1500,
              validation_data=(x_test, y_test),
              callbacks=[early_stop])
```

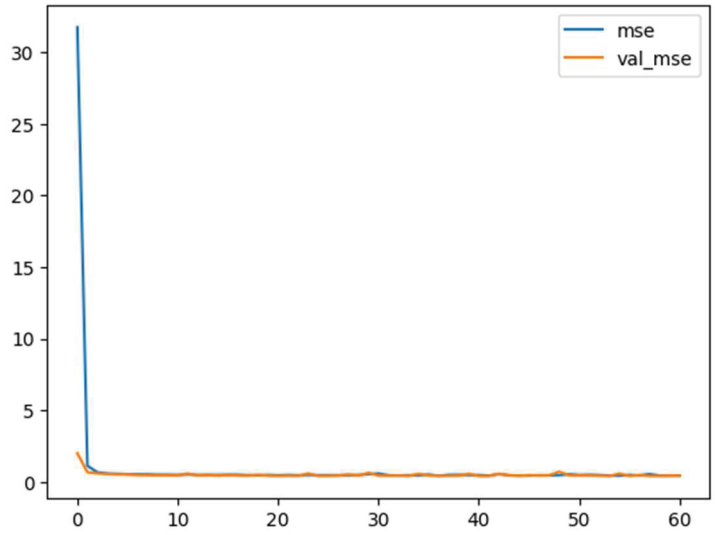
```
plt.plot( r.history["loss"], label="loss" )
plt.plot( r.history["val_loss"], label="val_loss" )
plt.legend()
```



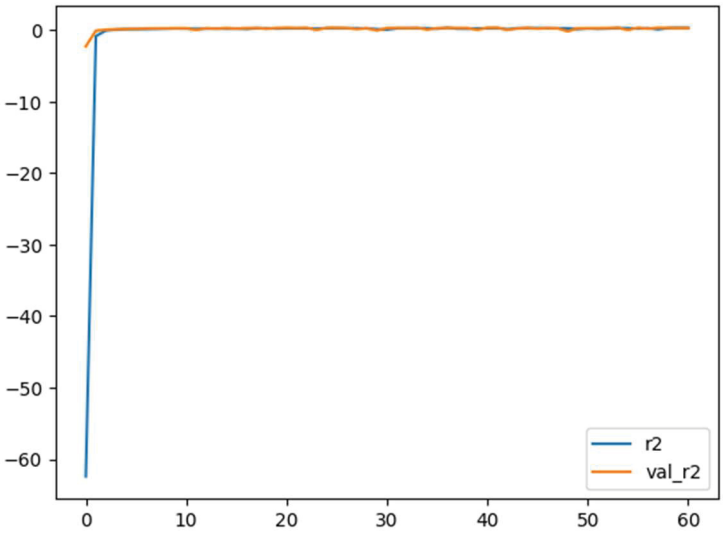

```
plt.plot( r.history["mae"], label="mae" )
plt.plot( r.history["val_mae"], label="val_mae" )
plt.legend()
```




```
plt.plot( r.history["mse"], label="mse" )
plt.plot( r.history["val_mse"], label="val_mse" )
plt.legend()
```




```
plt.plot( r.history["r2"], label="r2" )
plt.plot( r.history["val_r2"], label="val_r2" )
plt.legend()
```

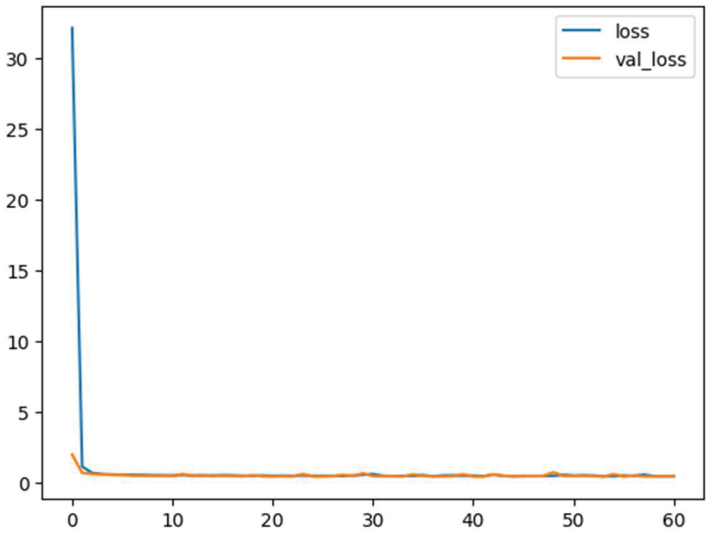



```
# Predição
y_pred = model.predict(x_test).flatten()
```

```
# Cálculo das métricas de acurácia: mae, mse e r2
mae  = mean_absolute_error(y_test, y_pred)
mse  = mean_squared_error(y_test, y_pred)
r2   = r2_score(y_test, y_pred)
```

```
# Resultados das métricas de acurácia
print("mae     = ", mae)
print("mse     = ", mse)
print("r2      = ", r2)
```

GRÁFICO 17 – OUTPUT FUNÇÃO DE PERDA REGRESSÃO



FONTE: O autor (ano)

Gráfico de Perda: O Gráfico de perda de certa forma ficou bem instável. Isso, pode ser devido ao uso
alto de valor de epochs que foi de 1500. O valor fico baixo, porém, é possível visualizar que os dados em muitos momentos aumentaram o valor. Isso, também pode ser levado em consideração para diminuir a quantidade de epochs utilizadas.

