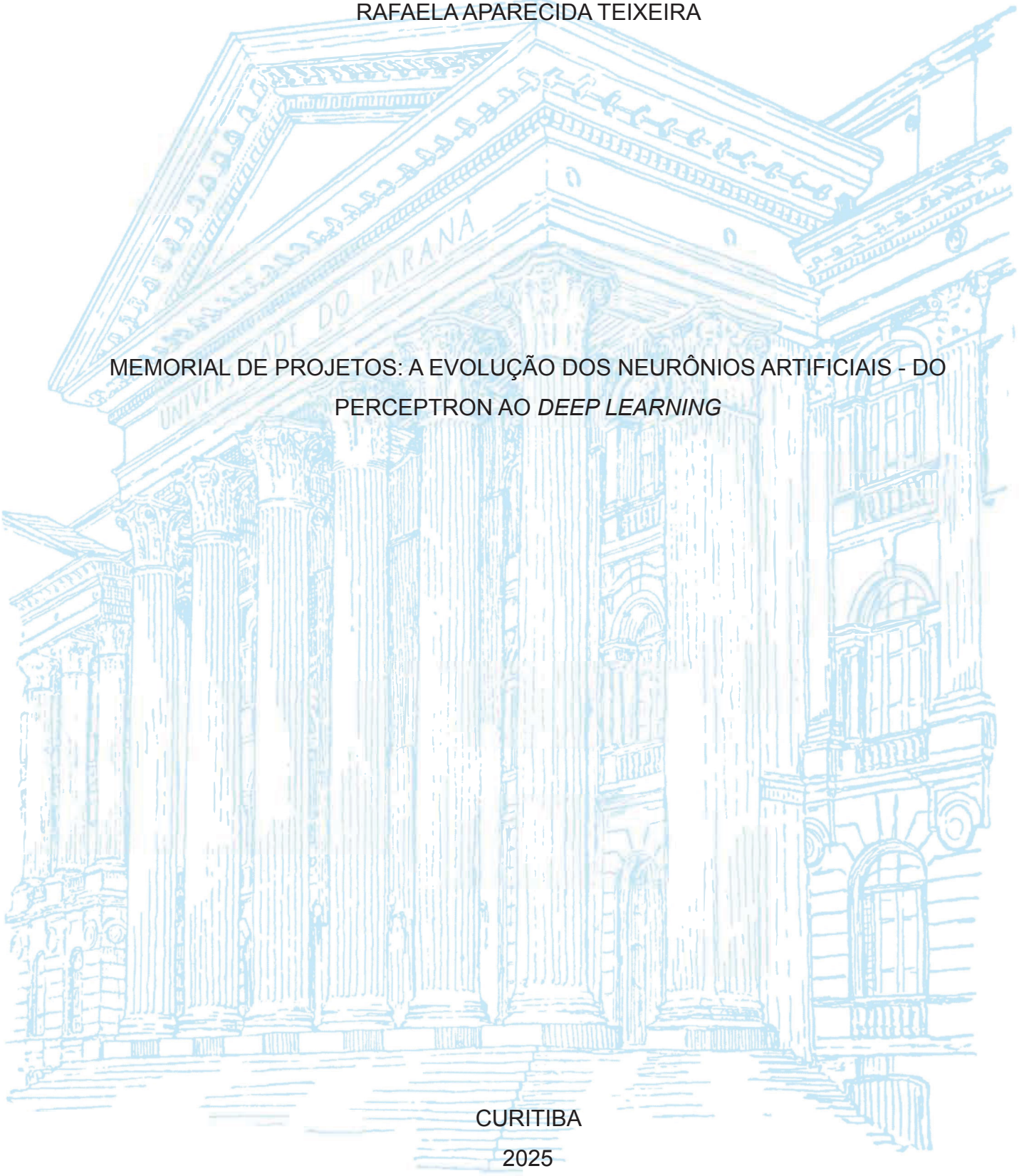
UNIVERSIDADE FEDERAL DO PARANÁ

RAFAELA APARECIDA TEIXEIRA



# MEMORIAL DE PROJETOS: A EVOLUÇÃO DOS NEURÔNIOS ARTIFICIAIS - DO PERCEPTRON AO *DEEP LEARNING*

CURITIBA

2025

RAFAELA APARECIDA TEIXEIRA

# MEMORIAL DE PROJETOS: A EVOLUÇÃO DOS NEURÔNIOS ARTIFICIAIS - DO PERCEPTRON AO *DEEP LEARNING*

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 **RAFAELA APARECIDA TEIXEIRA**, intitulada: **MEMORIAL DE PROJETOS: A EVOLUÇÃO DOS NEURÔNIOS ARTIFICIAIS - DO PERCEPTRON AO DEEP LEARNING**, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua _____ 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, 27 de Outubro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

## RESUMO

Este parecer técnico busca compreender a evolução dos neurônios artificiais, uma área fundamental dentro da Inteligência Artificial (IA), desde o desenvolvimento do perceptron até as modernas redes neurais profundas (*deep learning*). O objetivo é analisar as transformações nos modelos de redes neurais, evidenciando como as mudanças nos algoritmos e nas arquiteturas permitiram a superação de limitações de capacidade de processamento e de adaptação. A análise destaca o papel do perceptron como o primeiro modelo, suas limitações e o impacto das inovações subsequentes, como o algoritmo de retropropagação, que permitiu o treinamento de redes neurais mais complexas. Além disso, o parecer explora o impacto das redes neurais profundas, que têm impulsionado avanços significativos em áreas como reconhecimento de voz, processamento de imagem e tradução automática. A conclusão destaca a importância do aprendizado profundo e os desafios atuais, como a necessidade de grandes volumes de dados e o alto custo computacional, apontando direções futuras para a área.



**Palavras-chave:** redes neurais; Perceptron; retropropagação; aprendizado profundo; inteligência artificial.

## ABSTRACT

This technical report seeks to understand the evolution of artificial neurons, a fundamental area within Artificial Intelligence (AI), from the development of the perceptron to modern deep learning networks. The goal is to analyze the transformations in neural network models, highlighting how changes in algorithms and architectures allowed for the overcoming of processing and adaptation limitations. The analysis emphasizes the role of the perceptron as the first model, its limitations, and the impact of subsequent innovations such as the backpropagation algorithm, which enabled the training of more complex neural networks. Additionally, the report explores the impact of deep neural networks, which have driven significant advancements in areas like speech recognition, image processing, and machine translation. The conclusion highlights the importance of deep learning and current challenges such as the need for large datasets and high computational costs, pointing to future directions for the field.

**Keywords:** Neural Networks; Perceptron; Backpropagation; Deep Learning; Artificial Intelligence.

**SUMÁRIO**

# 1.PARECER TÉCNICO

## 1.1.INTRODUÇÃO

A evolução dos neurônios artificiais é uma das mais fascinantes jornadas dentro do campo da inteligência artificial (IA), refletindo não apenas avanços teóricos, mas também desenvolvimentos tecnológicos que têm moldado o cenário contemporâneo. Desde a criação do perceptron nos anos 1950 até os complexos sistemas de *deep learning* que caracterizam as redes neurais modernas, houve uma evolução significativa em termos de capacidades computacionais, algoritmos e aplicações práticas. Este parecer técnico busca explorar essa trajetória, destacando marcos importantes e o impacto dessa evolução no avanço da IA, com ênfase nas redes neurais profundas.

## 1.2.O PERCEPTRON: O INÍCIO DA JORNADA

O perceptron, desenvolvido por Frank Rosenblatt em 1958, foi uma das primeiras implementações práticas de um modelo de neurônio artificial, embora o conceito teórico original tenha sido proposto por McCulloch e Pitts (1943). Sua simplicidade, baseada em uma unidade de processamento que simula a maneira como um neurônio biológico funciona, estabeleceu as bases para o que viria a ser o campo das redes neurais. O perceptron de Rosenblatt é um classificador linear, projetado para resolver problemas de separação de classes em espaços de alta dimensão. No entanto, apesar de suas limitações, como a incapacidade de resolver problemas não linearmente separáveis (exemplo clássico: o problema XOR), ele foi fundamental para impulsionar a pesquisa subsequente em redes neurais (Rosenblatt, 1958).

## 1.3.A LIMITAÇÃO DO PERCEPTRON E A CRISE DAS REDES NEURAIS

Após o entusiasmo inicial, o campo das redes neurais entrou em um período de estagnação, em grande parte devido à limitação do perceptron e à descoberta de

que ele não poderia resolver certos tipos de problemas. A principal crítica surgiu com o trabalho de Minsky e Papert (1969), que demonstraram que redes neurais simples não poderiam aprender problemas não lineares. Isso levou a uma diminuição do interesse por redes neurais nas décadas de 1970 e 1980, período que ficou conhecido como a *inverno da IA*. Contudo, avanços significativos estavam prestes a surgir, com novas arquiteturas e algoritmos que superariam essas limitações.

## 1.4.O RENASCIMENTO DAS REDES NEURAIS: RETROPROPAGAÇÃO

O renascimento das redes neurais ocorreu nos anos 1980, com o desenvolvimento do algoritmo de retropropagação (*backpropagation*), amplamente popularizado por Geoffrey Hinton e outros (Rumelhart; Hinton; Williams, 1986). Este algoritmo permitiu o treinamento eficiente de redes neurais multicamadas (também conhecidas como redes neurais profundas ou *deep networks*), superando as limitações do perceptron simples e possibilitando a modelagem de problemas mais complexos. A retropropagação resolve problemas de otimização de redes neurais ajustando os pesos das conexões de acordo com os erros cometidos na previsão, tornando as redes neurais profundas mais eficientes e capazes de resolver problemas de classificação e previsão de alto nível.

## 1.5.O *DEEP LEARNING*: O AVANÇO CONTEMPORÂNEO

A verdadeira revolução nas redes neurais ocorreu no final dos anos 2000 e início dos anos 2010 com o avanço das redes neurais profundas (*deep learning*). A combinação de grandes volumes de dados (*big data*), poder computacional elevado e avanços em algoritmos de otimização, como o uso de GPUs para treinamento, permitiu a criação de redes muito mais complexas e com camadas adicionais de processamento (Lecun; Bengio; Hinton, 2015). Redes como as *Convolutional Neural Networks* (CNNs) e as *Recurrent Neural Networks* (RNNs) passaram a ser amplamente utilizadas em tarefas como reconhecimento de imagem, processamento

de linguagem natural e tradução automática, estabelecendo novos padrões de desempenho em várias áreas.

O aumento da complexidade das redes neurais profundas foi possível também graças a técnicas de regularização, como o *dropout* (Hinton *et al*., 2012), que evitam o sobreajuste, e otimizações de algoritmos de treinamento, como o Adam (Kingma; Ba, 2015), que garantem convergência mais rápida e eficiente.

## 1.6.IMPACTO E APLICAÇÕES DAS REDES NEURAIS PROFUNDAS

As redes neurais profundas têm gerado transformações em diversas áreas, como reconhecimento de voz, diagnóstico médico, automação industrial, e, especialmente, na indústria automobilística com veículos autônomos. A capacidade dessas redes de aprender padrões complexos a partir de grandes volumes de dados tem permitido avanços significativos em IA, tornando-a mais acessível e aplicável ao mundo real (Goodfellow; Bengio; Courville 2016).

## 1.7.DESAFIOS E FRONTEIRAS TECNOLÓGICAS

Apesar do impressionante progresso alcançado, as redes neurais profundas ainda enfrentam desafios consideráveis. Entre eles, destaca-se a necessidade de grandes quantidades de dados rotulados para treinamento, a explicabilidade limitada dos modelos e o alto custo computacional de treinamento de redes profundas. Pesquisas atuais estão focadas em soluções para esses problemas, como o desenvolvimento de métodos de aprendizado não supervisionado e auto-supervisionado (Henriquez, 2020), além de inovações no campo do aprendizado por transferência e redes neurais mais eficientes em termos computacionais.

## 1.8.CONCLUSÃO

A evolução dos neurônios artificiais, desde o perceptron até as redes neurais profundas, representa um dos avanços mais significativos da inteligência artificial. O perceptron forneceu a base teórica, mas as limitações do modelo original geraram um longo período de estagnação até a introdução de novos algoritmos e técnicas,

como a retropropagação, que permitiram a criação de redes neurais mais profundas e poderosas. O advento do *deep learning* revolucionou diversas áreas da ciência e tecnologia, e as redes neurais continuam a se expandir, oferecendo soluções inovadoras para desafios cada vez mais complexos. No entanto, ainda existem obstáculos a serem superados, especialmente relacionados ao custo computacional e à interpretabilidade dos modelos, mas as pesquisas atuais apontam para um futuro promissor, com aplicações cada vez mais práticas e amplas.

# REFERÊNCIAS

CHOUDHARY, Alok. **Evolução do hardware e seu impacto nas redes neurais profundas.** Journal of Artificial Intelligence Research, v. 31, p. 12-25, 2019.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge, MA: MIT Press, 2016. ISBN 9780262035613.

HENRIQUEZ, F. **Self-supervised learning for neural networks**. Journal of Machine Learning Research, v. 21, p. 1-35, 2020.

HINTON, Geoffrey E.; NAYLOR, Stephen; THARP, James. **Dropout: A simple way to prevent neural networks from overfitting**. Journal of Machine Learning Research, v. 15, p. 1929-1958, 2012.

KINGMA, D. P.; BA, J. Adam: **A method for stochastic optimization**. Proceedings of the 3rd International Conference on Learning Representations (ICLR), p. 1-15, San Diego, 2015.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. **Deep learning**. Nature, v. 521, p. 436-444, 2015.

McCULLOCH, W. S.; PITTS, W. **A logical calculus of the ideas immanent in nervous activity**. The Bulletin of Mathematical Biophysics, 5(4), 115-133, 1943.

MINSKY, Marvin; PAPERT, Seymour. **Perceptrons: An Introduction to Computational Geometry**. Cambridge, MA: MIT Press, 1969.

ROSENBLATT, Frank. **The Perceptron: A Perceiving and Recognizing Automaton**. Relatório técnico 85-460-1, Cornell Aeronautical Laboratory, Buffalo, 1957.

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. **Learning representations by backpropagating errors**. Nature, v. 323, p. 533-536, 1986.

# 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.

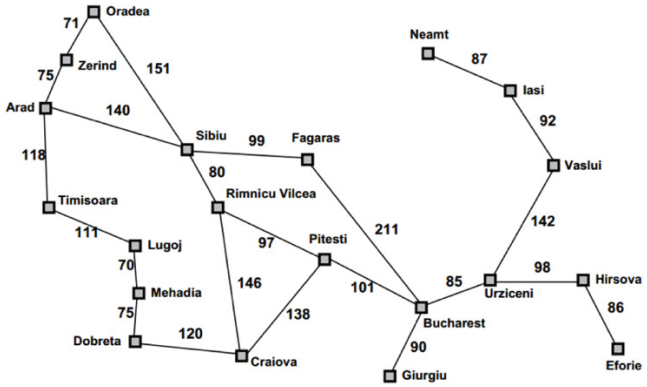
### 2 Busca Heurística

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

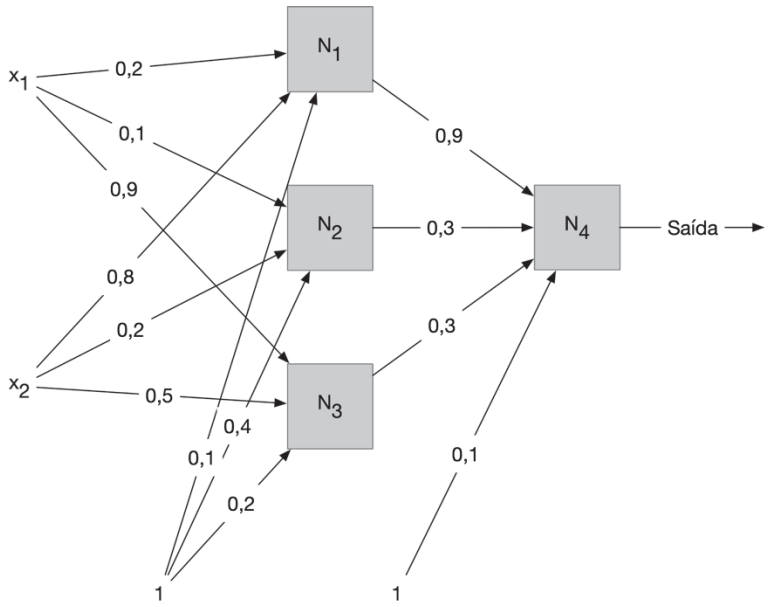
**Conjunção:** $\alpha,\ \beta \vdash \alpha \wedge \beta$

**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) Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e algoritmos capazes de realizar tarefas que normalmente exigiriam inteligência humana. Isso inclui habilidades como aprendizado, raciocínio, resolução de problemas, compreensão de linguagem natural, reconhecimento de padrões, tomada de decisões e muito mais. Os sistemas de IA são projetados para analisar grandes quantidades de dados, identificar padrões e tendências, aprender com exemplos passados e tomar decisões ou fornecer insights baseados nesses dados. Existem várias abordagens e técnicas dentro da IA incluindo aprendizado de máquina, redes neurais, lógica fuzzy, algoritmos genéticos, processamento de linguagem natural, visão computacional, entre outras. A IA tem uma ampla gama de aplicações em diversos setores, incluindo medicina, finanças, manufatura, automotivo, jogos, assistentes virtuais, entretenimento e muito mais. Seu objetivo final é criar sistemas que possam imitar a inteligência humana de maneira eficaz em várias tarefas e contextos.

   b)
      i) - A citação da frase “desenvolvimento de sistemas e algoritmos capazes de realizar tarefas que normalmente exigiriam inteligência humana”, por si só já assinala o uso da abordagem de “Pensar como os humanos” indicando a busca por desenvolver algoritmos que se aproximem deste estado.
      ii) – A resposta cita também que a IA é projetada para analisar grandes quantidades de dados o que indica a abordagem “Pensar racionalmente”.
      iii) – A observação que a IA possui uma ampla gama de aplicações em diversos setores nos indica a abordagem “Agir como humanos”
      iv) – O texto fala sobre que as IA são projetadas para aprender por exemplos e tomar decisões baseadas nesses dados, essa abordagem está relacionada a “Agir racionalmente”

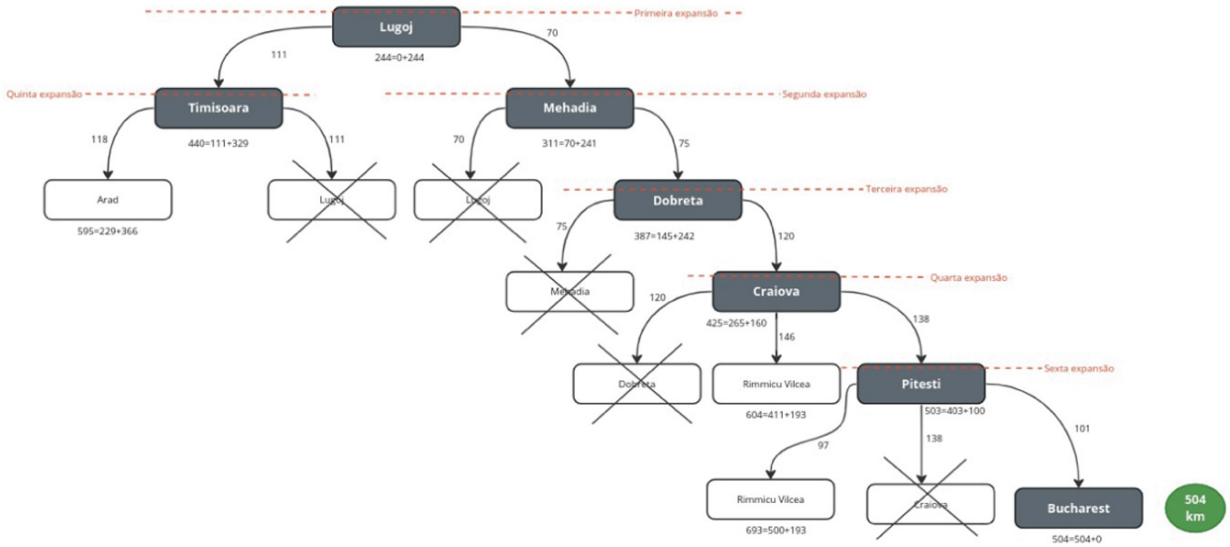
   c) Os conteúdos relacionados as repostas citam que o ChatGPT pode ser definido como um modelo de linguagem que tem como base IA e utiliza redes neurais, machine learning e avançados algoritmos de processamento de linguagem natural. Com foco em diálogos virtuais, sendo a evolução dos antigos chatbots, tendo sido treinado com grandes volumes de dados ou qual consegue manter uma conversa humana em linguagem natural, respondendo perguntas feitas pelos usuários, muito ú l para facilitar o acesso a informações e códigos a par r de seus dados.

      Porém criticado por possivelmente u lizar informações autorais ao qual indicaria plagio e por não indicar as fontes u lizadas para montar o racional da resposta, deixando assim lacunas e questionamento sobre as origens dos dados.
      Outro ponto é relação aos dados por nós imputados nas perguntas ao ChatGPT, elas são usadas para armazenamento e utilização posterior? São processadas e descartadas? As fontes nos indicam que temos que evitar de compartilhar informações confidenciais.
      O ChatGPT possui limitações: Bias: pois devido ao treinamento com grandes volumes de dados desatualizados ou enviesados, Contexto limitado: ele pode ter ainda dificuldades em responder nuances complexas ou informações implícitas.
      Referências:
      ChatGPT: o que é, como funciona e exemplos de como usar (investnews.com.br)
      ChatGPT é seguro? Entenda como funciona (santodigital.com.br)
      ChatGPT: o que é, como funciona e limitações (V.6. N.6. P.2, 2023) – Blog UFABC Divulga Ciência

   d) Eu classificaria o ChatGPT como um avanço tecnológico, devendo ser considerado tal qual a revolução industrial, uma fonte que não irá fazer o trabalho para o ser humano, mas facilitará e auxiliará dando agilidade em trabalhos “mecânicos” por assim dizer, permitindo que o

tempo seja aproveitado para desenvolvimento de novas ideias e criações. Baseado nas 4 abordagens estudadas podemos indicar que o ChatGPT as simula, tentando ao buscar as respostas ele agem como “Pensar como os humanos”, e dentro das grandes quantidades de material ele usa essa abordagem para separar os conteúdos como se pensasse racionalmente, claro que devido as suas limitações ainda não tem como ele agir como humanos, mas devido ao seu conhecimento os humanos podem aplicar seu conhecimento, o agir racionalmente pode ser atingido com maior facilidade uma vez que sua análise é baseada em dados.

2) Resposta:




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

| | |
|---|---|
| R4: r → p | Equivalência Implicação R3 |
| R5: q → p | Silogismo Hipotético R2, R4 |
| R6: (q → p) ^ (p → q) | Conjunção R5, R1 |
| R7: q ↔ p | Equivalência Bicondicional R6 |

3) Resposta:

a) (6,25 pontos) Valor de saída do neurônio N1
***Resposta***: N1 = 0,3

b) (6,25 pontos) Valor de saída do neurônio N2
***Resposta***: N2 = 0,3

c) (6,25 pontos) Valor de saída do neurônio N3
***Resposta***: N3 = -2

d) (6,25 pontos) Valor de saída da rede como um todo
***Resposta***: Saída = -0,1391

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

## 2 Visualização dos dados

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

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

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

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

a. Escolha as variáveis **numéricas** (modelos de Regressão) para serem as variáveis independentes do modelo.A variável target é **avg_price. Observação:** caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no modelo. Indique **quais variáveis** foram transformadas e **como** foram transformadas
b. Crie partições contendo 75% dos dados para treino e 25% para teste
c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros. **Observação**: caso julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo
d. Grave os valores preditos em variáveis criadas
e. Realize a análise de importância das variáveis para estimar a variável target, **para cada modelo treinado**
f. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis
g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e $R^2$
h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

# B - RESOLUÇÃO

## 1 - Análise Exploratória dos dados

**F).** Há valores faltantes em todas as colunas com a quantidade de 65.245 registros por coluna, removido as linhas onde era vazia para todas as colunas. Há 2 valores duplicados. Criado categorias para separar colunas numéricas e categóricas.