```
# Resultados das métricas de acurácia
print("mae     = ", mae)
print("mse     = ", mse)
print("r2      = ", r2)
```

FIGURA 82 – OUTPUT MÉTRICAS DE ACURÁCIA

```
# Resultados das métricas de acurácia
print("mae     = ", mae)
print("mse     = ", mse)
print("r2      = ", r2)

mae     =  0.49275920510292054
mse     =  0.4043108983506164
r2      =  0.3579245492739662
```

FONTE: O autor (ano)

A métrica que deve um maior desempenho foi a MAE com 49%, seguido da MSE com 40% e por último a R2 com o valor de 35%. O valor do resultado ficou bem baixo em todas as opções. Nesse caso, seria necessário revisar todos os valores e realizar uma outra estratégia para melhorar esse valor de acurácia.

3 Sistemas de Recomendação

```
# Importação dos dados
from google.colab import files
uploaded = files.upload()
```

```
import pandas as pd
import numpy as np

df = pd.read_csv("Base_livros(in).csv", sep=",", encoding="utf-8",
on_bad_lines='skip')

df['Notas'] = df['Notas;;;;;;'].str.replace(';', '', regex=False)
df['Notas'] = pd.to_numeric(df['Notas'], errors='coerce') # Converte para
numérico, valores inválidos viram NaN
df['Notas'] = df['Notas'].fillna(0) # Substitui NaN por 0
df = df.drop(columns=['Notas;;;;;;'])

df.head()
```

```
# Transformar dados em categoria

# Usuário
df.ID_usuario= pd.Categorical(df.ID_usuario)
df['new_ID_usuario'] = df.ID_usuario.cat.codes

# ISBN
df.ISBN = pd.Categorical(df.ISBN)
df['new_isbn'] = df.ISBN.cat.codes
```

```
# Dimensões
N = len(set(df.new_ID_usuario))
M = len(set(df.new_isbn))
```

```
# Dimensão do embedding
K = 10
```

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

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

x = Concatenate()([u_emb, m_emb])

x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)
```

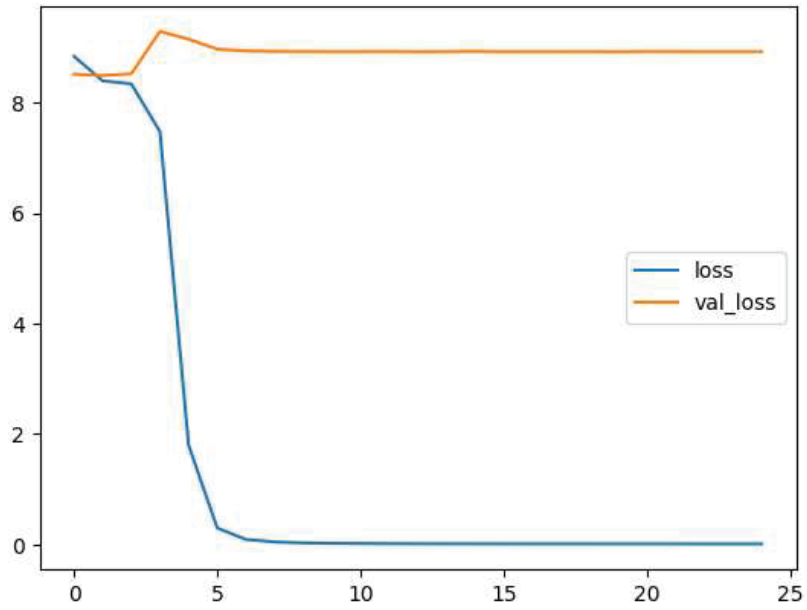
```
model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)
```

```
# Separação Pré-Processamento
ID_usuario_1, ISBN_1, Notas_1 = shuffle(df.new_ID_usuario, df.new_isbn,
df.Notas)

Ntrain = int(0.8 * len(Notas_1)) # separar os dados 80% x 20%

train_user = ID_usuario_1[:Ntrain]
train_ISBN = ISBN_1[:Ntrain]
train_Notas = Notas_1[:Ntrain]
test_user = ID_usuario_1[Ntrain:]
test_ISBN = ISBN_1[Ntrain:]
test_Notas = Notas_1[Ntrain:]
```

```
# Centralizar as notas
avg_Notas = train_Notas.mean()
train_Notas = train_Notas - avg_Notas
test_Notas = test_Notas - avg_Notas
```

```
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()
```

GRÁFICO 18 – OUTPUT FUNÇÃO DE PERDA DO SISTEMA DE RECOMENDAÇÃO



FONTE: O autor (ano)

Gráfico de perda: Os valores de validação e teste ficaram muito discrepantes um do outro. Sendo que o de teste chegou a ficando bem baixo chegando no zero e o de validação no ficou estabilizado no 8. Nesse caso, pode ser interessante verificar uma outra forma de dividir esses dados em teste e validação.

```
# Recomendação para o usuário 1
input_usuario = np.repeat(a=1, repeats=M)
Livro = np.array(list(set(ISBN_1)))

preds = model.predict( [input_usuario, Livro] )

# descentraliza as predições
rat = preds.flatten() + avg_Notas

# índice da maior nota
idx = np.argmax(rat)

print("Recomendação: Livro - ", Livro[idx], " / ", rat[idx] , "*")
```

4 Deepdream

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

```
# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
  name = url.split('/')[-1]
  image_path = tf.keras.utils.get_file(name, origin=url)
  img = PIL.Image.open(image_path)
  if max_dim:
    img.thumbnail((max_dim, max_dim))
  return np.array(img)

# Normalização da imagem
def deprocess(img):
  img = 255*(img + 1.0)/2.0
  return tf.cast(img, tf.uint8)

# Mostra a imagem
def show(img):
  display.display(PIL.Image.fromarray(np.array(img)))

# Redução do tamanho da imagem para facilitar o trabalho da RNN
original_img = download(url, max_dim=500)
show(original_img)
display.display(display.HTML('Image cc-by: <a
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-
cat_on_snow.jpg">Von.grzanka</a>'))
```

```
# Preparando o modelo
base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')
```

```
# Ativações das camadas – Maximizando
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]
```

```
# Criação do modelo
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)
```

```
# Calculo da perda
def calc_loss(img, model):
  img_batch = tf.expand_dims(img, axis=0)
  layer_activations = model(img_batch)
  if len(layer_activations) == 1:
    layer_activations = [layer_activations]
  losses = []
  for act in layer_activations:
    loss = tf.math.reduce_mean(act)
    losses.append(loss)
  return  tf.reduce_sum(losses)
```

```
# Subida do gradiente
class DeepDream(tf.Module):
  def __init__(self, model):
    self.model = model

  @tf.function(
      input_signature=(
        tf.TensorSpec(shape=[None,None,3], dtype=tf.float32),
        tf.TensorSpec(shape=[], dtype=tf.int32),
        tf.TensorSpec(shape=[], dtype=tf.float32),)
  )
  def __call__(self, img, steps, step_size):
      print("Tracing")
      loss = tf.constant(0.0)

      for n in tf.range(steps):
        with tf.GradientTape() as tape:
          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 loss, img
```

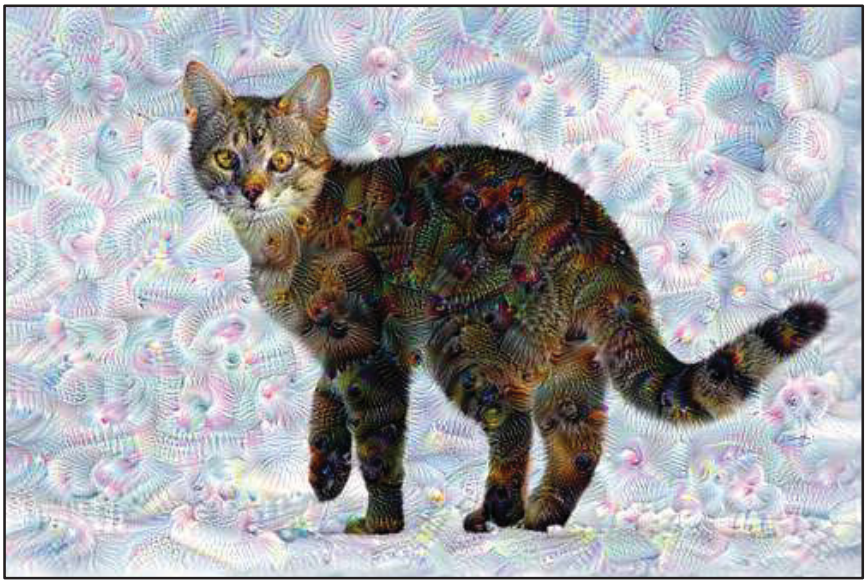
```
deepdream = DeepDream(dream_model)
```

```
# Main loop
def run_deep_dream_simple(img, steps=100, step_size=0.01):

  img = tf.keras.applications.inception_v3.preprocess_input(img)
  img = tf.convert_to_tensor(img)
  step_size = tf.convert_to_tensor(step_size)
  steps_remaining = steps
  step = 0
  while steps_remaining:
    if steps_remaining>100:
      run_steps = tf.constant(100)
    else:
      run_steps = tf.constant(steps_remaining)
    steps_remaining -= run_steps
    step += run_steps

    loss, img = deepdream(img, run_steps, tf.constant(step_size))

    display.clear_output(wait=True)
```

```
    show(deprocess(img))
    print ("Step {}, loss {}".format(step, loss))

  result = deprocess(img)
  display.clear_output(wait=True)
  show(result)

  return result
```

```
dream_img = run_deep_dream_simple(img=original_img,
                                  steps=100,
                                  step_size=0.01)
```