## 2 - Visualização dos dados

**E).** O modelo de engrenagem automática é mais caro que o modelo de engrenagem manual. A marca Renault tem o menor preço médio de carros com engrenagem automática. A marca Fiat tem o menor preço médio de carros com engrenagem manual.

**G).** Diesel é o tipo de combustível mais caro. Alcool é o tipo de combustível mais barato. A marca Ford tem o menor preço médio de carros com combustível diesel. A marca Renault tem o menor preço médio de carros com combustível gasolina. A marca Ford tem o menor preço médio de carros com combustível álcool.

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

**F).** A variável engine_size é a mais importante para todos os modelos.

**G).** O modelo Xgboost.

**H).** O modelo Xgboost obteve o melhor resultado com as métricas de avaliação MSE(Ccalcula o erro quadrático médio das predições do nosso modelo. Quanto maior o MSE, pior é o modelo.) e $R^2$(Métrica que varia entre 0 e 1 e é uma razão que indica o quão bom o nosso modelo. Quanto maior seu valor, melhor é o modelo).

## Linhas de código

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

import warnings
warnings.filterwarnings('ignore')

# Bibliotecas de machine learning
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
```

```
from sklearn.preprocessing import LabelEncoder

# Métricas de avaliação dos modelos
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Função read_csv para importar os dados da pasta do computador
dados = pd.read_csv('precos_carros_brasil.csv')

dados.columns

dados.head()

# Observando número de linhas e colunas
dados.shape

# Imprime o tipo de dado em cada coluna: object - variáveis categóricas; float64 e
int64 - variáveis numéricas
dados.dtypes

# Verificando se existem valores faltantes nos dados
dados.isna().any()

# Verificando a quantidade de valores faltantes por coluna
dados.isna().sum()

# Verificar a quantidade de linhas vazias em todas as colunas
dados.isnull().all(axis=1).sum()

# Remover linhas vazias de todas as colunas
dados = dados.dropna(how='all')

# Verificando a quantidade de valores faltantes por coluna
dados.isna().sum()

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

#Valores floats para inteiros
dados['year_of_reference'] = dados['year_of_reference'].astype(int)
dados['year_model'] = dados['year_model'].astype(int)
dados['engine_size'] = dados['engine_size'].astype(float)

dados.dtypes

dados.head()

# Verificando se temos valores duplicados
dados.duplicated().sum()

# Removendo valores duplicados
dados.drop_duplicates(inplace=True)

# Verificando se temos valores duplicados
dados.duplicated().sum()

# nº de linhas e colunas após mudanças.
dados.shape
```

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

        # Resumo das variáveis numéricas - Imprime alguns valores de medidas de
tendências centrais
        dados[numericas_cols].describe()

        # Resumo das variáveis categóricas - Imprime alguns valores de estatística descritiva
        dados[categoricas_cols].describe()

        # Verificando a quantidade de valores faltantes por coluna
        dados.isna().sum()

        # Contagem do nº de marcas de carros
        dados['brand'].value_counts()

        # Gráfico da distribuição da quantidade de carros por marca
        plt.figure(figsize=(20,10))
        plt.bar(dados['brand'].unique(), dados['brand'].value_counts()) # plt.bar para gráfico
de barras. Variáveis nos eixos X e Y
        plt.title('Distribuição de carros por marca') # plt.title para inserir título no gráfico
        plt.ylabel('Total de carros') # # plt.ylabel para inserir título no gráfico

        # Gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro
        plt.figure(figsize=(20,10))
        plt.bar(dados['gear'].unique(), dados['gear'].value_counts()) # plt.bar para gráfico de
barras. Variáveis nos eixos X e Y
        plt.title('Distribuição de carros por tipo de engrenagem') # plt.title para inserir título no
gráfico

        dados.head()


        # Calculando a média por ano e mês
        media_preço_mes = dados.groupby(['month_of_reference','year_of_reference'])
['avg_price_brl'].mean().round(0) # round para arrendodar
        media_preço_mes.head()
        # Utilizando a função reset_index para criar uma ordem e facilitar a criação do
gráfico
        media_preço_mes = media_preço_mes.reset_index(name='Preço médio')
        media_preço_mes.rename(columns={'month_of_reference':'Mês de referencia'},
inplace=True)
        media_preço_mes.head()

        # Função para atribuir valores numéricos com base nos meses
        def atribuir_valor_numerico(categoria):
            if categoria == 'January':
                return 1
            elif categoria == 'February':
                return 2
            elif categoria == 'March':
                return 3
            elif categoria == 'April':
                return 4
            elif categoria == 'May':
                return 5
```

```
            elif categoria == 'June':
                return 6
            elif categoria == 'July':
                return 7
            elif categoria == 'August':
                return 8
            elif categoria == 'September':
                return 9
            elif categoria == 'October':
                return 10
            elif categoria == 'November':
                return 11
            elif categoria == 'December':
                return 12
            else:
                return None

        media_preço_mes['Mês de referencia - numerico'] = media_preço_mes['Mês de
referencia'].apply(atribuir_valor_numerico)

        filtro_ano_2022 = media_preço_mes[media_preço_mes['year_of_reference'] == 2022]
        filtro_ano_2022.sort_values(by='Mês de referencia - numerico', inplace=True)

        filtro_ano_2022

        # Gráfico da evolução da média de preço dos carros ao longo dos meses de 2022
(variável de tempo no eixo X)
        plt.figure(figsize=(20,10)) # Aumentar tamanho da imagem que será impressa na tela
        plt.title('Evolução da média de preço dos carros ao longo dos meses de 2022') #
plt.title para inserir título no gráfico
        sns.barplot(x='Mês de referencia', y='Preço médio', hue='Mês de referencia',
data=filtro_ano_2022, hue_order=list(filtro_ano_2022['Mês de referencia'].to_numpy()))
        plt.xticks(rotation=45);



        # Calculando a média por marca e tipo de engrenagem
        media_preço_engrenagem = dados.groupby(['brand','gear'])
['avg_price_brl'].mean().round(0) # round para arrendodar
        media_preço_engrenagem.head()
        # Utilizando a função reset_index para criar uma ordem e facilitar a criação do
gráfico
        media_preço_engrenagem = media_preço_engrenagem.reset_index(name='Preço
médio')
        media_preço_engrenagem.rename(columns={'brand':'Marca'}, inplace=True)
        media_preço_engrenagem.rename(columns={'gear':'Tipo de engrenagem'},
inplace=True)
        media_preço_engrenagem.head()

        #Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de
engrenagem
        plt.figure(figsize=(20,10)) # Aumentar tamanho da imagem que será impressa na tela
        plt.title('Distribuição da média de preço dos carros por marca e tipo de
engrenagem') # plt.title para inserir título no gráfico
        sns.barplot(x='Marca', y='Preço médio', hue='Tipo de engrenagem',
data=media_preço_engrenagem, hue_order=['automatic', 'manual']);
        plt.xticks(rotation=45);
```

```
        # Calculando a média por marca e tipo de combustivel
        media_preço_combustivel = dados.groupby(['brand','fuel'])
['avg_price_brl'].mean().round(0) # round para arrendodar
        media_preço_combustivel.head()
        # Utilizando a função reset_index para criar uma ordem e facilitar a criação do
gráfico
        media_preço_combustivel = media_preço_combustivel.reset_index(name='Preço
médio')
        media_preço_combustivel.rename(columns={'brand':'Marca'}, inplace=True)
        media_preço_combustivel.rename(columns={'fuel':'Tipo de combustivel'},
inplace=True)
        media_preço_combustivel.head()

        media_preço_combustivel['Tipo de combustivel'].value_counts()

        #Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de
combustível
        plt.figure(figsize=(20,10))
        plt.title('Distribuição da média de preço dos carros por marca e tipo de combustivel')
        sns.barplot(x='Marca', y='Preço médio', hue='Tipo de combustivel',
data=media_preço_combustivel, hue_order=['Diesel', 'Gasoline', 'Alcohol', 's/n']);
        plt.xticks(rotation=45);

        sns.boxplot(dados['avg_price_brl']).set_title('Variável resposta (target)')

        dados.columns

        dados['month_of_reference_numeric'] =
dados['month_of_reference'].apply(atribuir_valor_numerico)

        dados_num = dados.drop(['fipe_code', 'authentication', 'brand', 'model', 'fuel', 'gear',
'month_of_reference'],axis = 1)
        dados_num.head()

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

        # Variável X contém apenas variáveis numéricas de interesse para a análise,
excluindo a variável target
        X = dados_num.drop(['avg_price_brl'],axis = 1)
        X.head()

        # Variável Y contém apenas a variável target
        Y = dados_num['avg_price_brl']
        Y.head()

        # Divisão: 30% dos dados são de teste e 70% de treinamento
        X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, random_state = 42)

        # Observando os dados de treinamento
        print(X_train.shape)
        X_train.head(1)

        # Observando os dados de teste
```

```
print(X_test.shape)
X_test.head(1)

# Observando a variável target
Y_test.head()


# Algoritmo Random Forest, sem especificar nenhum parâmetro (número de árvores, número de ramificações, etc)
model_rf = RandomForestRegressor()

# Ajuste do modelo, de acordo com as variáveis de treinamento
model_rf.fit(X_train, Y_train)

# Predição dos valores de salário com base nos dados de teste
valores_preditos_rf = model_rf.predict(X_test)


# Valores preditos
valores_preditos_rf

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


mse_randon_forest_sem_parametro = mean_squared_error(Y_test, valores_preditos_rf)
mse_randon_forest_sem_parametro

mae_randon_forest_sem_parametro = mean_absolute_error(Y_test, valores_preditos_rf)
mae_randon_forest_sem_parametro

r2_randon_forest_sem_parametro = r2_score(Y_test, valores_preditos_rf)
r2_randon_forest_sem_parametro

model_rf_parametros = RandomForestRegressor(max_depth=29, min_samples_leaf=32, min_samples_split=28,
                      n_estimators=208, random_state=43)


# Ajuste do modelo, de acordo com as variáveis de treinamento
model_rf_parametros.fit(X_train, Y_train)

# Predição dos valores de salário com base nos dados de teste
valores_preditos_rf_parametros = model_rf_parametros.predict(X_test)

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

mse_randon_forest_parametros = mean_squared_error(Y_test, valores_preditos_rf_parametros)
mse_randon_forest_parametros
```

```
        msa_randon_forest_parametros = mean_absolute_error(Y_test,
valores_preditos_rf_parametros)
        msa_randon_forest_parametros

        r2_randon_forest_parametros = r2_score(Y_test, valores_preditos_rf_parametros)
        r2_randon_forest_parametros

        model_xgboost = XGBRegressor()

        # Ajuste do modelo, de acordo com as variáveis de treinamento
        model_xgboost.fit(X_train, Y_train)

        # Predição dos valores de salário com base nos dados de teste
        valores_preditos_xgboost = model_xgboost.predict(X_test)
        valores_preditos_xgboost

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

        mse_xgboost = mean_squared_error(Y_test, valores_preditos_xgboost)
        mse_xgboost

        msa_xgboost = mean_absolute_error(Y_test, valores_preditos_xgboost)
        msa_xgboost

        r2_xgboost = r2_score(Y_test, valores_preditos_xgboost)
        r2_xgboost

        min(mse_randon_forest_sem_parametro, mse_randon_forest_parametros,
mse_xgboost)

        max(r2_randon_forest_sem_parametro, r2_randon_forest_parametros, r2_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:

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

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: $R^2$

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

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

- Erro padrão da estimativa: $S_{yx}$

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

## B – RESOLUÇÃO

### Exercício 1

**Questão 5 -** Pelos resultados obtidos através da matriz de confusão, olhando na tabela o valor predito e o valor de referência, o algoritmo preditivo Random Forest apresentou menor quantidade de erros na identificação da classe de solo.

Também teve a maior das acurácias, com 92,13% de precisão.

### Linhas de código

```
#Instala pacotes necessários

install.packages("mlbench")
install.packages("e1071")
install.packages("randomForest")
install.packages("kernlab")
install.packages("caret")

#Carrega a bibliotecas
library(mlbench)
```

```
library(mice)
library(caret)

##### 1. Carregue a base de dados Satellite
data("Satellite")

# Analisando estrutura
head(Satellite)
str(Satellite)
summary(Satellite)

# Tratando dados
temp_dados <- Satellite[c("x.17","x.18","x.19","x.20","classes")]
set.seed(7)
imp <- mice(temp_dados)
dados <- complete(imp,1)

##### 2. Crie par2ções contendo 80% para treino e 20% para teste
indices <- createDataPartition(dados$classes, p=0.80, list=FALSE)
treino <- Satellite[indices,]
teste <- Satellite[-indices,]

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

#treinando modelos
rf <- train(classes~., data=treino, method="rf")
svm <- train(classes~., data=treino, method="svmRadial")
rna <- train(classes~., data=treino, method="nnet", trace=FALSE)

#Confirmando predição com partição de teste
predicoes.rf <- predict(rf, teste)
predicoes.svm <- predict(svm, teste)
predicoes.rna <- predict(rna, teste)

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

confusionMatrix(predicoes.rf, teste$classes)
confusionMatrix(predicoes.svm, teste$classes)
```

```
confusionMatrix(predicoes.rna, teste$classes)

# O modelo escolhido foi o Random Forest, que tem acurácia de 92,13%.

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

# Pelos resultados da matriz de confusão, olhando o predito e o referência,
# o Random Forest é o modelo que apresentou menor quantidade de erros na identi-
# ficação da classe de solo
```

## Exercício 2

**Questão 7 -** Pelo coeficiente de determinação, erro padrão e erro padrão percentual observados dos quatro modelos avaliados, o Random Forest foi o que apresentou melhores resultados. Mas vale a menção que os valores estão bem próximos do modelo de Spurr, logo, também pode ser considerado uma alternativa viável.

## Linhas de código

```
#Instala pacotes necessários
install.packages("mlbench")
install.packages("e1071")
install.packages("randomForest")
install.packages("kernlab")
install.packages("caret")

#Carrega a bibliotecas
library(caret)

# Equação de Spurr Volume = b0 + b1 * dap2 * Ht

##### 1. Carregar o arquivo Volumes.csv

dados <- read.csv("http://www.razer.net.br/datasets/Volumes.csv", sep=";", dec=",")
```

```
# Analisando estrutura
str(dados)
summary(dados)


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


dados$NR <- NULL
set.seed(7)

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

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


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

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

predicoes.rf <- predict(rf, teste)
predicoes.svm <- predict(svm, teste)
predicoes.rna <- predict(rna, teste)
predicoes.alom <- predict(alom, teste)
```

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

##### observados

```
R2 <- function(Y_real, Y_pred){
  n = length(Y_pred)
  return ( 1 - ( sum( (Y_real - Y_pred)^2 ) / sum( ( Y_real- mean(Y_real) )^2 ) ) )
}

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

Syxp <- function(Syx, Y_real){
  return ( Syx / mean(Y_real) * 100 )
}

#Quanto mais perto de 1 melhor é o modelo
R2.rf <- R2(teste$VOL, predicoes.rf)
R2.svm <- R2(teste$VOL, predicoes.svm)
R2.rna <- R2(teste$VOL, predicoes.rna)
R2.alom <- R2(teste$VOL, predicoes.alom)

#Quanto mais perto de 0 melhor
Syx.rf <- Syx(teste$VOL, predicoes.rf)
Syx.svm <- Syx(teste$VOL, predicoes.svm)
Syx.rna <- Syx(teste$VOL, predicoes.rna)
Syx.alom <- Syx(teste$VOL, predicoes.alom)

#Quanto mais perto de 0 melhor
Syxp.rf <- Syxp(Syx.rf, predicoes.rf)
Syxp.svm <- Syxp(Syx.svm, predicoes.svm)
Syxp.rna <- Syxp(Syx.rna, predicoes.rna)
Syxp.alom <- Syxp(Syx.alom, predicoes.alom)
```

##### 7. Escolha o melhor modelo.

```
# Pelas métricas calculadas, o modelo RF é o que produz melhores resultados.
# Relativamente próximo do modelo de Spurr.
```

# 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.

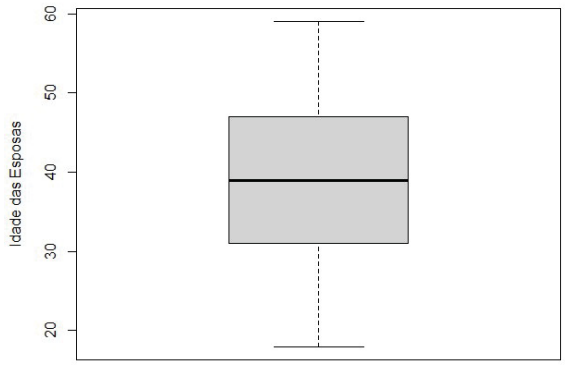
Obs:

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.

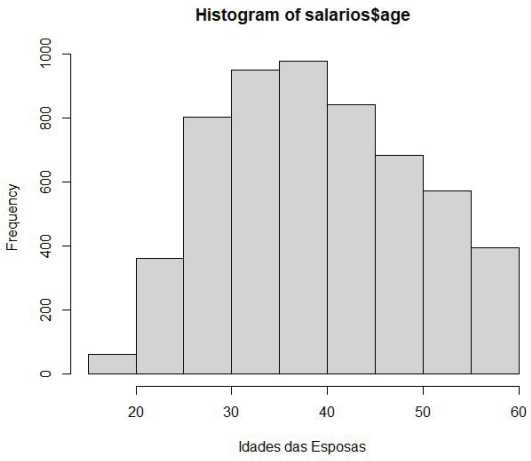
Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

## B – RESOLUÇÃO

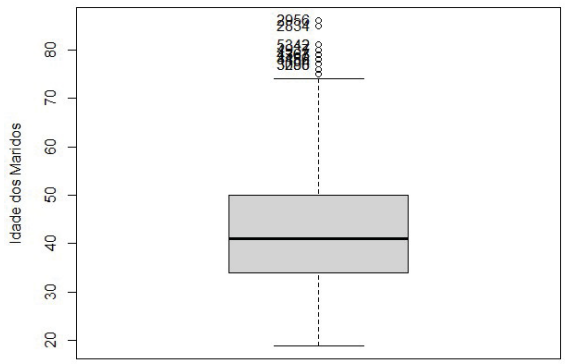
### 1-A)




6.

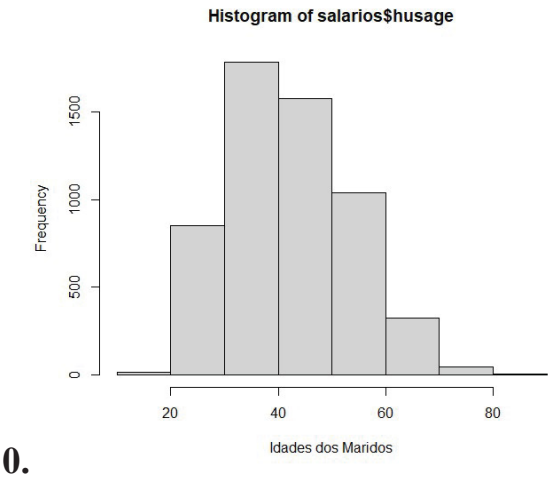



7.




8.

9.




10.

11.

Pelos gráficos podemos entender que os dados das Esposas estão mais distribuídas mas não avançam mais que 60 anos (distância interquartílica está maior nas Esposas pelo gráfico). Os Maridos, por sua vez estão mais concentrados perto da média e tem outliers do gráfico. Chegando a idades de 70 a 80 anos. A distribuição tem curtose à direita.

Tabela de frequência das idades das Esposas:

| 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

Tabela de frequência das idades dos Maridos:

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

Pela tabela de frequência vemos a idade das Esposas estarem concentradas no intervalo de maior ou igual a 35,7 e menor que 38,7 anos. Os Maridos se concentram no intervalo maior ou igual a 38,2 e menor que 43,1 anos.

Para Esposas:

Média: 39,4 anos

Mediana: 39 anos

Moda: 37 anos

Para Maridos:

Média: 39 anos

Mediana: 41 anos

Moda: 44 anos

**1-B)**

Os dados indicam que a média de idade das esposas é de 39,4 anos, com a mediana sendo 39 anos e a moda 37 anos. Isso sugere que a distribuição de idade das esposas é relativamente simétrica em torno da média, com uma leve concentração de valores em torno dos 37 anos. Corrobora com os gráficos do exercício anterior.

Para os maridos, a média de idade é de 39 anos, mas a mediana é mais alta, 41 anos, e a moda é ainda maior, 44 anos. Essa distribuição sugere uma assimetria, onde há uma concentração de maridos mais velhos, fazendo com que a mediana e a moda se desloquem para idades mais altas em comparação à média.

Fazendo comparação entre as médias 44 /37, temos que a idade dos maridos é 18,92% maior.

**2-A)**

Para Esposas:

Média: 39,4 anos

Mediana: 39 anos

Moda: 37 anos

Para Maridos:

Média: 39 anos

Mediana: 41 anos

Moda: 44 anos

Os dados indicam que a média de idade das esposas é de 39,4 anos, com a mediana sendo 39 anos e a moda 37 anos. Isso sugere que a distribuição de idade das esposas é relativamente simétrica em torno da média, com uma leve concentração de valores em torno dos 37 anos. Corrobora com os gráficos do exercício anterior.

Para os maridos, a média de idade é de 39 anos, mas a mediana é mais alta, 41 anos, e a moda é ainda maior, 44 anos. Essa distribuição sugere uma assimetria, onde há uma concentração de maridos mais velhos, fazendo com que a mediana e a moda se desloquem para idades mais altas em comparação à média.

Fazendo comparação entre as médias 44 /37, temos que a idade dos maridos é 18,92% maior.

**2-B)**

Para Esposas:

Variância: 99,8 anos

Desvio Padrão: 9,98 anos
Coeficiente Variância: 25,33%

Para Maridos:
Variância: 126,0 anos
Desvio Padrão: 11,22 anos
Coeficiente Variância: 26,44%

Para as esposas, os dados mostram uma variância de 99,8 anos e um desvio padrão de 9,98 anos, indicando que as idades estão dispersas em torno da média com essa variação padrão. O coeficiente de variação é de 25,33%, o que sugere uma dispersão moderada das idades em relação à média.

Para os maridos, a variância é de 126,0 anos e o desvio padrão é de 11,22 anos, o que aponta para uma dispersão maior das idades em comparação com as esposas. O coeficiente de variação é de 26,44%, indicando também uma dispersão moderada, mas ligeiramente maior em relação à média do que observado para as esposas. Esses dados ressaltam uma maior heterogeneidade nas idades dos maridos comparado às esposas.

26,38% mais variação de idade nos maridos e também há mais variação. O coeficiente de variação é 1 ponto percentual e alguns décimos maior nos maridos.

Exercício 3)

Teste de normalidade Esposas:

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

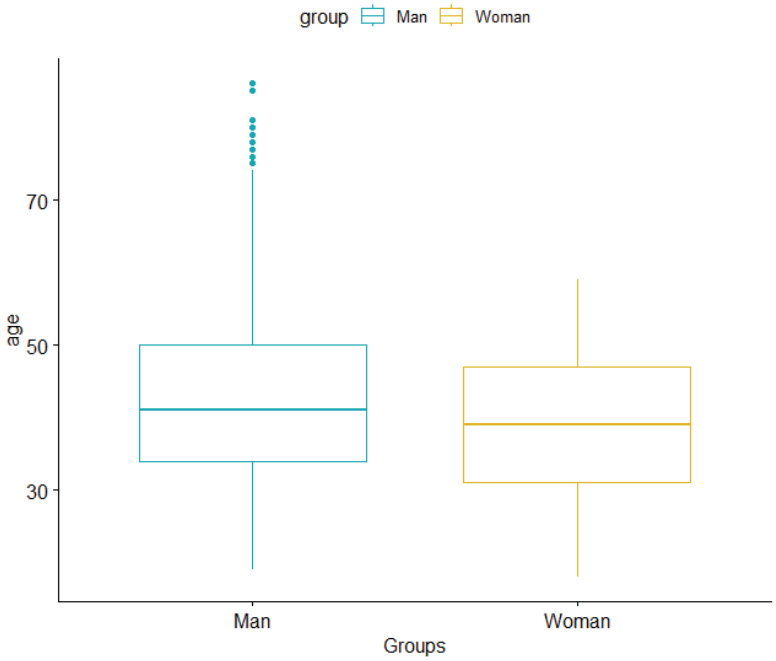
Teste de normalidade Maridos:

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

Regra de bolso: Como o p-value não é superior a 0.05, nenhuma das variáveis tem distribuição normal. Por conseguinte, seguiremos para um teste não paramétrico.

Relembrando o gráfico boxplot de ambos comparativamente testamos a hipóteses:
H0: As medianas de idade são iguais

Há: As medianas de idade não são iguais




Aplicando o teste:

Wilcoxon rank sum test with continuity correction

data: age by group
W = 18122044, p-value < 0.00000000000000022
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
2.000033 3.000024
sample estimates:
difference in location
2.999966

O teste de soma de postos de Wilcoxon com correção de continuidade mostra que existe uma diferença estatisticamente significativa entre as idades dos dois grupos analisados. O valor de W é 18122044 e o valor de p é extremamente baixo (< 0.00000000000000022), indicando que a diferença observada entre os grupos é muito improvável de ocorrer por acaso.

A hipótese alternativa de que as medianas são diferentes é suportada. O intervalo de confiança de 95% para a diferença de localização entre os grupos é de aproximadamente 2,00 a 3,00 anos, e a estimativa pontual dessa diferença é de aproximadamente 2,999966 anos. Isso sugere que um dos grupos tende a ter uma idade média significativamente maior que o outro por cerca de 3 anos.

## Linhas de código

```
        ##############################################################
######################
        ## Preparação                                          ###
        ##############################################################
######################

        # Lib Boxplot
        if(!require('car')) {
          install.packages('car')
          library('car')
        }

        # Lib tabela freq
        if(!require('fdth')) {
          install.packages('fdth')
          library('fdth')
        }


        if(!require('nortest')) {
          install.packages('nortest')
          library('nortest')
        }

        if(!require('ggpubr')) {
          install.packages('ggpubr')
          library("ggpubr")
        }

        # Para evitar a notação científica nos resultados
        options(scipen = 999)

        ## Carregando base de dados do trabalho *** Trocar pelo Path adequado
        setwd('C:/Users/luizc/Desktop/IAA004 - Trabalho/')
```

```
load("salarios.RData")

# Testando se carregou corretamente
summary(salarios)


##############################################################################
## 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 e histograma "age" (idade das esposas)
Boxplot( ~ age, data=salarios, id=list(method="y"),
        ylab="Idade das Esposas")

hist(salarios$age, breaks = 6,
    xlab = "Idades das Esposas" , ylab = "Frequency")



# Boxplot e histograma "husage" (idade dos maridos)
Boxplot( ~ husage, data=salarios, id=list(method="y"),
        ylab="Idade dos Maridos")

hist(salarios$husage, breaks = 6,
    xlab = "Idades dos Maridos", ylab = "Frequency")



## b) Elaborar a tabela de frequencias das variáveis "age" (idade da esposa)
##e "husage" (idade do marido) e comparar os resultados
```

```
# Tabela frequência Esposas
table_e <- fdt(salarios$age)
print (table_e)

# Tabela frequência Maridos
table_m <- fdt(salarios$husage)
print (table_m)
```

```
##########################################################################
######################
## 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

# média, mediana e moda para esposas
meanE <- mean(salarios$age)
medianE <- median(salarios$age)
table(salarios$age)
subset(table(salarios$age),
       table(salarios$age) == max(table(salarios$age)))

# média, mediana e moda para maridos
meanM <- mean(salarios$husage)
medianM <- median(salarios$husage)

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

```
# Para idades a mediana e média equivalentes significa que a distribuição é
# simétrica e unimodal (Embora no grupo masculino tenha outliers, não são
# representativos)

# A moda aponta claramente uma concentração maior na idade 44 para os maridos e
# 37 para as esposas. Fazendo 44 /37, temos que a idade dos maridos é 18,92%
# maior


## b) Calcular a variância, desvio padrão e coeficiente de variação das variá-
## veis "age" (idade da esposa) e "husage" (idade do marido) e comparar os
## resultados

# Variância, desvio padrão e coeficiente de variação para esposas

varE <- var(salarios$age)
sdE <- sd(salarios$age)
cvE <- (sdE/meanE)*100
varE
sdE
cvE

# Variância, desvio padrão e coeficiente de variação para maridos

varM <- var(salarios$husage)
sdM <- sd(salarios$husage)
cvM <- (sdM/meanM)*100
varM
sdM
cvM

((varM / varE)-1)*100


# 26,38% mais variação de idade nos maridos e também há mais variação
# O coeficiente de variação é 1 ponto e alguns décimos maior nos maridos
```

```
#############################################################
####################
## 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 (idade do marido) são iguais, construir os
## intervalos de confiança e comparar os resultados.


# Executanto teste de normalidade de Kolmogorov-Smirnov
lillie.test(salarios$husage)
lillie.test(salarios$age)

# Regra de bolso: Como o p-value não é superior a 0.05, nenhuma das variáveis
# tem distribuição normal. Por conseguinte seguiremos para um
# teste não paramétrico.


# Vamos entender as medianas dos dois grupos

# Agrupando
age <- data.frame(
  group = rep( c("Man", "Woman"), each = length(salarios$age) ),
  age = c(salarios$husage,  salarios$age)
)


# Contando, entendendo mediana e a distância inter quartílica.
group_by(age, group) %>%
  summarise(
    count = n(),
    median = median(age, na.rm = TRUE),
    IQR = IQR(age, na.rm = TRUE)
```

```
 )

# Visualmente os grupos
ggboxplot(age, x = "group", y = "age",
          color = "group", palette=c("#00AFBB", "#E7B800"),
          ylab = "age", xlab = "Groups")




# Para testar as hipóteses de igualdade usaremos Mann-Whitney "U" test.

# H0: A idade mediana dos homens é igual estatisticamente a idade
#     mediana das mulheres;
# Ha: A idade mediana dos homens não é igual estatisticamente a idade
#     mediana das mulheres;

# Testando na ordem do vetor de teste (Homens versus Mulheres)
resultado <- wilcox.test(age ~ group, data = age,
                         exact = FALSE, conf.int=TRUE)
resultado

# Como o p-value é < 0,05, temos a primeir hipótese negada. Logo a mediana não é
# estatisticamente igual. A hipótese alternativa é a mais provável.


# Para o intervalo de confiança de 95%, da diferença da idade, os valores estão
# entre 2,0 e 3,0, com mediana de 3
```

# 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

Visão geral dos dados carregados

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, 25, 26, 53, 3…
$ husunion <dbl> 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, 0, 1, 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, 240, 1450, 67…
$ 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, 11, 15, 17, 1…
$ 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, 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, 0, 1, 0, 0, 0, 0, …
$ 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, 40, 40, 91, 50…
$ 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, 1, 0, 1, 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, 1100, 434, 100…
$ 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, 33, 24, 52, 2…
$ 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, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 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, 10, 12, 18, 1…
$ 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, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, …
$ union <dbl> 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, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 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, 28, 10, 4, 15,…
$ 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, 0, 0, 0, 0, …
$ lwage <dbl> 1.897120, 2.484907, 2.431418, 1.629241, 2.302585, 2.496741, 2.957511, 1.783791, 1.945910, 2.600836, 2.014903, 2.011564, 2.525729

**Tamanho dos grupos na separação de 80/20:**

Treino

```
[1] 2059   17
```

Teste

```
[1] 515  17
```

**Avaliação pelo modelo de penalidade Ridge**

Penalidade: Adiciona uma penalidade do tipo L2 (soma dos quadrados dos coeficientes) ao termo de erro.

**Lambda ótimo do modelo Ridge:**

```
0.03162278
```

Betas das variáveis pelo modelo Ridge

```
15 x 1 sparse Matrix of class "dgCMatrix"
                          s0
husage    -0.004619353
husearns   0.230244252
```

```
huseduc   0.051374229
hushrs   -0.069514275
age       0.069223380
educ      0.327075655
husblck   0.217248348
hushisp   0.086154433
kidge6   -0.181933507
black    -0.277286478
hispanic -0.011410377
union     0.365674748
kidlt6   -0.042468361
husunion  0.001290081
exper    -0.016345064
```

Betas baixos indicam variáveis menos significativas na explicação da variável independente lwage.

Avaliando o erro quadrático médio e o coeficiente de determinação (R^2):

```
       RMSE   Rsquare
1 0.8411446 0.2921319
```

**Resultado:** O valor do salário-hora é de 9.422981 dólares, pelo modelo de regressão com penalidade de Ridge. Em relação ao intervalo de confiança temos inferior em 9.401122 dólares e superior em 9.444839 dólares.

**Avaliação pelo modelo de penalidade Lasso**

Penalidade: Adiciona uma penalidade do tipo L1 (soma dos valores absolutos dos coeficientes) ao termo de erro.

Lambda ótimo do modelo Lasso:

```
0.01258925
```

Betas das variáveis pelo modelo Lasso

```
15 x 1 sparse Matrix of class "dgCMatrix"
                       s0
husage    .
husearns  0.23003841
huseduc   0.03008230
hushrs   -0.06025180
```

```
age       0.04689039
educ      0.33916643
husblck   .
hushisp   .
kidge6   -0.14252071
black    -0.03169357
hispanic  .
union     0.34330221
kidlt6    .
husunion  .
exper     .
```

Variáveis com . são não significativas no modelo preditivo.

Avaliando o erro quadrático médio e o coeficiente de determinação (R^2):

```
          RMSE   Rsquare
1 0.8423523 0.2900977
```

**Resultado:** O valor do salário-hora é de 8.410696 dólares, pelo modelo de regressão com penalidade de Lasso (Least Absolute Shrinkage and Selection Operator). Em relação ao intervalo de confiança temos inferior em 8.388838 dólares e superior em 8.432554 dólares.

**Avaliação pelo modelo de penalidade Elastic Net**

Penalidade: Combina as penalidades L1 (Lasso) e L2 (Ridge).

Melhor estimativa do modelo para alfa e lambda:

```
      alpha     lambda
7 0.7429763 0.01233549
```

Betas do modelo:

```
15 x 77 sparse Matrix of class "dgCMatrix"
        [[ suppressing 67 column names ‘s0’, ‘s1’, ‘s2’ ... ]]
        [[ suppressing 67 column names ‘s0’, ‘s1’, ‘s2’ ... ]]
```

```
      husage   . .          .          .          .          .          .
.          .          .          .          .          .          .
      husearns . .          .          .         0.005265248 0.02487154
0.04296476 0.05963892 0.0750015 0.08913756 0.1021342 0.1140741 0.1250595
0.1351556
      huseduc  . .          .          .          .          .          .
.          .          .          .          .          .          .
      hushrs   . .          .          .          .          .          .
.          .          .          .          .          .          .
      age      . .          .          .          .          .          .
.          .          .          .          .          .          .
      educ     . 0.03507813 0.06778955 0.0982384 0.125145010 0.14617055
0.16558804 0.18350187 0.2000085 0.21520547 0.2291845 0.2420330 0.2531897
0.2625779
      husblck  . .          .          .          .          .          .
.          .          .          .          .          .          .

      husage   .          .          .          .          .          .
.          .          .          .          .          .          .
      husearns 0.1444226 0.1529183 0.1607027 0.1678319 0.173833446
0.179156492 0.1840182380  0.188394337  0.192895375  0.19701293  0.20076870
0.20420169
      huseduc  .          .          .          .         0.002054596
0.004588311 0.0070671711  0.009692001  0.012130047  0.01426931  0.01627562
0.01810500
      hushrs   .          .          .          .          .          .
.           -0.003737944 -0.009837325 -0.01541803 -0.02051807 -0.02518015
      age      .          .          .          .          .          .
0.0009931822  0.006234683  0.010683650  0.01475248  0.01846884  0.02186285
      educ     0.2711740 0.2790431 0.2862438 0.2928303 0.297810945
0.302025726 0.3057216471  0.309131551  0.312204359  0.31505880  0.31763223
0.31997957
      husblck  .          .          .          .          .          .
.          .          .          .          .          .          .

      husage    .          .          .          .          .
.          .          .          .          .          .
      husearns  0.20733894  0.21020519  0.21282324  0.21521407  0.21739697
0.21938968  0.22120847  0.22286826  0.22425417  0.22549168  0.22661338
```

```
      huseduc   0.01977252  0.02129243  0.02267775  0.02394035  0.02509108
0.02613982  0.02709559  0.02796661  0.02880179  0.02942634  0.03008557
      hushrs   -0.02944071 -0.03333332 -0.03688891 -0.04013596 -0.04310064
-0.04580704 -0.04827723 -0.05053150 -0.05259683 -0.05448882 -0.05621173
      age       0.02496184  0.02779091  0.03037314  0.03272971  0.03488003
0.03684191  0.03863164  0.04026417  0.04173584  0.04307650  0.04429662
      educ      0.32212067  0.32407348  0.32585439  0.32747840  0.32895922
0.33030941  0.33154039  0.33266264  0.33364741  0.33462081  0.33545519
      husblck   .           .           .           .           .
.           .           .           .           .           .

      husage    .           .           .           .           .
.           .           .           .           .           .
      husearns  0.22763557  0.22856772  0.22940155  0.23011730  0.23076951
0.23136351  0.23190493  0.23239844  0.23285117  0.23326236  0.23363590
      huseduc   0.03068984  0.03124069  0.03186811  0.03263283  0.03339727
0.03409753  0.03473628  0.03531876  0.03575080  0.03623007  0.03667901
      hushrs   -0.05778305 -0.05921617 -0.06050120 -0.06158733 -0.06257580
-0.06347681 -0.06429821 -0.06504699 -0.06573242 -0.06635490 -0.06692181
      age       0.04540920  0.04642363  0.04727635  0.04754591  0.04777970
0.04799196  0.04818484  0.04836014  0.04854642  0.04869006  0.04881966
      educ      0.33621383  0.33690521  0.33757363  0.33832341  0.33896436
0.33954668  0.34007726  0.34056075  0.34105880  0.34146336  0.34182488
      husblck   .           .           .           .           .
.           .           .           .           .           .

      husage    .           .           .           .           .
.           .           .           .           .           .
      husearns  0.23397613  0.234302403  0.23457019  0.23481649  0.23503758
0.23524178  0.23542489  0.23559450  0.23574624  0.23588727  0.23601308
      huseduc   0.03708983  0.037639033  0.03815880  0.03862704  0.03906080
0.03945096  0.03981272  0.04013708  0.04043848  0.04070774  0.04095866
      hushrs   -0.06743847 -0.067899109 -0.06831098 -0.06868668 -0.06902866
-0.06934063 -0.06962457 -0.06988364 -0.07011939 -0.07033455 -0.07053028
      age       0.04893746  0.049016780  0.04910094  0.04917805  0.04924724
0.04931099  0.04936814  0.04942096  0.04946823  0.04951205  0.04955117
      educ      0.34215342  0.342352008  0.34251725  0.34267117  0.34280687
0.34293384  0.34304554  0.34315074  0.34324284  0.34333025  0.34340631
      husblck   .           0.007295743  0.03634675  0.06238293  0.08708525
0.10887744  0.12959437  0.14773539  0.16506953  0.18012347  0.19460069
```

```
      husage    .           .          -0.0002716424 -0.002158684
-0.00358911 -0.00471071 -0.005783262 -0.006661823 ......
      husearns  0.23613051  0.23623490  0.2363815821  0.236542564
0.23667499  0.23679976  0.236911010  0.237016057 ......
      huseduc   0.04118170  0.04139039  0.0415540236  0.041691684
0.04181624  0.04192557  0.042029550  0.042116924 ......
      hushrs   -0.07070898 -0.07087150 -0.0710544985 -0.071224076
-0.07137585 -0.07151563 -0.071642336 -0.071758936 ......
      age       0.04958759  0.04962000  0.0498831915  0.051553402
0.05282458  0.05382458  0.054779523  0.055564264 ......
      educ      0.34347922  0.34354216  0.3436012153  0.343647277
0.34369778  0.34374575  0.343786969  0.343828708 ......
      husblck   0.20704468  0.21910724  0.2293417339  0.239228152
0.24848356  0.25603883  0.263460313  0.269436473 ......

       ..............................
       ........suppressing 10 columns and 1 rows in show(); maybe adjust
'options(max.print= *, width = *)'
       ..............................
        [[ suppressing 67 column names ‘s0’, ‘s1’, ‘s2’ ... ]]

      kidge6   . . . . . .          .          .          .
.        .         .         .         -0.007067256 -0.02156613
      black    . . . . . .          .          .          .
.        .         .         .          .           .
      hispanic . . . . . .          .          .          .
.        .         .         .          .           .
      union    . . . . . . 0.01387615 0.04486632 0.07331676
0.09941872 0.123352 0.1452849 0.1656124 0.1843023  0.201429528  0.21622602
      kidlt6   . . . . . .          .          .          .
.        .         .         .          .           .
      husunion . . . . . .          .          .          .
.        .         .         .          .           .
      exper    . . . . . .          .          .          .
.        .         .         .          .           .

      kidge6   -0.03467388 -0.04665826 -0.05762127 -0.06764355 -0.07680314
-0.08517209 -0.09281679 -0.09979837 -0.1061731 -0.1119926 -0.1173043
-0.1221519
      black     .           .          .          .          .
.          .           .          .          .          .          .
```

```
      hispanic  .           .           .           .           .
.           .           .           .           .           .          .
      union     0.22983826  0.24227059  0.25364022  0.26402875  0.27351847
0.28218523  0.29009879  0.29732326  0.3039175  0.3099356  0.3154272
0.3204375
      kidlt6    .           .           .           .           .
.           .           .           .           .           .          .
      husunion  .           .           .           .           .
.           .           .           .           .           .          .
      exper     .           .           .           .           .
.           .           .           .           .           .          .

      kidge6   -0.126561536 -0.1305701 -0.13423334 -0.13757501 -0.14062286
-0.1435986917 -0.147494967 -0.151076430 -0.15434390 -0.15732421 -0.16004235
      black    -0.005668806 -0.0125087 -0.01874408 -0.02443129 -0.02961807
-0.0341888440 -0.037914789 -0.041305121 -0.04439560 -0.04721274 -0.04978062
      hispanic  .           .           .           .           .
.           .           .           .           .           .          .
      union     0.325287423  0.3297732  0.33387756  0.33762152  0.34103604
0.3440766596  0.346741136  0.349181234  0.35140640  0.35343511  0.35528459
      kidlt6    .           .           .           .           .
-0.0005768108 -0.004429082 -0.007967649 -0.01119672 -0.01414263 -0.01682990
      husunion  .           .           .           .           .
.           .           .           .           .           .          .
      exper     .           .           .           .           .
.           .           .           .           .           .          .

      kidge6   -0.16244906 -0.16471269 -0.16677956 -0.16866426 -0.17047047
-0.17207870 -0.17354382 -0.17488091 -0.17609849 -0.17720968 -0.17822126
      black    -0.05213580 -0.05426880 -0.05621205 -0.05798308 -0.06650679
-0.09563775 -0.12177377 -0.14652834 -0.16840565 -0.18916520 -0.20738180
      hispanic  .           .           .           .           .
.           .           .           .           .           .          .
      union     0.35695564  0.35849197  0.35989405  0.36117226  0.36230063
0.36316725  0.36395958  0.36467629  0.36533379  0.36592821  0.36647429
      kidlt6   -0.01921953 -0.02145752 -0.02350105 -0.02536472 -0.02712410
-0.02867070 -0.03007995 -0.03136525 -0.03253623 -0.03360417 -0.03457696
      husunion  .           .           .           .           .
.           .           .           .           .           .          .
      exper     .           .           .           .           .
.           .           .           .           .           .          .
```

```
      kidge6   -0.17914455 -0.1799848 -0.18075183 -0.18144961 -0.18208675
-0.18266039 -0.18322243 -0.18373148 -0.18418786 -0.18460658
-0.18498279 ......
      black    -0.22475129 -0.2398732 -0.25438012 -0.26688641 -0.27897455
-0.28925476 -0.29913493 -0.30837523 -0.31596059 -0.32338481
-0.32940138 ......
      hispanic  .          .          .          .          .
.          .          .          .          .          .          ....
..
      union     0.36696743  0.3674212  0.36783037  0.36820759  0.36854727
0.36885728  0.36911340  0.36934646  0.36956744  0.36976486
0.36995081 ......
      kidlt6   -0.03546417 -0.0362722 -0.03700918 -0.03768022 -0.03829234
-0.03886517 -0.03958127 -0.04020237 -0.04074453 -0.04124557
-0.04168828 ......
      husunion  .          .          .          .          .
.          .          .          .          .          .          ....
..
      exper     .          .          .          .          .
.          .          .          .          .          .          ....
..
```

Avaliando o erro quadrático médio e o coeficiente de determinação (R^2):

```
       RMSE   Rsquare
1 0.8419627 0.2907543
```

**Resultado:** O valor do salário-hora é de 8.003162 dólares, pelo modelo de regressão com penalidade de Elastic Net. Em relação ao intervalo de confiança temos inferior em é 7.981303 dólares e superior em 8.02502 dólares.

**Conclusão**

Avaliando os modelos pelo menor erro quadrático médio e/ou maior coeficiente, temos:

```
        Model      RMSE  R_square
1       Ridge 0.9893314 0.2590861
2       Lasso 0.9899393 0.2581753
3 Elastic Net 0.9893725 0.2590245
```

Logo o modelo de regressão linear com penalidade escolhido foi o Ridge.