IMAGEM 6 – IMAGEM ONÍRICA OBTIDA POR MAIN LOOP



FONTE: O autor (ano)

A rede pegou a imagem do gato e começou a intensificar detalhes como olhos, texturas e formas, criando esse visual psicodélico com muitas repetições e distorções. É como se a rede estivesse se fosse um 'sonho' com partes exagerada.

```
# Levando a oitava
import time
start = time.time()

OCTAVE_SCALE = 1.30

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[:-1]
float_base_shape = tf.cast(base_shape, tf.float32)
```

```
for n in range(-2, 3):
  new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

  img = tf.image.resize(img, new_shape).numpy()

  img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```

IMAGEM 7 – IMAGEM ONÍRICA OBTIDA ELEVADA ATÉ UMA OITAVA

FONTE: O autor (ano)

Ao elevar o modelo a 8, aumentou o nível de amplificação dos padrões identificados pela rede neural. O resultado é uma imagem ainda mais detalhada e psicodélica, com repetições mais intensas nas formas e texturas, principalmente em relação aos olhos, esses outros padrões como se fosse 'geométricos' no fundo.

O algoritmo exagera muito os detalhes, criando um efeito ainda mais surreal no visual, como se a rede estivesse "sonhando alto" e enxergando padrões em quase tudo!A diferença entre as duas imagens geradas em a Main Loop e o modelo na oitava, foi referente a intensidade dos padrões, os detalhes e principalmente, no nível de distorção. Main Loop básico, os padrões oníricos são suaves e a imagem original é mais reconhecível. Já ao elevar o modelo até a oitava, os detalhes se intensificam, os padrões ficam exagerados e a imagem se torna muito mais complexa e fantasiosa.

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;

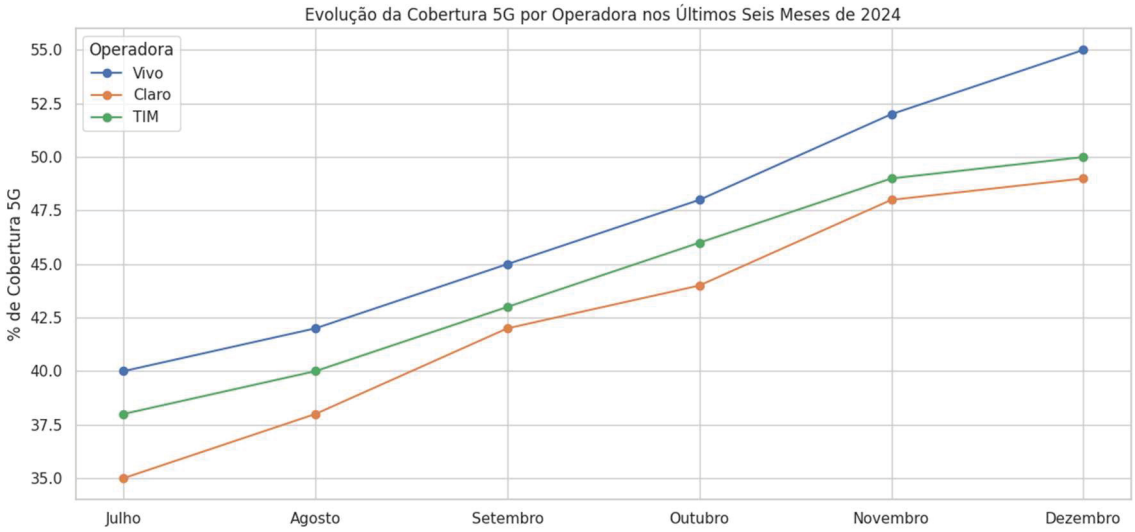
- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada**; (**50 pontos**)

B – RESOLUÇÃO

A Anatel (Agência Nacional de Telecomunicações) é uma entidade responsável por garantir o funcionamento adequado e a qualidade dos serviços de telecomunicações no Brasil, abrangendo telefonia fixa e móvel, internet, TV por assinatura, entre outros. Seus principais papéis incluem a regulação do setor, a fiscalização das operadoras, a gestão do espectro de radiofrequências, a proteção dos consumidores, a implementação de políticas públicas e o fomento à concorrência. A Anatel atua para evitar práticas abusivas de monopólio ou cartel, promovendo um ambiente competitivo e saudável no mercado. Além disso, tem a responsabilidade de incentivar investimentos no setor, visando a melhoria da infraestrutura e a qualidade dos serviços oferecidos. Atualmente, a Anatel está acompanhando de perto a renovação da infraestrutura das operadoras, com o objetivo de garantir a ampla disponibilização do 5G, atendendo à crescente demanda dos consumidores.

O Brasil está avançando de forma impressionante na implementação do 5G e as operadoras de telefonia móvel têm um papel absolutamente crucial nesse processo. Nos últimos seis meses, o número de antenas 5G aumentou de maneira significativa, com gigantes como Vivo e TIM liderando a expansão da cobertura. No entanto, apesar desse avanço em termos de infraestrutura, a adoção do 5G ainda enfrenta desafios consideráveis, com uma parte expressiva da população brasileira ainda sem acesso pleno a essa tecnologia revolucionária.

Logo abaixo o gráfico de linhas é especialmente útil para mostrar a variação de um determinado valor ao longo do tempo, facilitando a visualização de tendências, padrões sazonais e flutuações. O gráfico monstra de maneira clara a evolução da cobertura 5G em todo o território brasileiro. A Vivo com a sua ampla e consolidada cobertura, tem se destacado de forma significativa nessa corrida pela evolução tecnológica e proporcionando aos seus clientes o acesso ao 5G. No entanto, as operadoras Claro e TIM estão avançando rapidamente e conquistando cada vez mais espaço, o que torna a competição ainda mais acirrada. O crescimento consistente da cobertura ao longo dos meses é uma prova de que o Brasil está avançando na direção certa, mas é fundamental ressaltar que existe ainda um longo trajeto a ser percorrido para assegurar que todos os brasileiros e em todas as regiões do país tenham acesso pleno a essa revolução tecnológica transformadora.



Sem dúvida, este é um momento crucial para as operadoras. A expansão da cobertura 5G e o crescimento da base de clientes são fatores essenciais para garantir que o Brasil não fique para trás na competição global pela liderança no 5G. Agora, mais do que nunca, é o momento de investir fortemente em infraestrutura e proporcionar os consumidores sobre os incontáveis benefícios dessa transformação tecnológica. O futuro está sendo construído hoje e essa evolução pode trazer mais conectividade, inovação e oportunidades para todos os brasileiros.

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 Algoritmo Genético

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

```
#
def gerar_populacao(tamanho_populacao, num_cidades):
	populacao = []
	for _ in range(tamanho_populacao):
		percurso = np.arange(1, num_cidades)
		np.random.shuffle(percurso)
		percurso = np.insert(percurso, 0, 0)
		percurso = np.append(percurso, 0)
		população.append(percurso)
	return população

def selecao_torneio(populacao, cidades, tamanho_torneio=5):
	indices = np.random.choice(len(população), tamanho_torneio, replace=False)
	torneio = [populacao[i] for i in indices]
	torneio = sorted(torneio, key=lambda x: calcular_distancia(x, cidades))
	return torneio[0]
```

2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).

```
#
def calcular_distancia(percurso, cidades):
	distancia = 0
	for i in range(len(percurso) - 1):
		distancia += np.linalg.norm(cidades[percurso[i] -
cidades[percurso[i + 1]])
	return distancia
```

3) Utilize no mínimo uma população com 100 indivíduos

```
#
def gerar_populacao(tamanho_populacao, num_cidades):
	populacao = []
	for _ in range(tamanho_populacao):
		percurso = np.arange(1, num_cidades)
		np.random.shuffle(percurso)
		percurso = np.insert(percurso, 0, 0)
		percurso = np.append(percurso, 0)
		populacao.append(percurso)
	return populacao

def selecao_torneio(populacao, cidades, tamanho_torneio=5):
	indices = np.random.choice(len(populacao), tamanho_torneio, replace=False)
	torneio = [populacao[i] for i in indices]
	torneio = sorted(torneio, key=lambda x: calcular_distancia(x, cidades))
	return torneio[0]
```

4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação

```
#
def mutacao_swap(individuo):
	a, b = sorted(np.random.choice(range(1, len(individuo)-1), 2,
replace=False))
	individuo[a], individuo[b] = individuo[b], individuo[a]
	return individuo
```

5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover ox)

```
#
def crossover_ox(pai1, pai2):
	size = len(pai1)
```

```
	filho = [None]*size
	inicio, fim = sorted(np.random.choice(range(1, size-1), 2, replace=False))
	filho[inicio:fim] = pai1[inicio:fim]
	pointer = fim
	for gene in pai2:
		if gene not in filho:
			if pointer >= size - 1:
				pointer = 1
			filho[pointer] = gene
			pointer += 1

	filho[0] = filho[-1] = 0
	return np.array(filho)
```

6) Preserve sempre a melhor solução de uma geração para outra

```
#
def algoritmo_genetico(cidades, tamanho_populacao=100, geracoes=1000,
tx_mutacao=0.01, tx_crossover=0.9):
	populacao = gerar_populacao(tamanho_populacao, len(cidades))
	melhor_solucao = min(populacao, key=lambda x: calcular_distancia(x,
cidades))
	melhor_distancia = calcular_distancia(melhor_solucao, cidades)
	primeira_melhor_solucao = melhor_solucao.copy()
	primeira_melhor_distancia = melhor_distancia

	for _ in range(geracoes):
		nova_populacao = [melhor_solucao]

		while len(nova_populacao) < tamanho_populacao:
			pai1 = selecao_torneio(populacao, cidades)
			pai2 = selecao_torneio(populacao, cidades)

			if np.random.rand() < tx_crossover:
				filho = crossover_ox(pai1, pai2)
			else:
				filho = pai1.copy()

			if np.random.rand() < tx_mutacao:
				filho = mutacao_swap(filho)

			nova_populacao.append(filho)

		populacao = nova_populacao
		atual_melhor = min(populacao, key=lambda x: calcular_distancia(x,
cidades))
		atual_distancia = calcular_distancia(atual_melhor, cidades)