## Linhas de código

```
# Instalar os pacotes necessarios
#install.packages("plyr")
#install.packages("readr")
#install.packages("dplyr")
#install.packages("caret")
#install.packages("ggplot2")
#install.packages("repr")
#install.packages("glmnet")

# Carregar os pacotes
library(plyr)
library(readr)
library(dplyr)
library(caret)
library(ggplot2)
library(repr)
library(glmnet)


# Carregar o conjunto de dados
trabalhosalarios.RData")

# Guardar o conjunto de dados em um objeto
data <- trabalhosalarios

# Ver o conjunto de dados inteiro
View(data)

# Visualizar parte do conjunto de dados
glimpse(data)

set.seed(302)

# Criar um índice para particionar o conjunto de dados em 80% para treinamento
index <- sample(1:nrow(data), 0.8 * nrow(data))
```

```
# Criar o conjunto de dados de treinamento
train <- data[index,]

# Criar o conjunto de dados de teste
test <- data[-index,]

# Checar as dimensões dos conjuntos de treinamento e teste
dim(train)
dim(test)

# Padronizar as variáveis
# Criar um objeto com as variáveis a serem padronizadas
# Variáveis binárias não são padronizadas
cols <- c('husage', 'husearns', 'huseduc', 'hushrs',
        'age', 'educ', 'lwage', 'exper')

# Padronizar o conjunto de dados 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])

# Ver o resumo estatístico das variáveis padronizadas de cada conjunto de dados
summary(train)
summary(test)

#############################################################
###
#                  REGRESSÃO RIDGE                  #
#############################################################
###
# Estimar o salário-hora da esposa (lwage)

# Criar um objeto com as variáveis que serão usadas no modelo
cols_reg <- c('husage', 'husearns', 'huseduc', 'hushrs',
              'age', 'educ', 'lwage','husblck',
              'hushisp', 'kidge6', 'black', 'hispanic',
              'union', 'kidlt6', 'husunion', 'exper')
```

```
        # Gerar variáveis dummies para organizar os conjuntos de dados em objetos tipo
matriz
        dummies <- dummyVars(lwage ~ husage + husearns + huseduc + hushrs +
                               age + educ + husblck + hushisp +
                               kidge6 + black + hispanic + union + kidlt6 + husunion + exper,
                             data = data[,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))

        # Guardar a matriz de dados de treinamento das variáveis explicativas para o
modelo
        x <- as.matrix(train_dummies)

        # Guardar o vetor de dados de treinamento da variável dependente para o modelo
        y_train <- train$lwage

        # Guardar a matriz de dados de teste das variáveis explicativas para o modelo
        x_test <- as.matrix(test_dummies)

        # Guardar o vetor de dados de teste da variável dependente para o modelo
        y_test <- test$lwage

        # Calcular o valor ótimo de lambda
        # alpha = "0", é para regressão Ridge
        # Testar os lambdas de 10^-3 até 10^2, a cada 0.1
        lambdas <- 10^seq(2, -3, by = -0.1)

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

        # Ver qual o lambda ótimo
        best_lambda_ridge <- ridge_lamb$lambda.min
        best_lambda_ridge

        # Estimar o modelo Ridge
```

```
ridge_reg <- glmnet(x, y_train, nlambda = 25, alpha = 0,
                    family = 'gaussian',
                    lambda = best_lambda_ridge)
# Ver o resultado (valores) da estimativa (coeficientes)
ridge_reg[["beta"]]

# Calcular 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 métricas de performance do modelo:
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}

# Predição e avaliação nos dados de treinamento
predictions_train <- predict(ridge_reg,
                             s = best_lambda_ridge,
                             newx = x)

# As métricas do conjunto de treinamento
eval_results(y_train, predictions_train, train)

# Predição e avaliação nos dados de teste
predictions_test <- predict(ridge_reg,
                            s = best_lambda_ridge,
                            newx = x_test)

# As métricas do conjunto de teste
eval_ridge <- eval_results(y_test, predictions_test, test)


# Fazer uma predição
```

```
# husage = 40 anos (idade do marido)
husage = (40-pre_proc_val[["mean"]][["husage"]])/
  pre_proc_val[["std"]][["husage"]]

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

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

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

# hushisp = 0 (o marido não é hispânico)
hushisp = 0

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

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

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

# black = 0 (esposa não é preta)
black = 0

# educ = 13 (esposa possui 13 anos de estudo)
educ = (13-pre_proc_val[["mean"]][["educ"]])/
  pre_proc_val[["std"]][["educ"]]

# hispanic = 1 (esposa é hispânica)
```

```
hispanic = 1

# union = 0 (o casal não possui união registrada)
union = 0

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

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

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

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

predict_our_ridge <- predict(ridge_reg,
                             s = best_lambda_ridge,
                             newx = our_pred)

# O resultado da predição:
predict_our_ridge
```

```
# Convertê-lo para o valor nominal, consistente com o conjunto de dados original
pred_ridge <- (predict_our_ridge *
                  pre_proc_val[["std"]][["lwage"]]) +
  pre_proc_val[["mean"]][["lwage"]]


# Antilog
pred_ridge <- exp(pred_ridge)

# O resultado é:
pred_ridge


# O intervalo de confiança para o nosso exemplo é:
n <- nrow(train) # tamanho da amostra
m <- pred_ridge # valor médio predito
s <- pre_proc_val[["std"]][["lwage"]] # desvio padrão
dam <- s/sqrt(n) # distribuição da amostragem da média
Cllwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
Clupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior

# Os valores são:
Cllwr_ridge
Clupr_ridge


# O valor salário-hora é de 9.422981 dólares.
# O valor mínimo do salário-hora é 9.401122 dólares.
# O valor máximo do salário-hora é 9.444839 dólares.


#############################################################
###
#                  REGRESSAO LASSO                    #
#############################################################
###
# Estimar o salário-hora da esposa (lwage)
```

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

# Gerar variáveis dummies para organizar os conjuntos de dados em objetos tipo
matriz
dummies <- dummyVars(lwage ~ husage + husearns + huseduc + hushrs +
                       age + educ + husblck + hushisp +
                       kidge6 + black + hispanic + union + kidlt6 + husunion + exper,
                     data = data[,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))


# Guardar a matriz de dados de treinamento das variáveis explicativas para o
modelo
x <- as.matrix(train_dummies)

# Guardar o vetor de dados de treinamento da variável dependente para o modelo
y_train <- train$lwage

# Guardar a matriz de dados de teste das variáveis explicativas para o modelo
x_test <- as.matrix(test_dummies)

# Guardar o vetor de dados de teste da variável dependente para o modelo
y_test <- test$lwage

# Calcular o valor ótimo de lambda
# Atribuir alpha = 1 para implementar a regressão Lasso
# Testar os lambdas de 10^-3 até 10^2, a cada 0.1
lambdas <- 10^seq(2, -3, by = -0.1)

# Calcular o lambda
lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
```

```
                    lambda = lambdas,
                    standardize = TRUE, nfolds = 5)

# Guardar o lambda "ótimo"
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso



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

# Visualizar os coeficientes estimados
lasso_model[["beta"]]

# Fazer as predições na base de treinamento e avaliar a regressão Lasso
predictions_train <- predict(lasso_model,
                         s = best_lambda_lasso,
                         newx = x)

# Calcular 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 métricas de performance do modelo:
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}

# As métricas da base de treinamento são:
eval_results(y_train, predictions_train, train)
```

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

# As métricas da base de teste são:
eval_lasso <- eval_results(y_test, predictions_test, test)

# Fazer uma predição

# husage = 40 anos (idade do marido)
husage = (40 - pre_proc_val[["mean"]][["husage"]]) /
  pre_proc_val[["std"]][["husage"]]

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

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

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

# hushisp = 0 (o marido não é hispânico)
hushisp = 0

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

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

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

```
# black = 0 (esposa não é preta)
black = 0

# educ = 13 (esposa possui 13 anos de estudo)
educ = (13 - pre_proc_val[["mean"]][["educ"]]) /
  pre_proc_val[["std"]][["educ"]]

# hispanic = 1 (esposa é hispânica)
hispanic = 1

# union = 0 (o casal não possui união registrada)
union = 0

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

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

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

# Construir uma matriz de dados para a predição
our_pred <- as.matrix(data.frame(husage = husage,
                                 husearns = husearns,
                                 huseduc = huseduc,
                                 husblck = husblck,
                                 hushisp = hushisp,
                                 hushrs = hushrs,
                                 kidge6 = kidge6,
                                 age = age,
                                 black = black,
                                 educ = educ,
                                 hispanic = hispanic,
                                 union = union,
                                 kidlt6 = kidlt6,
```

```
                        husunion = husunion,
                        exper = exper))

predict_our_lasso <- predict(lasso_model,
                        s = best_lambda_lasso,
                        newx = our_pred)
predict_our_lasso

# Convertê-lo para valor compatível com o dataset original
pred_lasso <- (predict_our_lasso *
                    pre_proc_val[["std"]][["lwage"]]) +
  pre_proc_val[["mean"]][["lwage"]]


# Antilog
pred_lasso <- exp(pred_lasso)

# O resultado é:
pred_lasso

# Criar o intervalo de confiança
n <- nrow(train)
m <- pred_lasso
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
Cllwr_lasso <- m + (qnorm(0.025))*dam
Clupr_lasso <- m - (qnorm(0.025))*dam

# O intervalo de confiança é:
Cllwr_lasso
Clupr_lasso


# O valor salário-hora é de 8.410696 dólares.
# O valor mínimo do salário-hora é 8.388838 dólares.
# O valor máximo do salário-hora é 8.432554 dólares.
```

```
        ##########################################################################
###
        #                  REGRESSAO ELASTICNET                  #
        ##########################################################################
###
        # Estimar o salário-hora da esposa (lwage)

        # Criar um objeto com as variáveis que usaremos no modelo
        cols_reg <- c('husage', 'husearns', 'huseduc', 'hushrs',
                      'age', 'educ', 'lwage','husblck',
                      'hushisp', 'kidge6', 'black', 'hispanic',
                      'union', 'kidlt6', 'husunion', 'exper')

        # Gerar variáveis dummies para organizar os conjuntos de dados em objetos tipo
matriz
        dummies <- dummyVars(lwage ~ husage + husearns + huseduc + hushrs +
                               age + educ + husblck + hushisp +
                               kidge6 + black + hispanic + union + kidlt6 + husunion + exper,
                             data = data[,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))



        # Guardar a matriz de dados de treinamento das variáveis explicativas para o
modelo
        x <- as.matrix(train_dummies)

        # Guardar o vetor de dados de treinamento da variável dependente para o modelo
        y_train <- train$lwage

        # Guardar a matriz de dados de teste das variáveis explicativas para o modelo
        x_test <- as.matrix(test_dummies)

        # Guardar o vetor de dados de teste da variável dependente para o modelo
        y_test <- test$lwage
```

```
# Configurar o treinamento do modelo por cross validation, com 10 folders, 5
repetições e busca aleatória dos componentes das amostras de treinamento
train_cont <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 5,
                           search = "random",
                           verboseIter = TRUE)

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

# O melhor parâmetro alpha escolhido é:
elastic_reg$bestTune

# Os coeficientes do modelo são:
elastic_reg[["finalModel"]][["beta"]]


predictions_train <- predict(elastic_reg, x)

# Calcular 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 métricas de performance do modelo:
 data.frame(
  RMSE = RMSE,
  Rsquare = R_square
 )
```

```
}

# As métricas de performance na base de treinamento são:
eval_results(y_train, predictions_train, train)

# Fazer as predições na base de teste
predictions_test <- predict(elastic_reg, x_test)

# As métricas de performance na base de teste são:
eval_elastic <- eval_results(y_test, predictions_test, test)

# Fazer uma predição

# husage = 40 anos (idade do marido)
husage <- (40 - pre_proc_val[["mean"]][["husage"]]) /
  pre_proc_val[["std"]][["husage"]]

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

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

# husblck = 1 (o marido é preto)
husblck <- 1

# hushisp = 0 (o marido não é hispânico)
hushisp <- 0

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

# kidge6 = 1 (tem filhos maiores de 6 anos)
kidge6 <- 1
```

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

# black = 0 (esposa não é preta)
black <- 0

# educ = 13 (esposa possui 13 anos de estudo)
educ <- (13 - pre_proc_val[["mean"]][["educ"]]) /
  pre_proc_val[["std"]][["educ"]]

# hispanic = 1 (esposa é hispânica)
hispanic <- 1

# union = 0 (o casal não possui união registrada)
union <- 0

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

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

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

# Construir uma matriz de dados para a predição
our_pred <- as.matrix(data.frame(husage = husage,
                                 husearns = husearns,
                                 huseduc = huseduc,
                                 husblck = husblck,
                                 hushisp = hushisp,
                                 hushrs = hushrs,
                                 kidge6 = kidge6,
                                 age = age,
                                 black = black,
                                 educ = educ,
```

```
                    hispanic = hispanic,
                    union = union,
                    kidlt6 = kidlt6,
                    husunion = husunion,
                    exper = exper))

predict_our_elastic <- predict(elastic_reg, our_pred)
predict_our_elastic

# Converter para o nível dos valores originais do dataset
pred_elastic <- (predict_our_elastic *
                   pre_proc_val[["std"]][["lwage"]]) +
  pre_proc_val[["mean"]][["lwage"]]


# Antilog
pred_elastic <- exp(pred_elastic)

# O resultado é:
pred_elastic


# Criar o intervalo de confiança
n <- nrow(train)
m <- pred_elastic
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s / sqrt(n)
Cllwr_elastic <- m + (qnorm(0.025)) * dam
Clupr_elastic <- m - (qnorm(0.025)) * dam

# Os valores mínimo e máximo são:
Cllwr_elastic
Clupr_elastic

# O valor salário-hora é de 8.003162 dólares.
# O valor mínimo do salário-hora é 7.981303 dólares.
# O valor máximo do salário-hora é 8.02502 dólares.
```

```
# Combinar os resultados (RMSE e R²)
results <- data.frame(
  Model = c("Ridge", "Lasso", "Elastic Net"),
  RMSE = c(eval_ridge$RMSE, eval_lasso$RMSE, eval_elastic$RMSE),
  R_square = c(eval_ridge$Rsquare, eval_lasso$Rsquare, eval_elastic$Rsquare)
)

# Escolher o modelo com o menor RMSE e/ou o maior R²
best_model <- results[which.min(results$RMSE), ]

# Mostrar a tabela de resultados
print(results)

# Mostrar o melhor modelo
print(best_model)


# Combinar os resultados min, max e salario-hora
results_lwage <- data.frame(
  Model = c("Ridge", "Lasso", "Elastic Net"),
  min = c(Cllwr_ridge, Cllwr_lasso, Cllwr_elastic),
  max = c(Clupr_ridge, Clupr_lasso, Clupr_elastic),
  lwage = c(pred_ridge, pred_lasso,pred_elastic)
)

# Mostrar a tabela de resultados
print(results_lwage)
```

# 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

O objetivo desse trabalho é demonstrar o processo de "construção de atributos" e como ele é fundamental para o Reconhecimento de Padrões (RP).
Primeiro um conjunto de "amostras" previamente conhecido (classificado)
Figura1: Transformação de dados

Transformando amostras em **padrões**

```
import random

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

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

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

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

```
[['Where is the nearest restaurant?', 'inglês'], ['Estoy estudiando para mis exámenes.', 'espanhol'], ['¿Cuál es tu comida favori
```

### Saída

[['WHERE IS THE NEAREST RESTAURANT?', 'INGLÊS'], ['ESTOY ESTUDIANDO PARA MIS EXÁMENES.', 'ESPANHOL'], ['¿CUÁL ES TU COMIDA FAVORITA?', 'ESPANHOL'], ['MEU TIME DE FUTEBOL FAVORITO GANHOU O JOGO.', 'PORTUGUÊS'], ['¿PUEDES AYUDARME CON ESTO?', 'ESPANHOL'], ['WHAT IS YOUR FAVORITE COLOR?', 'INGLÊS'], ['VOU VISITAR MEUS AVÓS NO DOMINGO.', 'PORTUGUÊS'], ['ESTOU CANSADO DEPOIS DE UM LONGO DIA DE TRABALHO.', 'PORTUGUÊS'], ['¡QUE TENGAS UN BUEN DÍA!', 'ESPANHOL'], ["I'M LEARNING TO PLAY THE GUITAR.", 'INGLÊS'], ['VAMOS SAIR PARA JANTAR NO SÁBADO.', 'PORTUGUÊS'], ['VAMOS A DAR UN PASEO.', 'ESPANHOL'], ['THIS IS A BEAUTIFUL PLACE.', 'INGLÊS'], ['THE WEATHER IS NICE TODAY.', 'INGLÊS'], ['ME ENCANTA LEER LIBROS.', 'ESPANHOL'], ['ME GUSTARÍA IR DE VACACIONES.', 'ESPANHOL'], ['ESTOU ESTUDANDO PARA O VESTIBULAR.', 'PORTUGUÊS'], ['GOSTO DE FAZER CAMINHADAS NA NATUREZA.', 'PORTUGUÊS'], ['I LIKE TO COOK DINNER AT HOME.', 'INGLÊS'], ['DISFRUTO IR A LA PLAYA.', 'ESPANHOL'], ['HOW WAS YOUR WEEKEND?', 'INGLÊS'], ['A FESTA COMEÇA ÀS 20H.', 'PORTUGUÊS'], ['ESTOU APRENDENDO A COZINHAR PRATOS NOVOS.', 'PORTUGUÊS'], ['HELLO, HOW ARE YOU?', 'INGLÊS'], ['HOJE É MEU ANIVERSÁRIO!', 'PORTUGUÊS'], ['¿QUÉ HORA ES?', 'ESPANHOL'], ["I'M PLANNING A VACATION.", 'INGLÊS'], ['VAMOS FAZER UM CHURRASCO NO FINAL DE SEMANA.', 'PORTUGUÊS'], ['GOSTO DE OUVIR MÚSICA CLÁSSICA.', 'PORTUGUÊS'], ['O FILME QUE ASSISTI ONTEM FOI ÓTIMO.', 'PORTUGUÊS'], ['DO YOU SPEAK ENGLISH?', 'INGLÊS'], ['¿DÓNDE ESTÁ EL RESTAURANTE MÁS CERCANO?', 'ESPANHOL'], ['VOCÊ GOSTA DE DANÇAR?', 'PORTUGUÊS'], ['ADORO PASSAR TEMPO COM MINHA FAMÍLIA.', 'PORTUGUÊS'], ['¿DÓNDE PUEDO ENCONTRAR UN TAXI?', 'ESPANHOL'], ['ADORO EXPLORAR NOVOS LUGARES.', 'PORTUGUÊS'], ['¿TIENES ALGUNA RECOMENDACIÓN DE RESTAURANTES?', 'ESPANHOL'], ['I

LOVE TO READ BOOKS.', 'INGLÊS'], ['O RESTAURANTE TEM UMA VISTA INCRÍVEL.', 'PORTUGUÊS'], ['ESTOY APRENDIENDO A TOCAR EL PIANO.', 'ESPANHOL'], ['O PARQUE FICA CHEIO AOS FINAIS DE SEMANA.', 'PORTUGUÊS'], ['WHERE CAN I FIND A TAXI?', 'INGLÊS'], ['ESTE ES MI LIBRO FAVORITO.', 'ESPANHOL'], ['TENGO UNA REUNIÓN A LAS 2 PM.', 'ESPANHOL'], ["I'M SORRY FOR THE INCONVENIENCE.", 'INGLÊS'], ['NECESITO COMPRAR ALGUNAS FRUTAS.', 'ESPANHOL'], ['VAMOS FAZER UMA VIAGEM DE CARRO.', 'PORTUGUÊS'], ['VAMOS AO CINEMA NO SÁBADO.', 'PORTUGUÊS'], ['COULD YOU PASS ME THE SALT, PLEASE?', 'INGLÊS'], ['EL CLIMA ESTÁ AGRADABLE HOY.', 'ESPANHOL'], ['O TRÂNSITO ESTÁ TERRÍVEL HOJE.', 'PORTUGUÊS'], ['DO YOU HAVE ANY RECOMMENDATIONS FOR RESTAURANTS?', 'INGLÊS'], ['HAVE A GREAT DAY!', 'INGLÊS'], ['LAMENTO LAS MOLESTIAS.', 'ESPANHOL'], ['I WANT TO LEARN FRENCH.', 'INGLÊS'], ['VOU VIAJAR PARA O EXTERIOR NAS FÉRIAS.', 'PORTUGUÊS'], ['I LIKE LISTENING TO MUSIC.', 'INGLÊS'], ['CAN YOU HELP ME WITH THIS?', 'INGLÊS'], ['QUIERO APRENDER ITALIANO.', 'ESPANHOL'], ['PRECISO FAZER EXERCÍCIOS FÍSICOS REGULARMENTE.', 'PORTUGUÊS'], ['THE CAT IS SLEEPING.', 'INGLÊS'], ['TENHO UMA REUNIÃO IMPORTANTE AMANHÃ.', 'PORTUGUÊS'], ['I NEED TO BUY SOME GROCERIES.', 'INGLÊS'], ['QUERO APRENDER A TOCAR VIOLÃO.', 'PORTUGUÊS'], ['SHE SINGS BEAUTIFULLY.', 'INGLÊS'], ['ESTOY EMOCIONADO POR EL CONCIERTO.', 'ESPANHOL'], ['I ENJOY GOING TO THE BEACH.', 'INGLÊS'], ['ESTOU ANSIOSO PARA AS FÉRIAS DE VERÃO.', 'PORTUGUÊS'], ['VOCÊ JÁ VISITOU O RIO DE JANEIRO?', 'PORTUGUÊS'], ['PRECISO RESOLVER ESSE PROBLEMA O MAIS RÁPIDO POSSÍVEL.', 'PORTUGUÊS'], ['ESTOU INDO PARA O TRABALHO AGORA.', 'PORTUGUÊS'], ['¿ME PASAS LA SAL, POR FAVOR?', 'ESPANHOL'], ['GOSTO DE PRATICAR ESPORTES AO AR LIVRE.', 'PORTUGUÊS'], ['ESTOY PLANEANDO UNAS VACACIONES.', 'ESPANHOL'], ['WHAT TIME IS IT?', 'INGLÊS'], ['A COMIDA ESTAVA DELICIOSA!', 'PORTUGUÊS'], ['ME GUSTA BAILAR SALSA.', 'ESPANHOL'], ["LET'S GO FOR A WALK.", 'INGLÊS'], ['PRECISO COMPRAR LEITE E PÃO.', 'PORTUGUÊS'], ['¿HABLAS ESPAÑOL?', 'ESPANHOL'], ["I'M STUDYING FOR MY EXAMS.", 'INGLÊS'], ['ELLA CANTA HERMOSAMENTE.', 'ESPANHOL'], ['I HAVE A MEETING AT 2 PM.', 'INGLÊS'], ['EL PERRO ESTÁ JUGANDO.', 'ESPANHOL'], ['VOY AL PARQUE TODOS LOS DÍAS.', 'ESPANHOL'], ['¿CÓMO ESTUVO TU FIN DE SEMANA?', 'ESPANHOL'], ['O LIVRO QUE ESTOU LENDO É MUITO INTERESSANTE.', 'PORTUGUÊS'], ["I'M GOING TO THE MOVIES TONIGHT.", 'INGLÊS'], ['ME GUSTA COCINAR LA CENA EN CASA.', 'ESPANHOL'], ['I ENJOY PLAYING SOCCER.', 'INGLÊS'], ['HOLA, ¿CÓMO ESTÁS?', 'ESPANHOL'], ["I'M EXCITED FOR THE CONCERT.", 'INGLÊS']]

**DATA FRAME**

| | 0 | 1 |
|---|---|---|
| 0 | Where is the nearest restaurant? | inglês |
| 1 | Estoy estudiando para mis exámenes. | espanhol |
| 2 | ¿Cuál es tu comida favorita? | espanhol |
| 3 | Meu time de futebol favorito ganhou o jogo. | português |
| 4 | ¿Puedes ayudarme con esto? | espanhol |
| ... | ... | ... |
| 87 | I'm going to the movies tonight. | inglês |
| 88 | Me gusta cocinar la cena en casa. | espanhol |
| 89 | I enjoy playing soccer. | inglês |
| 90 | Hola, ¿cómo estás? | espanhol |
| 91 | I'm excited for the concert. | inglês |

**ACURACIA**

```
Acurácia nos dados de treinamento: 81.16%
[[14  0  8]
 [ 1 19  3]
 [ 1  0 23]]
              precision    recall  f1-score   support

    espanhol       0.88      0.64      0.74        22
      inglês       1.00      0.83      0.90        23
   português       0.68      0.96      0.79        24

    accuracy                           0.81        69
   macro avg       0.85      0.81      0.81        69
weighted avg       0.85      0.81      0.81        69

métricas mais confiáveis
[[6 0 2]
 [0 3 4]
 [0 0 8]]
              precision    recall  f1-score   support

    espanhol       1.00      0.75      0.86         8
      inglês       1.00      0.43      0.60         7
   português       0.57      1.00      0.73         8

    accuracy                           0.74        23
   macro avg       0.86      0.73      0.73        23
weighted avg       0.85      0.74      0.73        23
```

## Linhas de código

```
#
# 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.",
]

import random

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

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

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

random.shuffle(pre_padroes)
print(pre_padroes)

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

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

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

def contaPalavras(texto, palavras_comuns):
    contagem = 0
    for palavra in palavras_comuns:
        contagem += texto.lower().split().count(palavra)
        #print(contagem)
    return contagem

# Função: detectaComuns
# Dectar a frequência de palavras (artigos, preposições,
# contrações etc) mais frequentes em frases destes idiomas

def detectaComuns(texto):

  cont_pt = 0
```

```
 cont_en = 0
 cont_es = 0

 palavras_comuns_pt = [
      'de', 'a', 'o', 'que', 'e', 'do', 'da', 'em', 'um', 'para',
      'é', 'com', 'não', 'uma', 'os', 'no', 'se', 'na', 'por', 'mais'
  ]
 palavras_comuns_en = [ 'the', 'be', 'to', 'of', 'and', 'a', 'in', 'that', 'have', 'I',
      'it', 'for', 'not', 'on', 'with', 'he', 'as', 'you', 'do', 'at' ]
 palavras_comuns_es =  [
      'de', 'la', 'que', 'el', 'en', 'y', 'a', 'los', 'se', 'del',
      'las', 'por', 'un', 'para', 'con', 'no', 'una', 'su', 'al', 'lo'
  ]

 cont_pt = contaPalavras(texto,palavras_comuns_pt)
 cont_en = contaPalavras(texto,palavras_comuns_en)
 cont_es = contaPalavras(texto,palavras_comuns_es)

 if cont_pt > cont_en and cont_pt > cont_es:
    return 1  # Português
 elif cont_en > cont_pt and cont_en > cont_es:
    return 2  # Inglês
 elif cont_es > cont_pt and cont_es > cont_en:
    return 3  # Espanhol
 else:
    return 0  # Indeterminado



# Função: detectaDigrafos
# Dectar a frequência de dígrafos

def contaDigrafos(texto, digrafos):
   soma_total = 0
   for digrafo in digrafos:
      ocorrencias = texto.count(digrafo)
      soma_total += ocorrencias
   return soma_total
```

```
def detectaDigrafos(texto):

  cont_pt = 0
  cont_en = 0
  cont_es = 0

  digrafos_ingles = ["th", "sh", "ch", "ph", "wh", "ck"]
  digrafos_espanhol = ["ll", "ch", "rr", "qu", "gu"]
  digrafos_portugues = ["ch", "lh", "nh", "ss", "rr", "qu", "gu"]

  cont_pt = contaDigrafos(texto,digrafos_portugues)
  cont_es = contaDigrafos(texto,digrafos_espanhol)
  cont_en = contaDigrafos(texto,digrafos_ingles)

  if cont_pt > cont_en and cont_pt > cont_es:
     return 1  # Português
  elif cont_en > cont_pt and cont_en > cont_es:
     return 2  # Inglês
  elif cont_es > cont_pt and cont_es > cont_en:
     return 3  # Espanhol
  else:
     return 0  # Indeterminado


def extraiCaracteristicas(frase):
  # frase é um vetor [ 'texto', 'lingua' ]
  texto = frase[0]
  pattern_regex = re.compile('[^\w+]', re.UNICODE)
  texto = re.sub(pattern_regex,' ',texto)
  #print(texto)
  caracteristica1=tamanhoMedioFrases(texto)
  caracteristica2=detectaComuns(texto)
  caracteristica3=detectaDigrafos(texto)
  # acrescente as suas funcoes no vetor padrao
  padrao = [caracteristica1, caracteristica2, caracteristica3, frase[1] ]
  return padrao
```

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

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

dados = pd.DataFrame(padroes)
dados

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)


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))
```

# APÊNDICE 7 – APRENDIZADO DE MÁQUINA

## A – ENUNCIADO

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

## B – RESOLUÇÃO

Seed utilizado: 202475
(Ano atual com 4 dígitos + 2 algarismos do dígito verificador do CPF de um
dos integrantes)

### Especificações:

O trabalho pode ser feito por uma equipe de 1 a 6 integrantes.
Para cada problema, preencher as colunas dos quadros com o que pede.
Além disso, fazer as solicitações pedidas antes dos quadros.

### CLASSIFICAÇÃO

Para o experimento de Classificação:

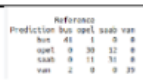
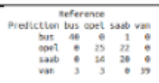
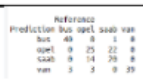
Ordenar pela Acurácia (descendente), ou seja, a técnica de melhor acurácia ficará em primeiro na tabela.

Após o quadro colocar:

Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia (criar um arquivo com novos casos à sua escolha)

A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

### VEÍCULO

| Técnica | Parâmetro | Acurácia | Matriz de Confusão |
|---|---|---|---|
| RNA – CV | size=21, decay=0.4 | 0.8443 |  |
| RF – Hold-out | mtry=10 | 0.7725 | Reference<br>Prediction bus opel saab van<br>bus 41 0 1 0<br>opel 0 27 17 0<br>saab 0 12 23 1<br>van 2 3 2 38 |
| RF – CV | mtry=10 | 0.7665 | Reference<br>Prediction bus opel saab van<br>bus 41 0 0 0<br>opel 0 26 18 0<br>saab 0 13 23 1<br>van 2 3 2 38 |
| SVM – CV | C=1, Sigma= 0.07587065 | 0.7425 |  |
| SVM – Hold-out | C=1, Sigma= 0.07587065 | 0.7425 |  |
| RNA – Hold-out | size=5, decay=0.1 | 0.6347 | Reference<br>Prediction bus opel saab van<br>bus 39 2 3 0<br>opel 2 31 36 5<br>saab 1 1 3 1<br>van 1 8 1 33 |
| KNN | k=1 | 0.6294 | Reference<br>Prediction bus opel saab van<br>bus 38 4 5 2<br>opel 2 14 20 1<br>saab 1 21 16 0<br>van 3 4 0 39 |

**Predição de novos casos:**

| | Comp | Circ | DCirc | RadRa | PrAxisRa | MaxLRa | ScatRa | Elong | PrAxisRect | MaxLRect | ScVarMaxis | ScVarmaxis | RaGyr | SkewMaxis | Skewmaxis | Kurtmaxis | KurtMaxis | HollRa | predict.rna |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 98 | 53 | 87 | 181 | 78 | 11 | 165 | 43 | 23 | 165 | 177 | 381 | 189 | 75 | 7 | 18 | 189 | 198 | van |
| 2 | 89 | 37 | 72 | 156 | 45 | 7 | 146 | 42 | 16 | 139 | 165 | 325 | 151 | 67 | 8 | 12 | 181 | 197 | saab |
| 3 | 106 | 58 | 89 | 210 | 87 | 12 | 211 | 36 | 28 | 155 | 219 | 648 | 185 | 76 | 7 | 7 | 186 | 193 | bus |

## LISTA DE COMANDOS

```
### Instalação dos pacotes necessários
#install.packages("e1071")
#install.packages("caret")
#install.packages("mlbench")
#install.packages("mice")
library(mlbench)
library("caret")

## Leitura dos dados
```

```
dados <- read.csv("6 - Veiculos - Dados.csv")


### Retira o atributo a
dados$a <- NULL
View(dados)


### Cria um arquivo com treino com 80% e teste com 20% das linhas de
forma randomizada
set.seed(202475)
indices <- createDataPartition(dados$tipo, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### indica o método cv e numero de folders 10
ctrl <- trainControl(method = "cv", number = 10)

### executa a RNA com esse ctrl
set.seed(202475)
rna  <-  train(tipo~.,  data=treino,  method="nnet",trace=FALSE,
trControl=ctrl)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))

#parametrização da RNA
### size, decay
grid <- expand.grid(size = seq(from = 1, to = 35, by = 10), decay =
seq(from = 0.1, to = 0.6, by = 0.3))
set.seed(202475)
rna <- train(
  form = tipo~. ,
  data = treino ,
  method = "nnet" ,
  tuneGrid = grid ,
  trControl = ctrl ,
  maxit = 2000,trace=FALSE)

# Exibir o modelo treinado
rna
```

```
### Faz as predições e mostra matriz de confusão
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))


### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("6 - Veiculos - Novos Dados.csv")
View(dados_novos_casos)

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

## DIABETES

| Técnica | Parâmetro | Acurácia | Matriz de Confusão |
|---|---|---|---|
| RNA – CV | size=11, decay=0.1 | 0.7647 | Reference<br>Prediction neg pos<br>neg 89 25<br>pos 11 28 |
| RF – Hold-out | mtry=2 | 0.7582 | Reference<br>Prediction neg pos<br>neg 85 22<br>pos 15 31 |
| RF – CV | mtry=5 | 0.7451 | Reference<br>Prediction neg pos<br>neg 84 23<br>pos 16 30 |
| RNA – Hold-out | size=3, decay=0.1 | 0.7451 | Reference<br>Prediction neg pos<br>neg 82 21<br>pos 18 32 |
| SVM – CV | C=0.25, Sigma= 0.1483148 | 0.732 | Reference<br>Prediction neg pos<br>neg 86 27<br>pos 14 26 |
| SVM – Hold-out | C=0.25, Sigma= 0.1483148 | 0.732 | Reference<br>Prediction neg pos<br>neg 86 27<br>pos 14 26 |
| KNN | k=9 | 0.7143 | Reference<br>Prediction neg pos<br>neg 78 32<br>pos 12 32 |

**Predição de novos casos:**

| | preg0nt | glucose | pressure | triceps | insulin | mass | pedigree | age | predict.rna |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 145 | 75 | 38 | 0 | 34 | 0.20 | 32 | neg |
| 2 | 3 | 90 | 62 | 33 | 168 | 27 | 2.29 | 21 | pos |
| 3 | 6 | 110 | 68 | 2 | 88 | 23 | 0.36 | 26 | neg |

## LISTA DE COMANDOS

```
### Instalação dos pacotes necessários
#install.packages("e1071")
#install.packages("caret")
install.packages("mlbench")
install.packages("mice")
library(mlbench)
library("caret")

## Leitura dos dados
dados <- read.csv("10 - Diabetes - Dados.csv")


### Retira o atributo num
dados$num <- NULL
View(dados)


### Cria um arquivo com treino com 80% e teste com 20% das linhas de
forma randomizada
set.seed(202475)
indices <- createDataPartition(dados$diabetes, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinar o modelo com Hold-out
set.seed(202475)
rna <- train(diabetes~., data=treino, method="nnet",trace=FALSE)
rna

### Predições dos valores do conjunto de teste
predict.rna <- predict(rna, teste)

### Matriz de confusão
confusionMatrix(predict.rna, as.factor(teste$diabetes))


########################## RNA - CV
```

```
        ### Instalação dos pacotes necessários
        #install.packages("e1071")
        #install.packages("caret")
        #install.packages("mlbench")
        #install.packages("mice")
        library(mlbench)
        library("caret")

        ## Leitura dos dados
        dados <- read.csv("10 - Diabetes - Dados.csv")

        ### Retira o atributo num
        dados$num <- NULL
        View(dados)

        ### Cria um arquivo com treino com 80% e teste com 20% das linhas de
forma randomizada
        set.seed(202475)
        indices <- createDataPartition(dados$diabetes, p=0.80,list=FALSE)
        treino <- dados[indices,]
        teste <- dados[-indices,]

        ### indica o método cv e numero de folders 10
        ctrl <- trainControl(method = "cv", number = 10)

        ### executa a RNA com esse ctrl
        set.seed(202475)
        rna <- train(diabetes~., data=treino, method="nnet",trace=FALSE,
trControl=ctrl)
        predict.rna <- predict(rna, teste)
        confusionMatrix(predict.rna, as.factor(teste$diabetes))

        #parametrização da RNA
        ### size, decay
        grid <- expand.grid(size = seq(from = 1, to = 35, by = 10), decay =
seq(from = 0.1, to = 0.6, by = 0.3))
        set.seed(202475)
        rna <- train(
          form = diabetes~. ,
          data = treino ,
          method = "nnet" ,
```

```
  tuneGrid = grid ,
  trControl = ctrl ,
  maxit = 2000,trace=FALSE)

# Exibir o modelo treinado
rna

### Faz as predições e mostra matriz de confusão
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))

### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("10 - Diabetes - Novos Dados.csv")
View(dados_novos_casos)

predict.rna <- predict(rna, dados_novos_casos)
dados_novos_casos$diabetes <- NULL
resultado <- cbind(dados_novos_casos, predict.rna)
resultado
```

## REGRESSÃO

Para o experimento de Regressão:

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

## ADMISSÃO

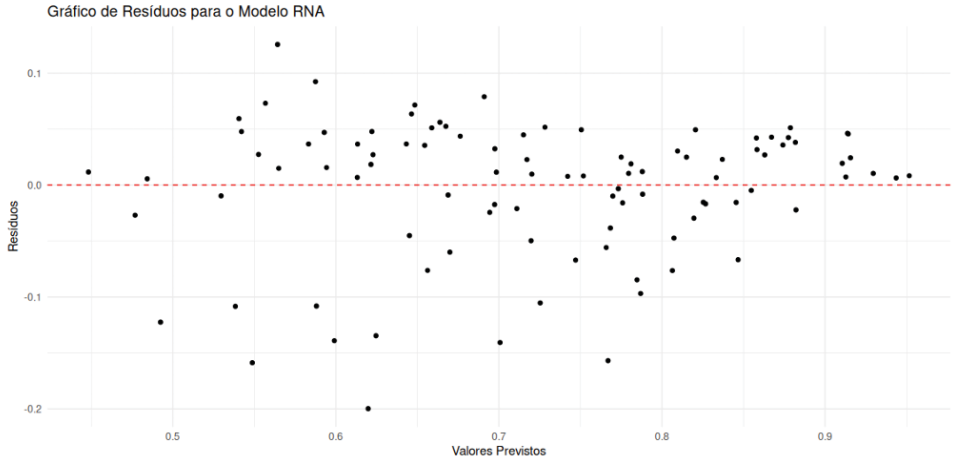
## Admissão

| Técnica | Parâmetro | R2 | Syx | Pearson | Rmse | MAE |
|---|---|---|---|---|---|---|
| RNA – CV | size=10, decay=0.1 | 0.8172428 | 0.061240715 | 0.90526694 | 0.060612589 | 0.045849687 |
| RF – Hold-out | mtry=2 | 0.8114712 | 0.062200211 | 0.90167994 | 0.061562243 | 0.042709547 |
| RF – CV | mtry=2 | 0.809526 13 | 0.062520251 | 0.90078255 | 0.061879 | 0.042564919 |
| SVM – Hold-out | C=0.5, Sigma= 0.174971 | 0.807239 05 | 0.062894481 | 0.90227164 | 0.062249393 | 0.043032732 |
| SVM – CV | C=1, Sigma= 0.1749711 | 0.806472 06 | 0.063019484 | .9012751 9 | 0.062373113 | 0.042353704 |
| RNA – Hold-out | size=5, decay=0.1 | 0.801807 08 | 0.063774502 | 0.89829229 | 0.063120388 | 0.049289792 |
| KNN | K=9 | 0.713566 58 | 0.076668127 | 0.84645202 | 0.075881766 | 0.057946712 |

**Predição de novos casos:**

```
  GRE.Score TOEFL.Score University.Rating SOP LOR CGPA Research predict.rna
1       300         105                 5 4.5 3.0  7.9        1        0.69
2       331          99                 3 4.0 2.0  8.4        0        0.61
3       305         107                 4 3.0 3.5  9.0        1        0.79
```

**Gráfico de Resíduos:**



## LISTA DE COMANDOS

```
### Pacotes necessários:
#install.packages("caret")
#install.packages("e1071")
#install.packages("mlbench")
#install.packages("mice")
#install.packages("Metrics")
library(Metrics)
library(mlbench)
library(caret)
library(mice)
library(ggplot2)

## Leitura dos dados
dados <- read.csv("9 - Admissao - Dados.csv")

### Retira o atributo num
dados$num <- NULL
View(dados)

### Cria arquivos de treino e teste
set.seed(202475)
ind <- createDataPartition(dados$ChanceOfAdmit, p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]

### CV e parametrizacao da RNA
control <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(size = seq(from = 1, to = 10, by = 1), decay
= seq(from = 0.1, to = 0.9, by = 0.3))

set.seed(202475)
rna <- train(ChanceOfAdmit~., data=treino, method="nnet",
trainControl=control, tuneGrid=tuneGrid, linout=T,
                MaxNWts=10000, maxit=2000, trace=F)
rna

### Predições e métricas
predicoes.rna <- predict(rna, teste)
rmse_valor <- rmse(teste$ChanceOfAdmit, predicoes.rna)
```

```
r2_valor <- r2(predicoes.rna, teste$ChanceOfAdmit)

# Cálculo do Erro Padrão da Estimativa (Syx)
syx <- function(observado, predito) {
  sqrt(sum((observado - predito)^2) / (length(observado) - 2))
}

syx_valor <- syx(teste$ChanceOfAdmit, predicoes.rna)

# Cálculo do Coeficiente de Correlação de Pearson
pearson_valor <- cor(teste$ChanceOfAdmit, predicoes.rna)

# Cálculo do MAE
mae <- function(observado, predito) {
  mean(abs(predito - observado))
}

mae_valor <- mae(teste$ChanceOfAdmit, predicoes.rna)

# Resultados
resultados <- list(
  R2 = format(r2_valor, digits = 8),
  Syx = format(syx_valor, digits = 8),
  Pearson = format(pearson_valor, digits = 8),
  RMSE = format(rmse_valor, digits = 8),
  MAE = format(mae_valor, digits = 8)
)

print(resultados)

# Calculo dos resíduos
residuos <- teste$ChanceOfAdmit - predicoes.rna

# Cria um data frame para o ggplot
df_residuos <- data.frame(predicoes = predicoes.rna, residuos =
residuos)

# Crie o gráfico de resíduos
ggplot(df_residuos, aes(x = predicoes, y = residuos)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
```

```
  labs(title = "Gráfico de Resíduos para o Modelo RNA",
       x = "Valores Previstos",
       y = "Resíduos") +
  theme_minimal()

### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("9 - Admissao - Novos Dados.csv")
View(dados_novos_casos)

predict.rna <- predict(rna, dados_novos_casos)
dados_novos_casos$ChanceOfAdmit <- NULL
resultado <- cbind(dados_novos_casos, predict.rna)
Resultado
```

## BIOMASSA

**Biomassa**

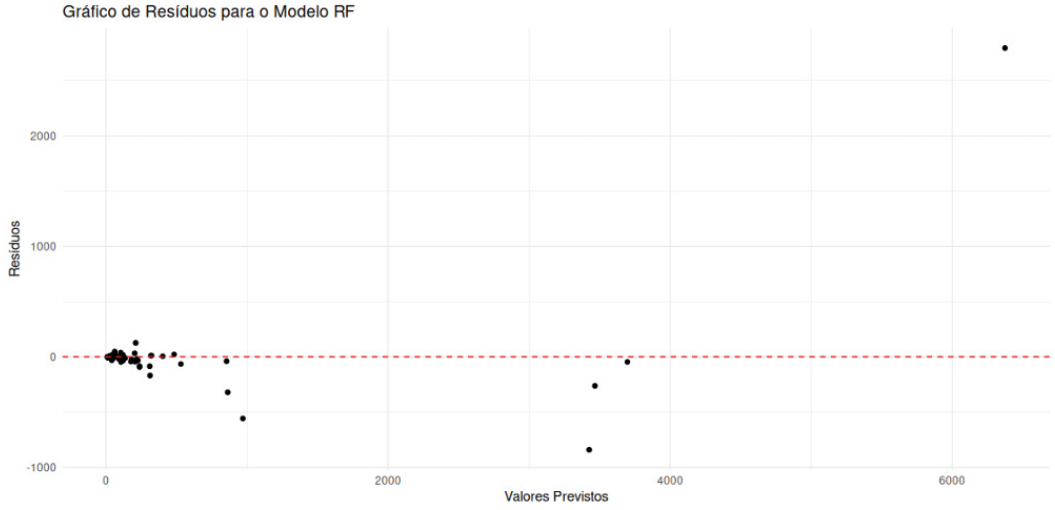
| Técnica | Parâmetro | R2 | Syx | Pearson | Rmse | MAE |
|---|---|---|---|---|---|---|
| RF – CV | mtry=2 | 0.91359863 | 396.03396 | 0.96719006 | 389.37745 | 102.82337 |
| RF – Hold-out | mtry=2 | 0.89921349 | 427.73355 | 0.96240691 | 420.54423 | 106.57287 |
| RNA – CV | size=8, decay=0.7 | 0.797106 | 606.88543 | 0.90592035 | 596.68495 | 154.11875 |
| KNN | K=9 | 0.78619982 | 622.98291 | 0.91016038 | 612.51186 | 180.985 |
| SVM – CV | C=0.25, Sigma= 4.342076 | 0.43398421 | 1013.6462 | 0.72185385 | 996.60887 | 250.34411 |
| SVM – Hold-out | C=0.25, Sigma= 4.342076 | 0.43398421 | 1013.6462 | 0.72185385 | 996.60887 | 250.34411 |

| | | | | | | |
|---|---|---|---|---|---|---|
| RNA – Hold-out | size=5, decay=0.1 | 0.014193276 | 1337.7285 | 0.8499397 | 1315.2441 | 448.10302 |

**Predição de novos casos:**

```
  dap   h   Me predict.rf
1 7.9 4.0 0.80         16
2 6.5 5.5 0.83         14
3 6.8 5.7 1.06         15
```

**Gráfico de Resíduos:**




## LISTA DE COMANDOS

```
### Pacotes necessários:
#install.packages("caret")
#install.packages("e1071")
#install.packages("kernlab")
#install.packages("Metrics")
library(Metrics)
library("caret")

## Leitura dos dados
dados <- read.csv("5 - Biomassa - Dados.csv")


### Visualizar dados
View(dados)
```

```
        ### Cria arquivos de treino e teste
        set.seed(202475)
        ind <- createDataPartition(dados$biomassa, p=0.80, list = FALSE)
        treino <- dados[ind,]
        teste <- dados[-ind,]

        ### Treinar Randon Forest com a base de Treino
        set.seed(202475)
        rf <- train(biomassa~., data=treino, method="rf")
        rf

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

        ### Calcular as métricas
        rmse_valor <- rmse(teste$biomassa, predicoes.rf)
        r2 <- function(predito, observado) {
            return(1  -  (sum((predito-observado)^2)  /  sum((observado-
mean(observado))^2)))
        }
        r2_valor <- r2(predicoes.rf,teste$biomassa)


        # Cálculo do Erro Padrão da Estimativa (Syx)
        syx <- function(observado, predito) {
          sqrt(sum((observado - predito)^2) / (length(observado) - 2))
        }

        syx_valor <- syx(teste$biomassa, predicoes.rf)

        # Cálculo do Coeficiente de Correlação de Pearson
        pearson_valor <- cor(teste$biomassa, predicoes.rf)

        # Cálculo do MAE
        mae <- function(observado, predito) {
          mean(abs(predito - observado))
        }

        mae_valor <- mae(teste$biomassa, predicoes.rf)

        # Resultados
```

```
resultados <- list(
  R2 = format(r2_valor, digits = 8),
  Syx = format(syx_valor, digits = 8),
  Pearson = format(pearson_valor, digits = 8),
  RMSE = format(rmse_valor, digits = 8),
  MAE = format(mae_valor, digits = 8)
)

resultados

# Calculo dos resíduos
residuos <- teste$biomassa - predicoes.rf

# Cria um data frame para o ggplot
df_residuos <- data.frame(predicoes = predicoes.rf, residuos =
residuos)

# Crie o gráfico de resíduos
ggplot(df_residuos, aes(x = predicoes, y = residuos)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(title = "Gráfico de Resíduos para o Modelo RF",
       x = "Valores Previstos",
       y = "Resíduos") +
  theme_minimal()

### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("5 - Biomassa - Novos Dados.csv")
View(dados_novos_casos)

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

**AGRUPAMENTO**

## VEÍCULO

Lista de Clusters gerados:

- 10 primeiras linhas do arquivo com o cluster correspondente.
- Usa 10 clusters no experimento.

```
Cluster modes:
   Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis ScVarmaxis RaGyr SkewMaxis Skewmaxis Kurtmaxis KurtMaxis HollRa tipo
1    86   37    66   126       56      7    128    52         18       127        159        246   132        85         1         6       201    183  van
2    89   45    85   150       64     10    155    45         19       144        169        354   187        72         5         9       188    196  van
3    93   38    74   176       59      7    169    39         20       131        184        259   137        67         1         3       192    202 opel
4   101   55   101   193       62     10    212    31         24       159        219        682   214        74         5        11       187    197 opel
5    89   36    66   136       58      7    144    46         19       128        164        314   127        66         0        14       181    185 saab
6   104   54   103   209       57     11    213    31         24       162        223        706   218        72         0        19       188    198 opel
7    89   40    83   197       63      9    177    37         21       139        197        290   151        71         4         1       189    198 opel
8    82   43    70   141       64      7    151    44         19       143        175        341   172        75         4         0       183    187  bus
9    85   43    70   120       54      7    150    45         19       145        170        327   171        85         6        12       180    184  bus
10   89   46    85   162       64     11    157    43         20       148        170        363   186        68         1        11       189    199 saab
```