		if atual_distancia < melhor_distancia:
```

```
                    melhor_solucao = atual_melhor
                    melhor_distancia = atual_distancia

      return primeira_melhor_solucao, primeira_melhor_distancia, melhor solução,
melhor_distancia, cidades
```

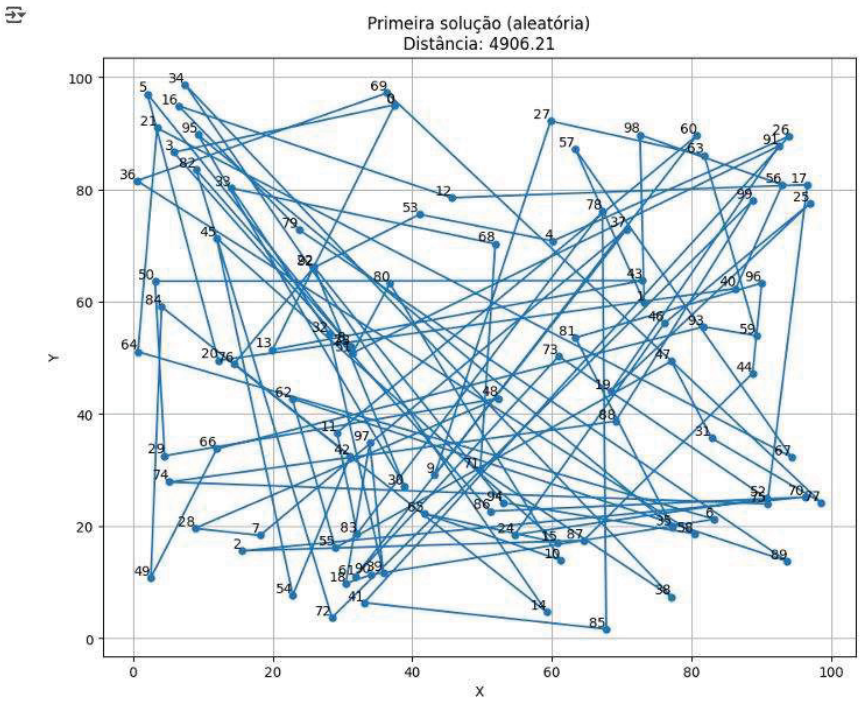
Código

```
import numpy as np
import matplotlib.pyplot as plt

# Configuração inicial
num_cidades = 100
np.random.seed(42)
cidades = np.random.rand(num_cidades, 2) * 100

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

def gerar_populacao(tamanho_populacao, num_cidades):
      populacao = []
      for _ in range(tamanho_populacao):
            percurso = np.arange(1, num_cidades)
            np.random.shuffle(percurso)
            percurso = np.insert(percurso, 0, 0)
            percurso = np.append(percurso, 0)
            população.append(percurso)
      return populacao

def selecao_torneio(populacao, cidades, tamanho_torneio=5):
      indices = np.random.choice(len(população), tamanho_torneio, replace=False)
      torneio = [populacao[i] for i in indices]
      torneio = sorted(torneio, key=lambda x: calcular_distancia(x, cidades))
      return torneio[0]

def crossover_ox(pai1, pai2):
      size = len(pai1)
      filho = [None]*size
      inicio, fim = sorted(np.random.choice(range(1, size-1), 2, replace=False))
      filho[inicio:fim] = pai1[inicio:fim]
      pointer = fim
      for gene in pai2:
            if gene not in filho:
                  if pointer >= size - 1:
                        pointer = 1
```

```
                    filho[pointer] = gene
                    pointer += 1
        filho[0] = filho[-1] = 0
        return np.array(filho)

def mutacao_swap(individuo):
        a, b = sorted(np.random.choice(range(1, len(individuo)-1), 2,
replace=False))
        individuo[a], individuo[b] = individuo[b], individuo[a]
        return individuo

def algoritmo_genetico(cidades, tamanho_populacao=100, geracoes=1000,
tx_mutacao=0.01, tx_crossover=0.9):
        população = gerar_populacao(tamanho_populacao, len(cidades))
        melhor_solucao = min(população, key=lambda x: calcular_distancia(x,
cidades))
        melhor_distancia = calcular_distancia(melhor_solucao, cidades)
        primeira_melhor_solucao = melhor_solucao.copy()
        primeira_melhor_distancia = melhor_distancia


        for _ in range(geracoes):
                nova_publicacao = [melhor_solucao]

                while len(nova_populacao) < tamanho_populacao:
                        pai1 = selecao_torneio(populacao, cidades)
                        pai2 = selecao_torneio(populacao, cidades)

                        if np.random.rand() < tx_crossover:
                                filho = crossover_ox(pai1, pai2)
                        else:
                                filho = pai1.copy()

                        if np.random.rand() < tx_mutacao:
                                filho = mutacao_swap(filho)