## LISTA DE COMANDOS

```
### Instalação dos pacotes necessários
install.packages("klaR")
library(klaR)

## Leitura dos dados
dados <- read.csv("6 - Veiculos - Dados.csv")

### Retira o atributo a
dados$a <- NULL
View(dados)

### Resultados
set.seed(202475)
cluster.results <- kmodes(dados, 10, iter.max = 10, weighted = FALSE )
cluster.results
```

## REGRAS DE ASSOCIAÇÃO

## MUSCULAÇÃO

Regras geradas com uma configuração de suporte e confiança.

```
summary of quality measures:
    support         confidence        coverage            lift           count
 Min.   :0.03846   Min.   :0.7000   Min.   :0.03846   Min.   :0.8667   Min.   : 1.000
 1st Qu.:0.07692   1st Qu.:0.8571   1st Qu.:0.07692   1st Qu.:1.5294   1st Qu.: 2.000
 Median :0.07692   Median :1.0000   Median :0.07692   Median :1.7143   Median : 2.000
 Mean   :0.14665   Mean   :0.9328   Mean   :0.16799   Mean   :1.7468   Mean   : 3.813
 3rd Qu.:0.23077   3rd Qu.:1.0000   3rd Qu.:0.26923   3rd Qu.:2.0000   3rd Qu.: 6.000
 Max.   :0.46154   Max.   :1.0000   Max.   :0.65385   Max.   :3.2500   Max.   :12.000
```

## LISTA DE COMANDOS

```
### Instalação dos pacotes necessários
install.packages('arules', dep=T)
library(arules)
library(datasets)

## Leitura dos dados
dados <- read.transactions(file="2 - Musculacao - Dados.csv",format="basket",sep=";")
inspect(dados[1:4])

### Podemos ver a frequência dos 10 primeiros itens:
itemFrequencyPlot(dados, topN=10, type='absolute')

### Suporte=0,001 e Confiança=0,7
set.seed(202475)
rules <- apriori(dados, parameter = list(supp = 0.001, conf = 0.7, minlen=2))
summary(rules)
```

# 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)

```
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
# https://www.cs.toronto.edu/~kriz/cifar.html
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)

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

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

**Model: "functional"**

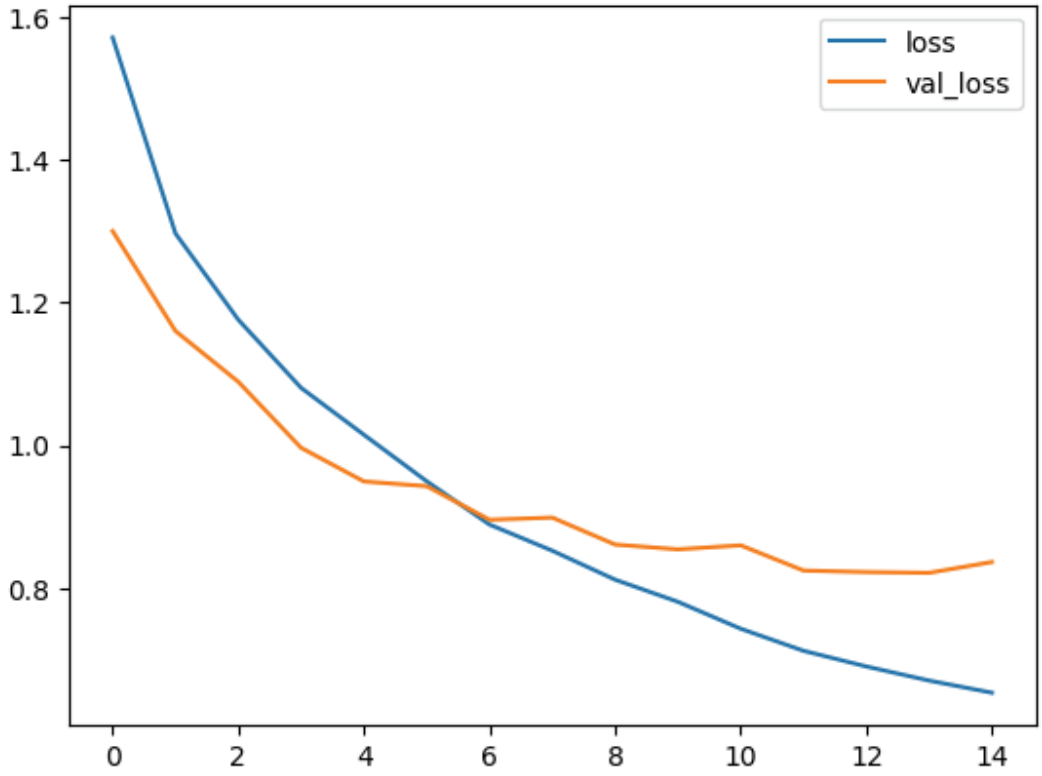
| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | (None, 32, 32, 3) | 0 |
| conv2d (Conv2D) | (None, 15, 15, 32) | 896 |
| conv2d_1 (Conv2D) | (None, 7, 7, 64) | 18,496 |
| conv2d_2 (Conv2D) | (None, 3, 3, 128) | 73,856 |
| flatten (Flatten) | (None, 1152) | 0 |
| dropout (Dropout) | (None, 1152) | 0 |
| dense (Dense) | (None, 1024) | 1,180,672 |
| dropout_1 (Dropout) | (None, 1024) | 0 |
| dense_1 (Dense) | (None, 10) | 10,250 |

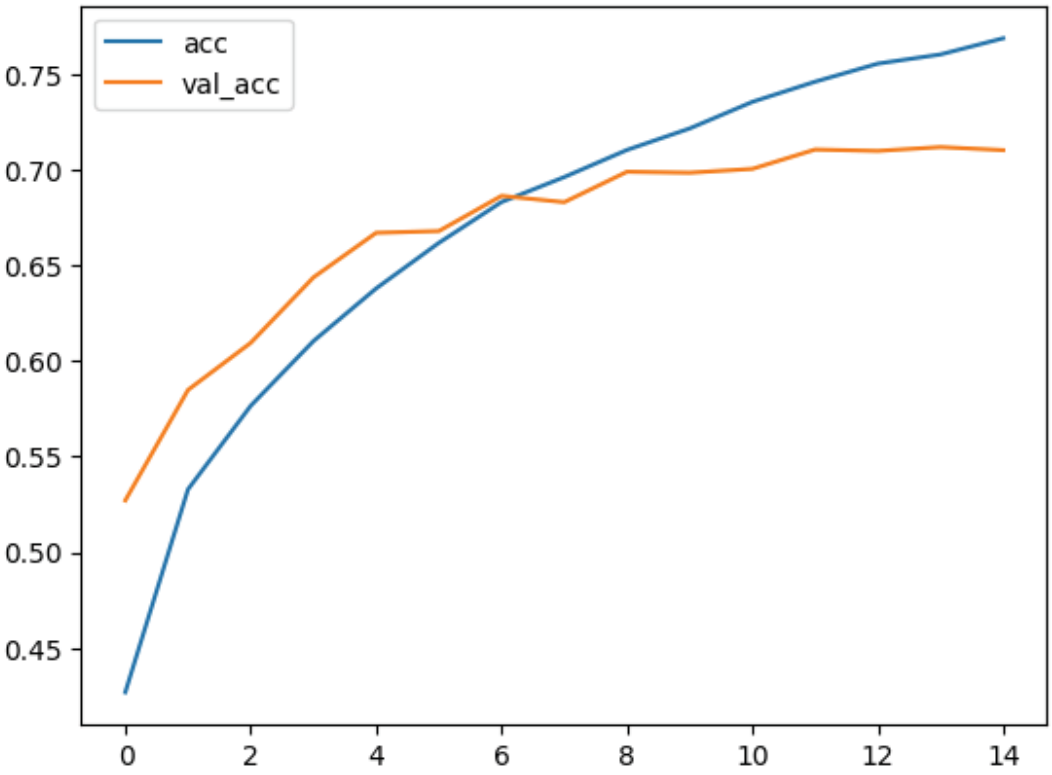
**Total params:** 1,284,170 (4.90 MB)

**Trainable params:** 1,284,170 (4.90 MB)

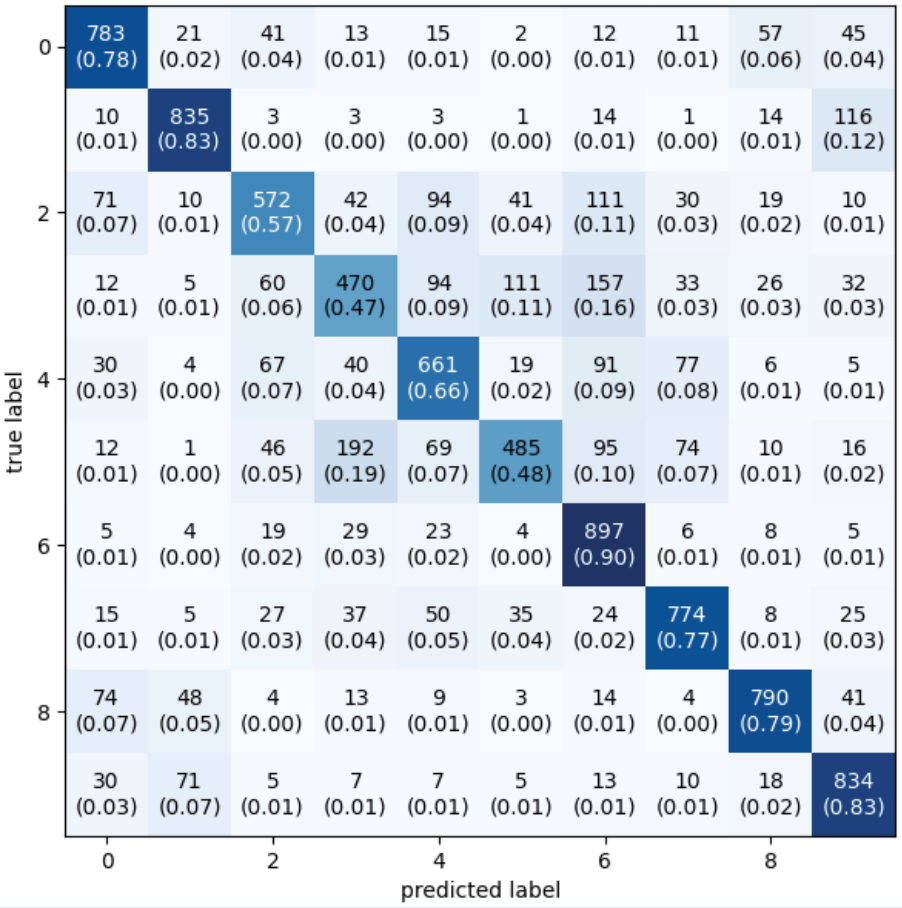
**Non-trainable params:** 0 (0.00 B)

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

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


loss
val_loss
1.6
1.4
1.2
1.0
0.8
0
2
4
6
8
10
12
14


acc
val_acc
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
2
4
6
8
10
12
14

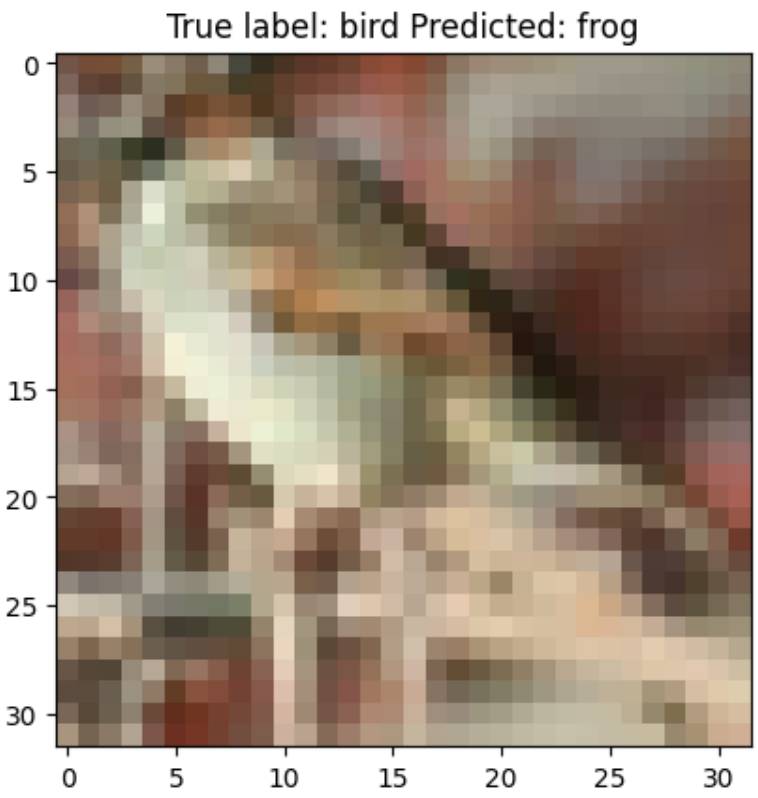
```
# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
show_normed=True)
```




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

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

```
Text(0.5, 1.0, 'True label: bird Predicted: frog')
```

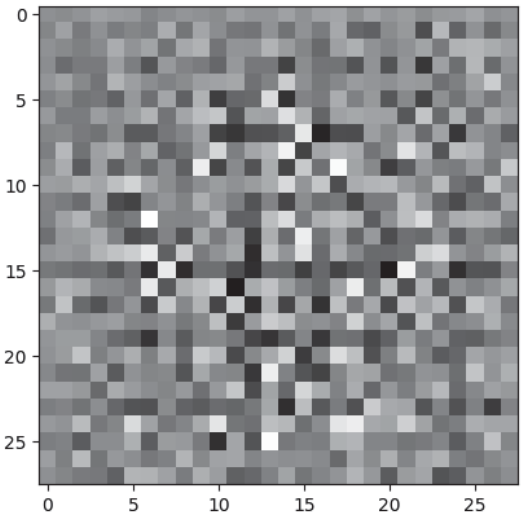



2) Gerador de Dígitos Fake (GAN)

```
# Para Gerar os GIFs
!pip install imageio
!pip install tensorflow_docs

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

```
      import time
      from IPython import display

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

      # 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)

          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')
```

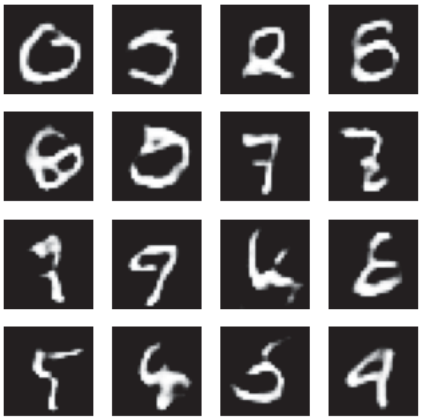



```
      # Cria o DISCRIMINADOR
      def make_discriminator_model():

        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")
c h e c k p o i n t                                                    =
tf.train.Checkpoint(generator_optimizer=generator_optimizer,discriminator_o
ptimizer=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)
```

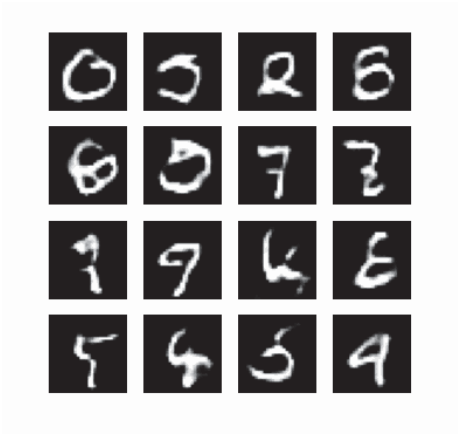
```
      seed = tf.random.normal([num_examples_to_generate, noise_dim])


      # 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))
```



```
# 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)
```



3) Detector de SPAM (RNN)

```
# Importação das Bibliotecas
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
from tensorflow.keras.layers import GlobalMaxPooling1D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer


# carrega e arruma a base
```

```
      !wget  https://filebin.net/o7trbqexn1eg7706/spam.csv  #http://
www.razer.net.br/datasets/spam.csv
      df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
      df.head()
      df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
      df.columns = ["labels", "data"]
      df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
      y = df["b_labels"].values

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

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

      # Acerta o tamanho das sequências (padding)

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

      # Define o modelo
      D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido
      M = 5 # tamanho do hidden state, quantidade de unidades LSTM
      i = Input(shape=(T,)) # Entra uma frase inteira
      x = Embedding(V+1, D)(i)
      x = LSTM(M)(x)
```

```
x = Dense(1, activation="sigmoid")(x) # Sigmoide pois só tem 2 valores
model = Model(i, x)

model.summary()
```

**Model: "functional"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | (None, 189) | 0 |
| embedding (Embedding) | (None, 189, 20) | 142,560 |
| lstm (LSTM) | (None, 5) | 520 |
| dense (Dense) | (None, 1) | 6 |

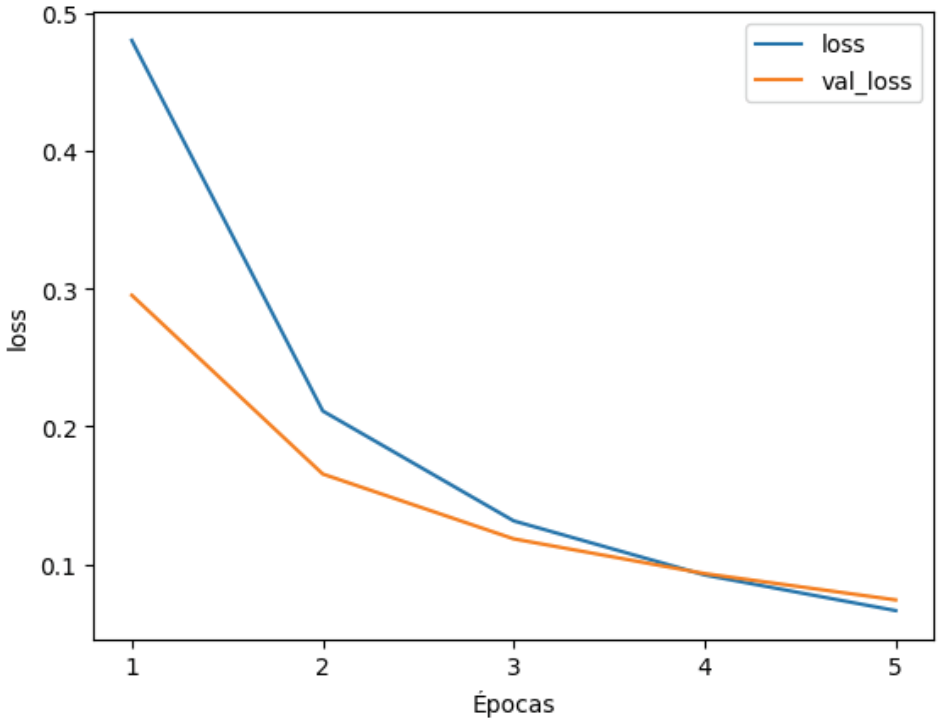
**Total params:** 143,086 (558.93 KB)

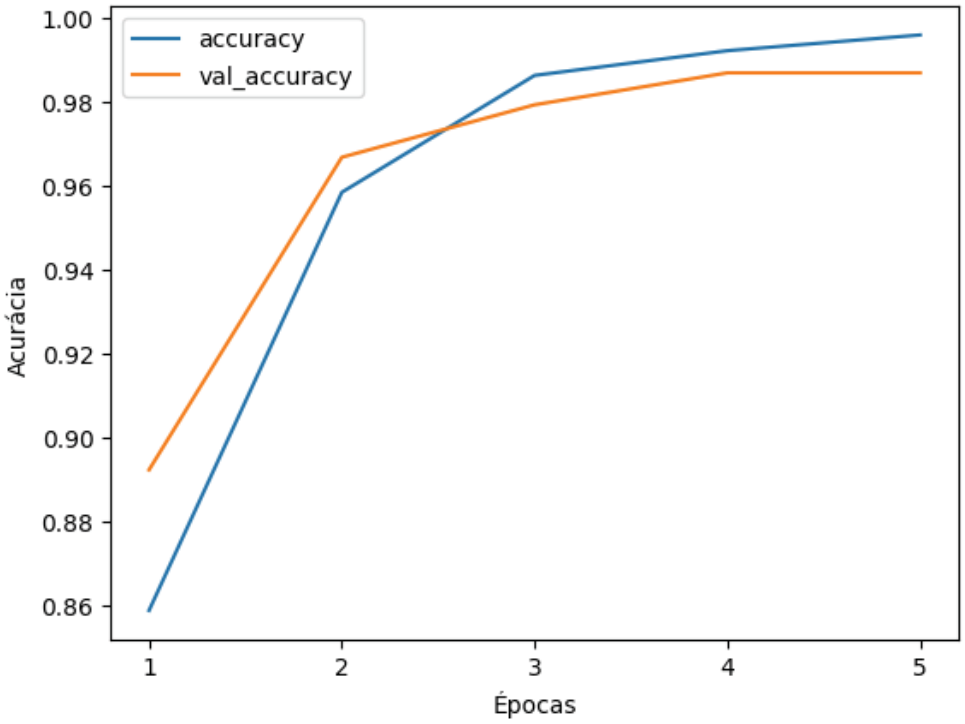
**Trainable params:** 143,086 (558.93 KB)

**Non-trainable params:** 0 (0.00 B)

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

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

```
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))

plt.legend()
plt.show()
```



```
# Efetua a predição de um texto novo
#texto = "Hi, my name is Razer and want to tell you something."
texto = "Is your car dirty? Discover our new product. Free for all.
Click the link."
seq_texto = tokenizer.texts_to_sequences([texto]) # Tokeniza
data_texto = pad_sequences(seq_texto, maxlen=T) # Padding
pred = model.predict(data_texto) # Predição
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 51ms/step
[[0.65158933]]
SPAM
```

4) Tradutor de Textos (Transformer)

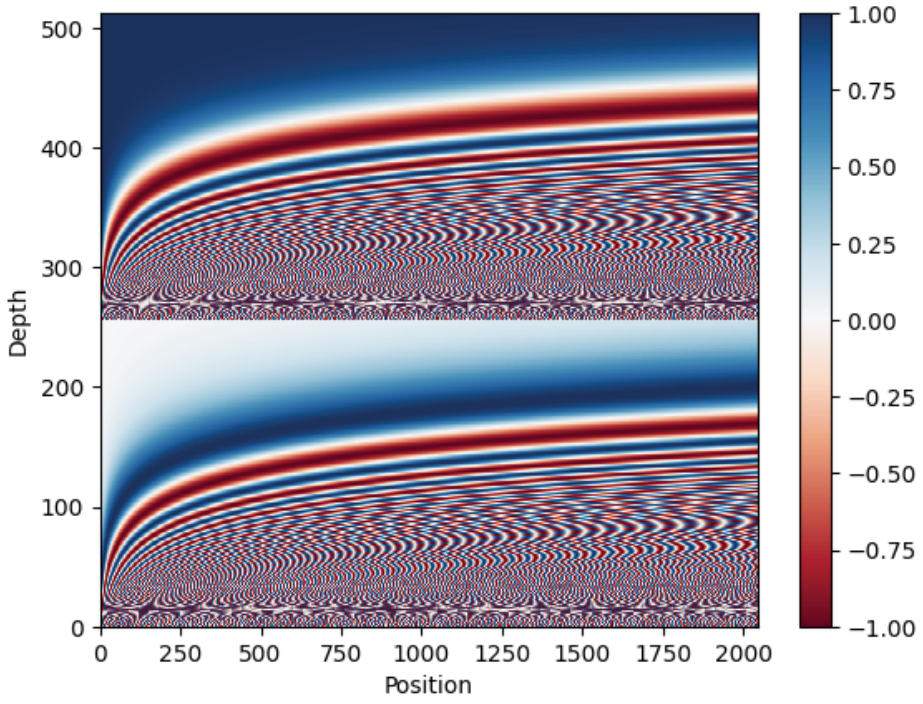
```
# Instalação e importação
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0

import collections
import logging
import os
import pathlib
import 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))
plt.pcolormesh(pos_encoding, cmap='RdBu')
plt.ylabel('Depth')
plt.xlabel('Position')

plt.colorbar()
plt.show()
```

```
(1, 2048, 512)
```




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

# Máscara futura, usada no decoder
def create_look_ahead_mask(size):
```

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

      # Função de Atenção
      def scaled_dot_product_attention(q, k, v, mask):
        # Q K^T
         matmul_qk = tf.matmul(q, k, transpose_b=True) # (..., seq_len_q,
seq_len_k)
        # converte matmul_qk para float32
        dk = tf.cast(tf.shape(k)[-1], tf.float32)
        # divide por sqrt(d_k)
        scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
         #Soma a máscara, e os valores faltantes serão um número próximo a
-inf if mask is not None:
        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)
          attn_output = self.dropout1(attn_output, training=training)
          out1 = self.layernorm1(x + attn_output)


          ffn_output = self.ffn(out1)
          ffn_output = self.dropout2(ffn_output, training=training)
          out2 = self.layernorm2(out1 + ffn_output)

          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)

          # (batch_size, input_seq_len, d_model)
          return x

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

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

          for i in range(self.num_layers):
            x, block1, block2 = self.dec_layers[i](x, enc_output, training,
                                                   look_ahead_mask,
padding_mask)
              attention_weights[f'decoder_layer{i+1}_block1'.format(i+1)] =
block1
              attention_weights[f'decoder_layer{i+1}_block2'.format(i+1)] =
block2
          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

      c l a s s
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)
      l o s s _ o b j e c t
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')

      class Translator(tf.Module):
        def __init__(self, tokenizers, transformer):
          self.tokenizers = tokenizers
          self.transformer = transformer
        def __call__(self, sentence, max_length=20):
          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
          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:, :]
            predicted_id = tf.argmax(predictions, axis=-1)
            output_array = output_array.write(i+1, predicted_id[0])
            if predicted_id == end:
              break
          output = tf.transpose(output_array.stack())
          text = tokenizers.en.detokenize(output)[0]
          tokens = tokenizers.en.lookup(output)[0]
           _, attention_weights = self.transformer([encoder_input, output[:,
:-1]], training=False)
          return text, tokens, attention_weights


      translator = Translator(tokenizers, transformer)

      sentence = "Eu li sobre triceratops na enciclopédia."

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

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

```
Prediction     : b'i read about thornal thornass in encyclood .'
```

# APÊNDICE 9 – BIG DATA

## A – ENUNCIADO

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

## B – RESOLUÇÃO

**ESTUDO DE CASO: APLICAÇÃO DE BIG DATA EM E-COMMERCE COM FERRAMENTAS NOSQL E NEWSQL**

### Introdução

No cenário competitivo do comércio eletrônico, empresas líderes como Amazon, Netflix e Uber lidam diariamente com volumes massivos de dados provenientes de transações, interações de usuários, avaliações de produtos e atividades em redes sociais. Gerenciar e extrair valor desses dados é essencial para melhorar a experiência do cliente, otimizar operações e impulsionar as vendas. Este estudo de caso explora como essas empresas utilizam ferramentas de *Big Data*, especificamente bancos de dados *NoSQL* e *NewSQL*, para atender às suas demandas de processamento e análise de dados.

### Caracterização dos Dados e dos Vs

Essas empresas enfrentam desafios relacionados aos cinco Vs do Big Data:

- Volume:
  - Amazon: Processa petabytes de dados diariamente, incluindo histórico de compras, navegação no site e interações em tempo real (AMAZON WEB SERVICES, 2018).
  - Netflix: Gerencia enormes volumes de dados de streaming e preferências de milhões de usuários globalmente (AMATRAIN, 2013).
  - Uber: Lida com dados geoespaciais e transações em tempo real de milhões de corridas diárias (UBER ENGINEERING, 2016).
- Velocidade:
  - Amazon: Necessita de atualizações em milissegundos para inventário e processamento de pedidos (AMAZON SCIENCE, 2020).
  - Uber: Calcula rotas e tarifas em tempo real, exigindo processamento instantâneo de dados (UBER ENGINEERING, 2018).

- Netflix: Oferece recomendações imediatas baseadas no comportamento atual do usuário (GOMEZ-URIBE; HUNT, 2015).

- Variedade:
  - Amazon: Dados estruturados (transações), semi-estruturados (logs de cliques) e não estruturados (comentários, avaliações) (AMAZON SCIENCE, 2019).
  - Netflix: Dados de streaming, logs de reprodução, interações do usuário e métricas de qualidade de serviço (AMATRAIN, 2013).
  - Uber: Dados geoespaciais, transações financeiras, feedback de usuários e dados de tráfego (UBER ENGINEERING, 2017).
- Veracidade:
  - Amazon: Investe em sistemas para garantir a autenticidade de avaliações e evitar fraudes (AMAZON, 2021).
  - Uber: Necessita de dados precisos para segurança e confiabilidade do serviço (UBER NEWSROOM, 2019).
  - Netflix: Garante a qualidade dos dados para oferecer recomendações precisas (GOMEZ-URIBE; HUNT, 2015).
- Valor:
  - Amazon: Utiliza análise de dados para personalizar ofertas e melhorar a satisfação do cliente (AMAZON SCIENCE, 2020).
  - Uber: Otimiza rotas e tempos de espera, melhorando a eficiência operacional (UBER ENGINEERING, 2016).
  - Netflix: A análise de dados direciona a produção de conteúdo original e retenção de usuários (AMATRAIN, 2013).

## Ferramentas Utilizadas

Para lidar com esses desafios, as empresas implementaram uma combinação de bancos de dados NoSQL e NewSQL.

## Bancos de Dados NoSQL

- Amazon DynamoDB (Amazon)
  - Uso: A Amazon utiliza o DynamoDB para gerenciar dados de produtos, carrinhos de compras e sessões de usuários (AMAZON WEB SERVICES, 2023).
  - Benefícios: Oferece latência de milissegundos em qualquer escala, permitindo que a Amazon mantenha alta performance durante picos de acesso, como na Black Friday.
  - Modelagem: Adota um modelo de chave-valor e documento, proporcionando flexibilidade na estrutura dos dados.
- Apache Cassandra (Netflix)
  - Uso: A Netflix emprega o Cassandra para armazenar dados de streaming, preferências do usuário e histórico de visualização (NETFLIX TECH BLOG, 2011).

- Benefícios: Fornece alta disponibilidade e escalabilidade horizontal, essencial para o serviço global da Netflix.
- Modelagem: Utiliza um modelo de colunas largas, ideal para grandes volumes de dados e acessos rápidos.

## Bancos de Dados NewSQL

- Google Spanner (Google)
  - Uso: Embora não seja uma empresa de e-commerce tradicional, o Google utiliza o Spanner para serviços que requerem consistência global, como transações financeiras no Google Ads (CORBETT et al., 2013).
  - Benefícios: Combina a escalabilidade de sistemas NoSQL com transações ACID, garantindo consistência em escala global.
  - Modelagem: Mantém um esquema relacional tradicional, facilitando consultas complexas com SQL padrão.
- MemSQL (SingleStore) (Uber)
  - Uso: A Uber utiliza o MemSQL para processar transações financeiras e dados de telemetria em tempo real (UBER ENGINEERING, 2017).
  - Benefícios: Oferece processamento rápido de transações e análises, suportando decisões em tempo real.
  - Modelagem: Combina armazenamento em memória e disco, permitindo consultas rápidas sem sacrificar a persistência dos dados.

## Modelagem Necessária

A escolha do modelo de dados adequado é fundamental para otimizar o desempenho e a eficiência dos sistemas

## Modelagem em *NoSQL*

- Modelo de Documento (DynamoDB na Amazon):
  - Estrutura: Armazena dados em formatos flexíveis como JSON.
  - Aplicação: Permite à Amazon adicionar novos atributos aos produtos sem migrações complexas.
  - Benefícios: Escalabilidade e flexibilidade para acomodar diferentes tipos de produtos e categorias.
- Modelo de Colunas Largas (Cassandra na Netflix):
  - Estrutura: Organiza dados em tabelas com linhas e colunas, mas com a capacidade de armazenar milhões de colunas.
  - Aplicação: Ideal para o histórico de visualização de usuários e logs de atividades.

- Benefícios: Otimiza operações de gravação e leitura em grandes volumes de dados distribuídos.

## Modelagem em *NewSQL*

- Modelo Relacional Distribuído (Spanner no Google):
  - Estrutura: Mantém tabelas relacionais com suporte a transações ACID.
  - Aplicação: Gerencia dados que requerem consistência forte e transações distribuídas.
  - Benefícios: Combina a familiaridade do SQL com a escalabilidade necessária para operações globais.
- Modelo Híbrido (MemSQL na Uber):
  - Estrutura: Une aspectos de bancos de dados relacionais e NoSQL, suportando SQL e armazenamento em memória.
  - Aplicação: Processa dados de corridas, pagamentos e localização em tempo real.
  - Benefícios: Permite análises rápidas e decisões imediatas sem comprometer a integridade dos dados.

## Conclusão

A adoção de bancos de dados NoSQL e NewSQL permitiu que empresas como Amazon, Netflix e Uber superassem os desafios associados ao Big Data, proporcionando:

- Melhor Desempenho: Processamento eficiente de grandes volumes de dados em tempo real, melhorando a responsividade dos serviços e a satisfação do cliente.
- Flexibilidade: Capacidade de adaptar-se rapidamente a novas fontes de dados e requisitos de negócios sem reestruturações significativas do banco de dados.
- Confiabilidade: Garantia de integridade e consistência dos dados críticos, como transações financeiras e informações de inventário.

Os desafios enfrentados incluíram a complexidade na integração de diferentes sistemas de banco de dados e a necessidade de profissionais qualificados em tecnologias diversas. Essas empresas superaram esses obstáculos através de arquiteturas bem planejadas, investimentos em infraestrutura e foco no desenvolvimento de equipes especializadas.

Este estudo de caso destaca a importância de selecionar as ferramentas e modelos de dados apropriados para atender às necessidades específicas do negócio. Empresas líderes como Amazon, Netflix e Uber demonstram que o uso estratégico de tecnologias *NoSQL* e *NewSQL* é crucial para maximizar o valor extraído dos dados e manter a competitividade no mercado global de e-commerce.

## Referências

AMATRIAIN, X. Big & Personal: Data and Models Behind Netflix Recommendations. Netflix Tech Blog, 2013. Disponível em: https://netflixtechblog.com/.

AMAZON. Combating Fake Reviews and Ratings. Amazon Official Site, 2021. Disponível em: https://www.amazon.com.

AMAZON SCIENCE. How AI is Enhancing the Customer Experience at Amazon. Amazon Science, 2020. Disponível em: https://www.amazon.science/.

AMAZON SCIENCE. Personalization and Recommendation. Amazon Science, 2019. Disponível em: https://www.amazon.science/.

AMAZON WEB SERVICES. DynamoDB: Under the Hood. AWS re:Invent, 2018. Disponível em: https://aws.amazon.com/dynamodb/.

AMAZON WEB SERVICES. Amazon DynamoDB—NoSQL Database Service. AWS, 2023. Disponível em: https://aws.amazon.com/dynamodb/.

CORBETT, J. C. et al. Spanner: Google's Globally Distributed Database. ACM Transactions on Computer Systems, vol. 31, no. 3, 2013.

GOMEZ-URIBE, C. A.; HUNT, N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. ACM Transactions on Management Information Systems, vol. 6, no. 4, 2015.

NETFLIX TECH BLOG. Cassandra at Netflix. Medium, 2011. Disponível em: https://netflixtechblog.com/.

UBER ENGINEERING. Geo-spatial Indexing with Uber H3. Uber Engineering Blog, 2018. Disponível em: https://eng.uber.com/h3/.

UBER ENGINEERING. How Uber Engineering Uses Data to Improve the User Experience. Uber Engineering Blog, 2016. Disponível em: https://eng.uber.com/.

UBER ENGINEERING. Real-time Streaming Data Processing at Uber Using Apache Flink. Uber Engineering Blog, 2017. Disponível em: https://eng.uber.com/flink/.

UBER ENGINEERING. Optimizing Uber's Map Services. Uber Engineering Blog, 2018. Disponível em: https://eng.uber.com/maps/.

UBER NEWSROOM. Safety at Uber: Technology and Tools. Uber Newsroom, 2019. Disponível em: https://www.uber.com/newsroom/.

# 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

## Extração de Características

O Random Forest demonstrou ser o modelo mais eficaz, sobretudo por ter apresentado os melhores índices de sensibilidade e especificidade entre os três avaliados. Isso reflete sua capacidade superior tanto em identificar corretamente os casos positivos quanto em minimizar os falsos positivos. Apesar de a RNA ter alcançado o melhor F1-Score, o Random Forest se destacou por proporcionar um desempenho mais uniforme e consistente nas três métricas consideradas.

## Redes Neurais

O modelo VGG16, treinado sem a utilização de técnicas de Data Augmentation, mostrou-se extremamente eficiente conforme demonstrado pela matriz de confusão. Esta matriz evidenciou um total de 369 previsões corretas de um universo de 371 possíveis.

A elevada especificidade do modelo indica uma eficácia notável em evitar falsos positivos. Por outro lado, a sensibilidade de 0.4973 sugere que o modelo foi capaz de reconhecer uma porção considerável dos exemplos verdadeiramente positivos.

Os resultados obtidos com o ResNet50 apontam para uma dificuldade do modelo em aprender a classificação. Diante disso, algumas estratégias poderiam ser adotadas para aprimorar o desempenho do modelo, tais como:

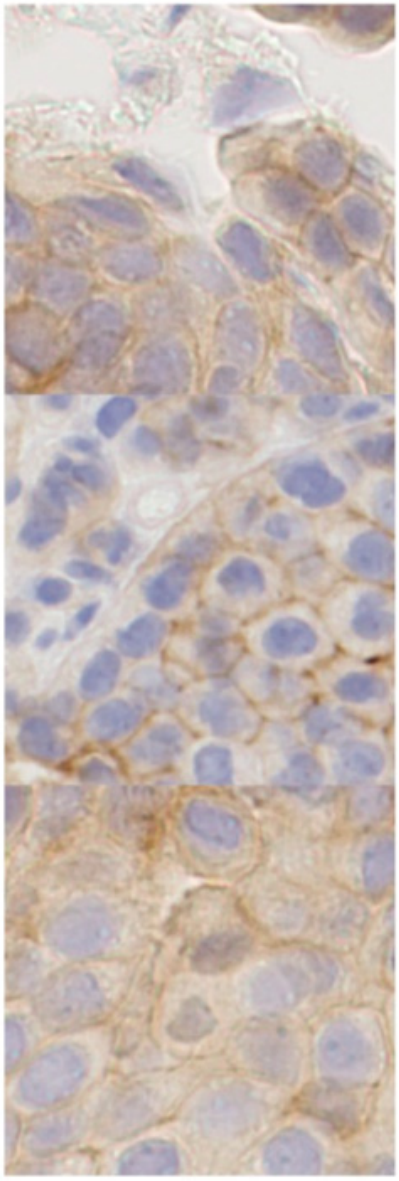
- Descongelar um número maior de camadas para permitir uma aprendizagem mais flexível;
- Incrementar o número de épocas para que o modelo tenha mais oportunidades de aprender com os dados;
- Realizar o balanceamento do conjunto de dados, a fim de evitar um viés do modelo em favor das classes mais frequentes;
- Garantir a qualidade dos dados, verificando se estão adequadamente formatados e rotulados, para assegurar a eficácia do treinamento do modelo.

# LINHAS DE CÓDIGO

```
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report
from keras.applications.vgg16 import VGG16
```

```
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from skimage.feature import local_binary_pattern
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import f1_score, recall_score, precision_score
from sklearn import metrics
import zipfile
from keras.preprocessing.image import img_to_array, load_img
import numpy as np
from sklearn.model_selection import train_test_split
import cv2
from google.colab.patches import cv2_imshow


#Preparação da base
train_zip_path = '/content/Train_Warwick.zip'
test_zip_path = '/content/Test_Warwick.zip'

train_extract_dir = '/content/Train_Warwick'
test_extract_dir = '/content/Test_Warwick'

def unzip_file(zip_path, extract_dir):
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)

unzip_file(train_zip_path, train_extract_dir)
unzip_file(test_zip_path, test_extract_dir)

train_dir_contents = os.listdir(train_extract_dir)
test_dir_contents = os.listdir(test_extract_dir)

train_image_dir = os.path.join(train_extract_dir,
'Train_4cls_amostra')
test_image_dir = os.path.join(test_extract_dir, 'Test_4cl_amostra')

train_files = os.listdir(train_image_dir)
test_files = os.listdir(test_image_dir)

train_dir_contents, test_dir_contents

# Verificar a estrutura dentro dos diretórios de treino e teste
train_image_dir = os.path.join(train_extract_dir,
'Train_4cls_amostra')
test_image_dir = os.path.join(test_extract_dir, 'Test_4cl_amostra')

train_files = os.listdir(train_image_dir)
test_files = os.listdir(test_image_dir)

# Exibir os primeiros arquivos encontrados
train_files, test_files

# Função para carregar as imagens usando OpenCV
def load_images_opencv(base_dir):
    images = []
    labels = []
    for label_dir in os.listdir(base_dir):
        label_path = os.path.join(base_dir, label_dir)
        if os.path.isdir(label_path):  # Verificar se é um diretório
de classe
            for img_file in os.listdir(label_path):
                if img_file.endswith(‘.png'):
```

```
                        img_path = os.path.join(label_path, img_file)
                        img = cv2.imread(img_path)
                        img = cv2.resize(img, (250, 250))
                        images.append(img)
                        labels.append(label_dir)  # O nome do diretório é
o rótulo
          return np.array(images), np.array(labels)

      # Carregar as imagens e rótulos do conjunto de treino e teste
      train_images, train_labels = load_images_opencv(train_image_dir)
      test_images, test_labels = load_images_opencv(test_image_dir)

      # Exibir o tamanho das bases carregadas
      train_images.shape, len(train_labels), test_images.shape,
len(test_labels)

      #Exibir amostra
      cv2_imshow(train_images[1])
      cv2_imshow(train_images[2])
      cv2_imshow(train_images[3])
```



```
      # Garantir que a separação seja feita por paciente
      unique_patients = np.unique(train_labels)

      # Separar 80% para treino e 20% para validação, garantindo a
separação por paciente
      train_patients, val_patients = train_test_split(unique_patients,
test_size=0.2, random_state=42)

      # Criar índices de treino e validação com base nos pacientes
      train_idx = [i for i, label in enumerate(train_labels) if label in
train_patients]
      val_idx = [i for i, label in enumerate(train_labels) if label in
val_patients]

      # Criar os conjuntos de treino e validação
      X_train, y_train = train_images[train_idx], train_labels[train_idx]
      X_val, y_val = train_images[val_idx], train_labels[val_idx]

      # Exibir o tamanho dos conjuntos de treino e validação
      X_train.shape, len(y_train), X_val.shape, len(y_val)

      def extract_lbp_features(images):
          lbp_features = []
          for img in images:
              gray_img = cv2.cvtColor(img.astype('uint8'),
cv2.COLOR_BGR2GRAY)
              lbp = local_binary_pattern(gray_img, 24, 3, method="uniform")
              hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, 27),
range=(0, 26))
              hist = hist.astype('float')
              hist /= (hist.sum() + 1e-6)  # Normalização
              lbp_features.append(hist)
          return np.array(lbp_features)

      def extract_vgg16_features(images):
          vgg_model = VGG16(weights='imagenet', include_top=False)
          features = vgg_model.predict(images)
          features = features.reshape(features.shape[0], -1)  # Flatten
          return features


      # Extração para treino
      lbp_train_features = extract_lbp_features(X_train)
      vgg_train_features = extract_vgg16_features(X_train)

      # Extração para validação
      lbp_val_features = extract_lbp_features(X_val)
      vgg_val_features = extract_vgg16_features(X_val)

      # Salvar como CSV
      pd.DataFrame(lbp_train_features).to_csv('lbp_train_features.csv')
      pd.DataFrame(vgg_train_features).to_csv('vgg_train_features.csv')

      def avg_specificity(confusion_matrix):
          num_classes = confusion_matrix.shape[0]
          specificities = []

          for i in range(num_classes):
              # Verdadeiros Negativos (VN) são todos os valores fora da
linha e coluna da classe i
```

```
            true_negatives = np.sum(confusion_matrix) -
(np.sum(confusion_matrix[i, :]) + np.sum(confusion_matrix[:, i]) -
confusion_matrix[i, i])
            # Falsos Positivos (FP) são a soma dos valores na coluna da
classe i, exceto a diagonal (falsos positivos)
            false_positives = np.sum(confusion_matrix[:, i]) -
confusion_matrix[i, i]

            # Evitar divisão por zero
            if (true_negatives + false_positives) > 0:
                specificity = true_negatives / (true_negatives +
false_positives)
            else:
                specificity = 0  # Definir como 0 quando o denominador
for 0
            specificities.append(specificity)

        # Especificidade média
        specificity_mean = np.mean(specificities)
        return specificity_mean

    def train_and_evaluate(model, X_train, y_train, X_val, y_val):
        model.fit(X_train, y_train)
        y_pred = model.predict(X_val)
        cm = confusion_matrix(y_val, y_pred)
        sensitivity = recall_score(y_val, y_pred, average='macro')
        specificity = avg_specificity(cm)
        f1 = f1_score(y_val, y_pred, average='macro')
        return sensitivity, specificity, f1

    # Modelos
    rf_model = RandomForestClassifier()
    svm_model = SVC()
    rna_model = MLPClassifier()

    train_and_evaluate(rf_model, vgg_train_features, y_train,
vgg_val_features, y_val)
    train_and_evaluate(svm_model, vgg_train_features, y_train,
vgg_val_features, y_val)
    train_and_evaluate(rna_model, vgg_train_features, y_train,
vgg_val_features, y_val)

    lbp_test_features = extract_lbp_features(test_images)
    vgg_test_features = extract_vgg16_features(test_images)

    rf_test_sens, rf_test_spec, rf_test_f1 = train_and_evaluate(rf_model,
vgg_train_features, y_train, vgg_test_features, test_labels)
    svm_test_sens, svm_test_spec, svm_test_f1 =
train_and_evaluate(svm_model, vgg_train_features, y_train,
vgg_test_features, test_labels)
    rna_test_sens, rna_test_spec, rna_test_f1 =
train_and_evaluate(rna_model, vgg_train_features, y_train,
vgg_test_features, test_labels)

    print(f"Random Forest - Teste Sensibilidade: {rf_test_sens},
Especificidade: {rf_test_spec}, F1-Score: {rf_test_f1}")
    print(f"SVM - Teste Sensibilidade: {svm_test_sens}, Especificidade:
{svm_test_spec}, F1-Score: {svm_test_f1}")
    print(f"RNA - Teste Sensibilidade: {rna_test_sens}, Especificidade:
{rna_test_spec}, F1-Score: {rna_test_f1}")
```

```
      models = {'Random Forest': rf_test_f1, 'SVM': svm_test_f1, 'RNA':
rna_test_f1}
      best_model = max(models, key=models.get)
      print(f"O melhor modelo foi {best_model} com F1-Score de
{models[best_model]}")

      from keras.applications.vgg16 import VGG16
      from keras.applications.resnet import ResNet50
      from keras.layers import Dense, Flatten, GlobalAveragePooling2D
      from keras.models import Model
      from keras.optimizers import Adam
      from tensorflow.keras.preprocessing.image import ImageDataGenerator

      import warnings
      warnings.filterwarnings('ignore')

      def resize_images(images):
          resized_images = []
          for img in images:
              resized_img = cv2.resize(img, (224, 224))  # Redimensionar
para 224x224
              if img.shape[-1] == 1:  # Se a imagem tiver um único canal
(em tons de cinza), converte para 3 canais
                  resized_img = cv2.cvtColor(resized_img,
cv2.COLOR_GRAY2RGB)
              resized_images.append(resized_img)
          return np.array(resized_images)