                        nova_populacao.append(filho)

                populacao = nova_populacao
                atual_melhor - min(populacao, key=lambda x: calcular_distancia(x,
cidades))
                atual_distancia = calcular_distancia(atual_melhor, cidades)

                if atual_distancia < melhor_distancia:
                        melhor_solucao = atual_melhor
                        melhor_distancia = atual_distancia

        return primeira_melhor_solucao, primeira_melhor_distancia, melhor_solucao,
melhor_distancia, cidades

def plotar_solucao(solucao, cidades, titulo):
```

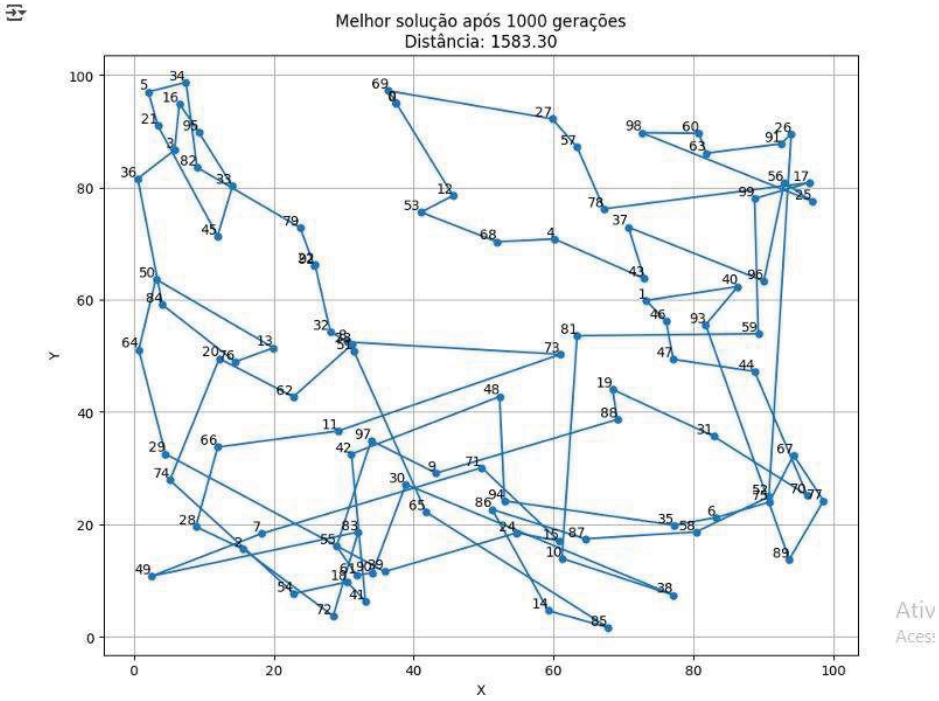
```
        plt.figure(figsize=(10,8))
        plt.plot(cidades[solucao, 0], cidades[solucao, 1], '-o', markersize=5,
linewidth=1.5)

        # Adiciona os números das cidades nos pontos
        for i, cidade in enumerate(solucao):
                plt.text(cidades[cidade, 0], cidades[solucao, 1], str(cidade),
fontsize=10, ha='right', va='bottom')

        plt.title(titulo)
        plt.xlabel('X')
        plt.ylabel('Y')
        plt.grid(True)
        plt.show()

# Executar o algoritmo
primeira_solucao, primeira_distancia, melhor_solucao, melhor_distancia, cidades
= algoritmo_genetico(
        cidades, tamanho_populacao=100, geracoes=1000)

# Plotar soluções - Primeira Solução
plotar_solucao(primeira_solucao, cidades, f'Primeira solução
(aleatória)\nDistância: {primeira_distancia:.2f}')
```

GRÁFICO 19 – PRIMEIRA SOLUÇÃO (ALEATÓRIA)



FONTE: O autor (2025)

GRÁFICO 20 – MELHOR SOLUÇÃO APÓS 1000 GERAÇÕES



FONTE: O autor (2025)

Os gráficos gerados mostram a evolução do percurso ao longo das gerações.

A primeira solução aleatória é comparada com a melhor solução encontrada, evidenciando a eficiência do algoritmo genético na redução da distância total percorrida. O algoritmo genético apresentou bons resultados na solução do Problema do Caixeiro Viajante, otimizando o percurso com o uso de operadores de cruzamento e mutação. A preservação da melhor solução garantiu a manutenção de resultados de alta qualidade ao longo da evolução.

2 Compare a representação de dois modelos vetoriais

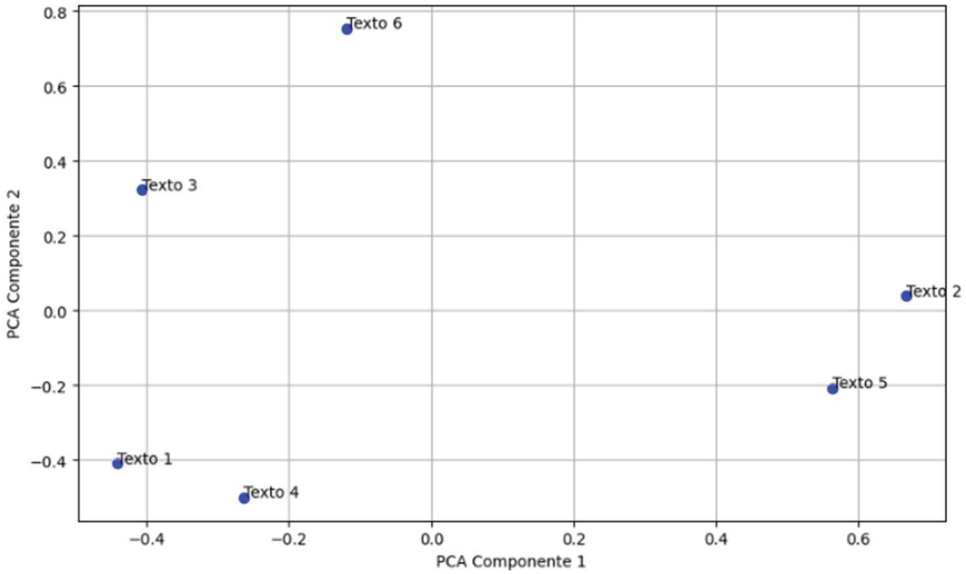
TEXTO:
O cachorro corre no parque.
O gato está dormindo na cama.
O cachorro é muito amigável.
A pessoa caminha pelo parque com o cachorro.
O gato gosta de brincar com a bola.
Os animais são muito fofos

```
!pip install gensim
!pip install numpy==1.23.5
!pip install --upgrade gensim

import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer
import gensim.downloader as api

textos = [
  "O cachorro corre no parque.",
  "O gato está dormindo na cama.",
  "O cachorro é muito amigável.",
  "A pessoa caminha pelo parque com o cachorro.",
  "O gato gosta de brincar com a bola.",
  "Os animais são muito fofos."

]

tfidf_vectorizer = tfidf_vectorizer()
tfidf_vectors = tfidf_vectorizer.fit_transform(textos).toarray()

# Carregar modelo pré-treinado
model = api.load("glove-wiki-gigaword-50")

# Função para gerar embedding médio de sentenças
def sentence_embedding(sentence, model):
  words = sentence.lower().split()
  embeddings = [model[word] for word in words if word in model]

  if len(embeddings) == 0:
    return np.zeros(model.vector_size)
  return np.mean(embeddings, axis=0)

# Gerar embeddings das sentenças
word2vec_vectors = np.array([sentence_embedding(sentence, model) for sentence in
textos])

# PCA para CBOW
pca_tfidf = PCA(n_components=2)
tfidf_reduced = pca_tfidf.fit_transform(tfidf_vectors)

# PCA para Word2Vec
pca_word2vec = PCA(n_components=2)
word2vec_reduced = pca_word2vec.fit_transform(word2vec_vectors)

def plot_vectors(vectors, labels, title):
  plt.figure(figsize=(10,6))
  plt.scatter(vectors[:, 0], vectors[:, 1], color='blue')

  for i, label, in enumerate(labels):
    plt.annotate(label, (vectors[i, 0], vectors[i, 1]))

  plt.title(tittle)
  plt.xlabel('PCA Componente 1')
```

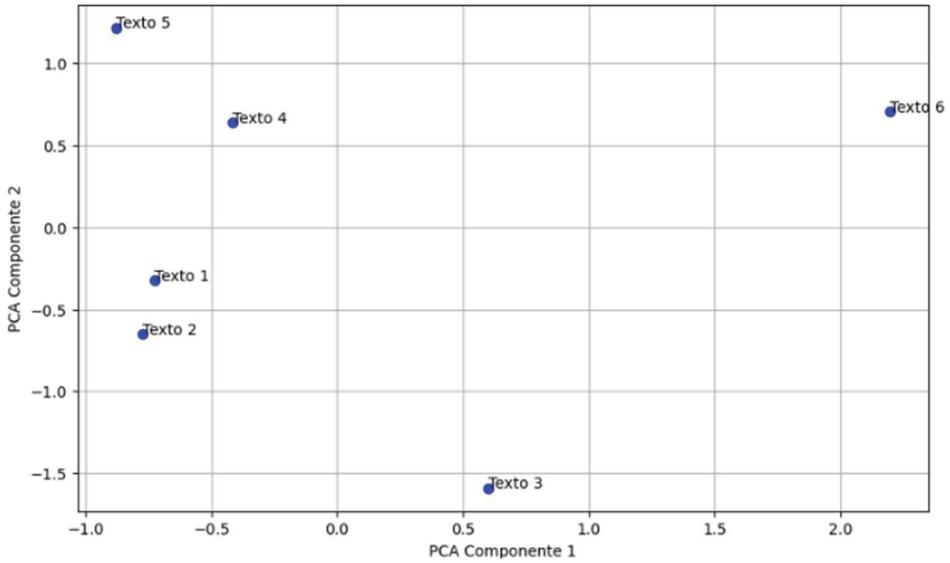
```
plt.ylabel('PCA Componente 2')
plt.grid(True)
plt.show()
```

GRÁFICO 21 – REPRESENTAÇÃO VETORIAL - CBOW COM PCA



FONTE: O autor (2025)

GRÁFICO 22 – REPRESENTAÇÃO VETORIAL - WORD2VEC COM PCA



FONTE: O autor (2025)