      # Carregar as imagens e rótulos do conjunto de treino e validação
      X_train_resized = resize_images(X_train)  # De 250x250 para 224x224
      X_val_resized = resize_images(X_val)
      X_test_resized = resize_images(test_images)

      # Verificar as formas das imagens redimensionadas
      print("Forma das imagens de treino:", X_train_resized.shape)
      print("Forma das imagens de validação:", X_val_resized.shape)
      print("Forma das imagens de teste:", X_test_resized.shape)

      # Verificar os rótulos e garantir que há 4 classes
      print(np.unique(y_train))

      # One-hot encode dos rótulos com 4 classes (0, 1, 2, 3)
      from sklearn.preprocessing import LabelBinarizer

      lb = LabelBinarizer()
      lb.fit([0, 1, 2, 3])  # Garantindo que 4 classes são configuradas
corretamente
      y_train_bin = lb.transform(y_train)
      y_val_bin = lb.transform(y_val)
      y_test_bin = lb.transform(test_labels)

      # Verificar a forma dos rótulos binarizados
      print("Forma dos rótulos de treino:", y_train_bin.shape)

      # Função para criar o modelo baseado na VGG16 com camadas Fully
Connected ajustadas
      def create_vgg16_model():
          base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
          x = base_model.output
          x = Flatten()(x)  # Usando a camada Flatten para achatar as
saídas
```

```
          x = Dense(128, activation='relu')(x)
          predictions = Dense(4, activation='softmax')(x)  # Ajustar para 4
classes
          model = Model(inputs=base_model.input, outputs=predictions)

          # Congelar as camadas da base da VGG16
          for layer in base_model.layers:
              layer.trainable = False

          model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
          return model

      # Função para criar o modelo baseado na ResNet50 com camadas Fully
Connected ajustadas
      def create_resnet50_model():
          base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
          x = base_model.output
          x = GlobalAveragePooling2D()(x)  # Usando GlobalAveragePooling2D
em vez de Flatten
          x = Dense(128, activation='relu')(x)
          predictions = Dense(4, activation='softmax')(x)  # Ajustar para 4
classes
          model = Model(inputs=base_model.input, outputs=predictions)

          # Congelar as camadas da base da ResNet50
          for layer in base_model.layers:
              layer.trainable = False

          model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])
          return model

      # Função para treinar o modelo com ou sem Data Augmentation
      def train_model(model, X_train, y_train, X_val, y_val,
augment=False):
          if augment:
              datagen = ImageDataGenerator(
                  rotation_range=20,
                  width_shift_range=0.2,
                  height_shift_range=0.2,
                  horizontal_flip=True

              )
              datagen.fit(X_train)
              history = model.fit(datagen.flow(X_train, y_train,
batch_size=32),
                                  validation_data=(X_val, y_val),
                                  epochs=10)
          else:
              history = model.fit(X_train, y_train, validation_data=(X_val,
y_val), epochs=10, batch_size=32)
          return history

      # Criar os modelos
      vgg16_model = create_vgg16_model()
      resnet50_model = create_resnet50_model()

      # Treinar o modelo VGG16 sem Data Augmentation
      print("Treinando VGG16 sem Data Augmentation...")
```

```
      train_model(vgg16_model, X_train_resized, y_train_bin, X_val_resized,
y_val_bin, augment=False)

      # Treinar o modelo VGG16 com Data Augmentation
      print("Treinando VGG16 com Data Augmentation...")
      train_model(vgg16_model, X_train_resized, y_train_bin, X_val_resized,
y_val_bin, augment=True)

      # Treinar o modelo ResNet50 sem Data Augmentation
      print("Treinando ResNet50 sem Data Augmentation...")
      train_model(resnet50_model, X_train_resized, y_train_bin,
X_val_resized, y_val_bin, augment=False)

      # Treinar o modelo ResNet50 com Data Augmentation
      print("Treinando ResNet50 com Data Augmentation...")
      train_model(resnet50_model, X_train_resized, y_train_bin,
X_val_resized, y_val_bin, augment=True)

      def evaluate_model(model, X_test, y_test):
          y_pred = np.argmax(model.predict(X_test), axis=1)
          y_true = np.argmax(y_test, axis=1)

          # Identificar as classes presentes
          labels_present = np.unique(y_true)

          # Matriz de confusão
          cm = confusion_matrix(y_true, y_pred, labels=labels_present)
          print("Matriz de Confusão:")
          print(cm)

          # Relatório de classificação apenas para as classes presentes
          report = classification_report(y_true, y_pred,
labels=labels_present, target_names=[str(l) for l in labels_present],
zero_division=1)
          print(report)

          # Cálculo de Sensibilidade, Especificidade e F1-Score
          sensitivity = recall_score(y_true, y_pred, average='macro')
          specificity = np.mean(np.diag(cm) / np.sum(cm, axis=1))
          f1 = f1_score(y_true, y_pred, average='macro')

          return sensitivity, specificity, f1

      # Avaliar os modelos na base de teste

      print("Avaliando VGG16 sem Data Augmentation...")
      vgg16_sens, vgg16_spec, vgg16_f1 = evaluate_model(vgg16_model,
X_test_resized, y_test_bin)

      print("Avaliando ResNet50 sem Data Augmentation...")
      resnet50_sens, resnet50_spec, resnet50_f1 =
evaluate_model(resnet50_model, X_test_resized, y_test_bin)

      # Comparar os resultados
      print(f"VGG16 - Sensibilidade: {vgg16_sens}, Especificidade:
{vgg16_spec}, F1-Score: {vgg16_f1}")
      print(f"ResNet50 - Sensibilidade: {resnet50_sens}, Especificidade:
{resnet50_spec}, F1-Score: {resnet50_f1}")

      # Comparar os modelos e exibir o melhor resultado
      models = {'VGG16': vgg16_f1, 'ResNet50': resnet50_f1}
      best_model = max(models, key=models.get)
```

```
    print(f"O melhor modelo foi {best_model} com F1-Score de
{models[best_model]}")
```

```
Avaliando VGG16 sem Data Augmentation...
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 183ms/step
Matriz de Confusão:
[[369]]
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       371

   micro avg       1.00      0.99      1.00       371
   macro avg       1.00      0.99      1.00       371
weighted avg       1.00      0.99      1.00       371

Avaliando ResNet50 sem Data Augmentation...
12/12 ━━━━━━━━━━━━━━━━━━━━ 10s 348ms/step
Matriz de Confusão:
[[0]]
              precision    recall  f1-score   support

           0       1.00      0.00      0.00     371.0

   micro avg       1.00      0.00      0.00     371.0
   macro avg       1.00      0.00      0.00     371.0
weighted avg       1.00      0.00      0.00     371.0

VGG16 - Sensibilidade: 0.4973045822102426, Especificidade: 1.0, F1-Score: 0.49864864864864866
ResNet50 - Sensibilidade: 0.0, Especificidade: nan, F1-Score: 0.0
O melhor modelo foi VGG16 com F1-Score de 0.49864864864864866
```

# 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

## 1 INTRODUÇÃO

O serviço de chat, com respostas geradas por uma inteligência artificial, da Open AI, cresceu sua base de usuários de forma assustadoramente rápida. O Chat GPT (acrônimo de Chat Generative Pre-trained Transformer) foi liberado ao público em 2022 - sendo sua primeira versão privada criada em 2018 - e atingiu a marca de 1 milhão de usuários em apenas 5 dias (BUCHHOLZ, 2023). Em dois meses atingiu a marca de 100 milhões de usuários (HU, 2023). Tornando-se o serviço digital com a adesão mais rápida da história. Tamanha evolução causa riscos e preocupações éticas que não tínhamos até então nesta particular área do desenvolvimento de software. Urge a discussão dos efeitos causados na segurança, trabalho humano e atenção à discriminação e desigualdades sociais para que este desenvolvimento preserve os direitos fundamentais. E que seja assegurado desenvolvimento tecnológico seguro em benefício da pessoa humana.

No Brasil, tramita no Senado Federal a PL 2.338/2023 (BRASIL, 2023), texto inicial do Senador e presidente da casa Rodrigo Pacheco (PSD-MG), com a participação de uma comissão. Tal projeto de lei é considerado a primeira base para o Marco Legal da Inteligência Artificial no Brasil. Mas avança lentamente e as discussões parecem não ter eco nas empresas como fora o Marco Civil da Internet. Embora tenham realizado quatro audiências públicas, um seminário internacional, doze painéis temáticos e consulta a mais de 60 especialistas (BRASIL, 2023).

## 2 REVISÃO DE LITERATURA

## 1.1.RISCO DO USO EXCESSIVO

A motivação do uso massivo do Chat GPT, um LLM (Large Language Model) ou modelo de aprendizagem de máquina com bilhões de parâmetros, é bem clara. Conseguimos respostas rápidas para qualquer assunto, somos capazes de usá-lo para criar textos com estilos diversos, criar código de programação, traduzir documentos em idiomas diferentes, aprender sobre novos temas, obter insights de como resolver problemas, dos mais complexos aos mais frugais, como por resolver uma equação integral esférica para encontrar o volume de uma esfera imperfeita ou como trocar uma lâmpada de tungstênio.

Mas não se engane, como toda ferramenta, existem imperfeições, neste caso o termo usualmente utilizado para descrever estas imperfeições é alucinações. É quando o modelo de linguagem não consegue inferir uma resposta correta e acaba produzindo um texto falso, seja por vieses, poucos dados de treinamento ou overfitting, dos algoritmos. Emsley (2023) defende que não são alucinações e sim invenções e falsificações. Ele descreve a imperfeição como:

> *"A causa destas falsificações tem sido associada a uma perturbação na produção linguística, com resultados probabilísticos baseados em estimativas de similaridade semântica" - (EMSLEY,2023)*

Emsley (2023) cita que em uma investigação de acurácia e autenticidade de artigos médicos, o ChatGPT citou 115 referências. Destas 47% foram fabricadas, 46% são autênticas porém imprecisas e apenas 7% foi identificada como autêntica e precisa.

Em que nível está esta alucinação? Só recentemente a OpenAI adicionou o alerta que passa desapercebido pela maioria das pessoas: "ChatGPT pode cometer erros. Considere verificar informações importantes.". Mas não há monitoramento de sua qualidade. Fica evidente a necessidade de supervisão humana qualificada. E mais, conscientização de seu uso, dado que a confiabilidade na assertividade está ainda longe de ser uma constante.

Como toda ferramenta nova, seu uso intenso pode causar uma série de efeitos colaterais não conhecidos ou não relacionados ainda. Por exemplo, no início da revolução da comunicação causada pelos dispositivos móveis cada vez mais cheio de recursos, aliados à expansão da própria internet, os teclados destes dispositivos causaram surtos de problemas nos tendões, inflamação que recebeu o apelido de "BlackBerry Thumb". Embora tenha sido descoberta há anos com a alcunha de síndrome de Quervain (ORTHOPEDIC AND SPORTS SPECIALISTS, 2024), somente com a explosão massiva dos tecladinhos diminutos é que a síndrome foi experimentada massivamente.

A ansiedade humana da busca pelo conhecimento também viu na interconexão da internet, crises de ansiedade e doenças novas como a nomofobia (no mobile phobia), que é quando o indivíduo não consegue ficar longe do celular (YOUGOV, 2023). O uso massivo do assistente sugere que o caminho de novas doenças ou deficiências também seja trilhado. Por isso carece de estudo sério e de discussões a seu respeito, para que seja legislado sobre.

Vale ressaltar que a partir da versão 4-o do Chat GPT você pode conversar com a IA através do aplicativo móvel. O uso excessivo do assistente virtual sugere riscos nas relações humanas, uma vez que impreterivelmente podem ser substituídas em certo grau - o assistente lhe dá as respostas que você quer ouvir, reforçando suas crenças. Não estamos distantes do filme Her (HER, 2013), onde um escritor solitário chamado Theodore, se apaixona por um sistema operacional, cuja voz fora personificada como Samantha.

A modalidade do ensino remoto também já tem seus efeitos. Há estudos que indicam que os tempos de concentração são inferiores aos presenciais, tornando necessário intervenções regulares para garantir coesão (STROM et al., 2023). Agora mude para o cenário do Chat GPT. Entenda que neste estudo, você tenha à disposição algo que não só te indique o caminho como lhe responda de forma relativamente objetiva o que for perguntado. Ou que seja capaz de fazer a pesquisa por você. Mais do que um tutor, a IA se comporta como um parceiro de trabalho. Se o usuário não for consciente dessas implicações e das limitações da ferramenta, as consequências podem ser ainda mais desastrosas no campo do aprendizado.

## 1.2.NEUTRALIDADE E REPRESENTATIVIDADE

Para dar respostas mais precisas, o modelo de aprendizagem de máquina não supervisionado é treinado com um grande volume de dados. De acordo com (BROWN et al, 2020), o Chat GPT versão 3 (175 bilhões de parâmetros) usou para ser treinado 8 anos de indexação de sites, publicações do serviço, publicações do serviço de forum Reddit com mais votos, bibliotecas de livros digitalizados - inclusive acadêmicos, e a versão em inglês da Wikipedia. Sendo que destes 93% dos dados estão no idioma inglês.

Na versão 4 do ChatGPT (MAZZONI et al., 2023), além de usar dados licenciados de provedores terceiros, o algoritmo consegue consultar dados públicos na internet. E usa um mecanismo chamado de Reinforcement Learning from Human Feedback (RLHF) para que a resposta seja direcionada para a pergunta do usuário (diretrizes ou limitações impostas ou user guardrails).

Com esta formulação é dificílimo pensar em representatividade de toda a diversidade cultural existente no mundo ou isonomia de pensamento, já que majoritariamente usa só o idioma de uma nação. Com uma base notavelmente focada na interpretação do mundo pela ótica dos Estados Unidos. É o viés consumado. Quase a garantia de amplificação de preconceitos existentes nesta sociedade. Uma ferramenta de perpetuação de injustiças e discriminações.

Não precisamos ir longe para traçar paralelos com o que já vivenciamos. Veja o caso do cinema nacional, que teve sua criatividade e pujança duramente ofuscadas pela massividade de produções cinematográficas norte-americanas. Não só como efeito de dominação cultural, mas também legislativa, dado que impreterivelmente privilegiava o estrangeiro com financiamentos vultosos. Tal situação foi denunciada, discutida, escrutinada por Paulo Emílio Gomes, no livro “Uma situação Colonial?” (Gomes, 2016). Uma das frases do autor marca a situação:

> *“O cinema é incapaz de encontrar dentro de si próprio, energias que lhe permitam escapar à condenação do subdesenvolvimento” (GOMES, 2016)*

Isto é, mesmo que o cinema nacional faça uma obra sui generis, o gosto do público já foi tomado pela outra cultura. E não há espaço ou valia no reconhecimento da própria identidade. Por outro ângulo, a questão da neutralidade, devemos ter a ciência de que a ferramenta não está disponível em todos os países do mundo. Seja pela legislação do país, seja por motivos puramente estratégicos, bélicos ou financeiros. Pelo site oficial é possível obter uma lista de países suportados (OPENAI, 2024). Nota-se a ausência de China, Rússia, Síria, Cuba, Coreia do Norte dentre outros. O ponto de análise é: se a IA aprende através de trocas de mensagens com os usuários e através de pesquisas em sites da internet, artificialmente (ou naturalmente, fosse um humano) aprenderá pouco da cultura Oriental - com a devida visão da origem. Suas inferências probabilísticas serão, portanto, apenas espelhos da realidade. Eventualmente reforçadores de estereótipos.

Não é difícil pensar também em diferentes manifestações de racismo. A geração de imagens, recurso do DALL-E, incorporado aos modelos do Chat GPT tem inúmeros casos de falta de representatividade. (NUNES, 2022) faz uma importante revelação ao testar a busca por termos como “homem foto” e “mulher foto” e contar as ocorrências. Os algoritmos de IA das buscas foram treinados

para encontrar referências de homem e mulher brancos europeus. É o racismo estrutural sedimentado.

Nunes (2022) também cita (BUOLAMWINI, 2016), pesquisadora que criou a Algorithimc Justice League (ALGORITHMIC JUSTICE LEAGUE, 2024), para que algoritmos evitem vieses. Ela demonstrou em sua pesquisa no MIT Media Lab que os algoritmos foram treinados para identificar rostos de brancos. Ela percebeu que precisava usar uma máscara branca para ser identificada como humana por alguns softwares de identificação facial.

Extrapolando a ideia, se solicitar ao ChatGPT para gerar uma imagem de um casal, sem passar qualquer instrução que sugira diversidade, na maioria dos casos a imagem gerada será um casal branco. Não que ele tenha sido programado para isto, mas a questão do treinamento e dos dados utilizados acabam carregando os chamados vieses algorítmicos. Imagine que as empresas desenvolvam aplicações de classificação de pessoas para fornecer um empréstimo, por exemplo. O bem-estar estaria ameaçado.

## 1.3.TRANSPARÊNCIA

É muito claro que vivenciamos uma corrida para melhores modelos de IA generativa. Empresas como Microsoft, Open AI, Claude AI, Meta e outras, investem pesado a caminho de uma IA multimodal mais completa com capacidade cognitiva semelhante a humana, conhecida como IA Geral ou AGI. Mas as questões de transparência levantaram pressões internas nestas empresas. É tão crítico que funcionários e ex-funcionários da própria OpenAI e de outras empresas fronteiriças de IA, criaram uma carta aberta chamada Right to Warn AI. Na carta aberta (RIGHT TO WARN, 2024) deixam claro que as empresas têm fortes incentivos financeiros para evitar a supervisão eficaz.

Alegam ainda que as empresas possuem relatórios de falhas, de imprecisões, seus controles, riscos implícitos de seus processos e que estes não são públicos. As empresas têm pouca obrigação de partilhar estas informações com governos e sugere nas entrelinhas que as empresas não dão espaço para uma cultura aberta que fomente a discussão.

Para se ter uma ideia de precificação, o chat GPT gasta diariamente cerca de 700 mil dólares para funcionar (PATEL; AHMAD, 2023). Convertendo em reais, sem impostos, a soma passaria 1,384 bilhões de reais. Fora os custos de treinamento, licenciamento de conteúdo e demais custos operacionais.

Embora seja razoavelmente público que os primeiros modelos usaram anos de coleta de informações da internet, pouco se sabe a respeito dos métodos que foram usados na filtragem destes dados. Somente recentemente a Open AI lançou um programa de parceria de dados, o Open AI Data Partnerships (OPENAI, 2024). E através dele fez parcerias com o governo da Islândia e com uma instituição sem fins lucrativos chamada FreeLaw para usar sua base de casos jurídicos no treinamento. São bases tratadas e revisadas.

## 1.4.PRIVACIDADE DOS DADOS

Desde o Chat GPT versão 2, o serviço de chat utiliza um mecanismo de aprendizagem por reforço com recompensas a partir das perguntas dos humanos (ZIEGLER et al., 2019). Para isso o histórico de uso é armazenado e o perfil do usuário vai sendo aos poucos traçados.

Em 2023, o serviço de proteção a dados da Itália bloqueou o serviço de chatbot pelo não cumprimento das normas estabelecidas pela GDPR (General Data Protection Regulation), dado que não havia naquele tempo tratamento adequado a adolescentes abaixo de 13 anos. E mais grave, detectaram que até 1,2% de dados pessoais de um indivíduo poderiam vazar para outro usuário do sistema.

Controles foram estabelecidos para que o histórico de conversas seja desabilitado e dados pessoais possam não só ser exportados, com limpos. Além de um parco controle de acesso a menores de idade. Mas ainda é questionável a qualidade destas proteções, dado que os controles são ativados por padrão – e a maioria dos usuários nem se dá conta disso.

Existem estudos como o de (LIU et al., 2023), que demonstram de forma empírica prompts que conseguiram liberar todas as restrições do modelo de linguagem. A preocupação excessiva com vazamentos não parece ser infundada portanto.

## 3.METODOLOGIA

Com todas estas questões analisadas, criamos um questionário qualitativo para avaliar como o uso nas empresas tem se dado. O questionário foi disparado para grupos de entusiastas de IA e profissionais da computação que trabalham em áreas diversas.

As questões do questionário foram:

1. Como que frequência você usa IA em seu contexto de trabalho?
2. Você acredita que a sua organização tem políticas suficientes para evitar o uso excessivo de IA?
3. Na sua opinião, a IA utilizada pela sua organização é neutra e imparcial?
4. Sua empresa toma medidas para garantir que os dados utilizados para treinar modelos de IA sejam representativos da diversidade da população?
5. A sua organização informa claramente aos usuários sobre o uso de IA em seus serviços/produtos?
6. . Os algoritmos e processos de decisão da IA são acessíveis e compreensíveis para os funcionários?
7. Você considera que a sua empresa está em conformidade com as regulamentações de privacidade de dados ao usar IA?
8. A sua organização possui uma estrutura clara de accountability para o uso de IA?
9. Quem é responsável por monitorar e garantir a ética na utilização de IA em sua empresa?

10. Quais procedimentos são seguidos quando ocorre um erro significativo em um sistema de IA?

## 4.APRESENTAÇÃO DOS RESULTADOS

Embora os resultados não tenham sido em grande volume – apenas 17 respostas - para que a amostra seja considerada, ela fornece preciosos insights de como as empresas estão em sua maioria despreparadas para o uso equalitário e responsável.

47,1% das respostas afirmam que utilizam muito frequentemente o IA em suas empresas. Mas 35,3% deles não acreditam que a empresa tenha política para evitar o uso excessivo. Outros 35,3% acreditam que a empresa tem parcialmente política para evitar o uso excessivo. Apenas 23,5% acreditam que o algoritmo de IA é imparcial ou neutro. Apenas 29,5% afirmam que as empresas tomam sempre medidas para que os modelos sejam representativos da diversidade populacional.

47,1% das empresas informam claramente seus usuários sobre o uso de IA em seus produtos. Mas 11,8% das empresas nunca informam.

Apenas 29,5% acreditam que a IA das empresas são acessíveis e compreensíveis para todos os funcionários.

O único ponto positivo da pesquisa é que 47,1% das pessoas acreditam que suas empresas estão em conformidade de privacidade de dados. Mas 35,3% não sabem se a empresa tem alguma estrutura clara de prestação de contas para o uso de IA. Em geral a área de TI ou de Compliance absorve a responsabilidade pelo uso ético.

## 5.CONSIDERAÇÕES FINAIS

Ao adotarmos o Chat GPT para as atividades cotidianas, possibilitamos a automação de tarefas e a substituição de funções, o que pode reduzir a necessidade de força de trabalho humana. É crucial examinarmos as implicações dessas mudanças. Para mitigar os impactos negativos, devemos equilibrar e, se necessário, restringir o uso da IA assegurando a proteção dos direitos individuais, a promoção da transparência e da responsabilidade, a prevenção de discriminação e viés, e a preservação da autonomia e dignidade humanas.

Recomendamos que as empresas e a sociedade civil criem ambientes que fomentem discussões de forma constante, para que o uso de IA respeite os seguintes pontos:

**Neutralidade Algorítmica:** Criar algoritmos que evitem preconceitos e discriminações, sendo desenvolvidos e implementados com esse propósito;

**Transparência:** Prover explicações claras sobre o processo de coleta, armazenamento e utilização de dados, assim como os critérios e mecanismos de decisão dos algoritmos;

**Prestação de Contas** (Accountability): Implementar mecanismos efetivos de responsabilização, com auditorias e avaliações éticas;

**Proteção da Privacidade:** Adotar políticas rigorosas para a proteção de dados pessoais, assegurando que a coleta e uso de dados sejam éticos e seguros, com respeito à privacidade dos usuários, e fomentar o debate público sobre essas práticas;

**Equidade e Justiça:** Garantir que as decisões automatizadas sejam justas e equitativas, evitando discriminações de qualquer natureza, como racial, de gênero ou por origem, através da implementação de critérios transparentes e auditáveis, com a devida informação ao usuário;

**Inovação Responsável:** Fomentar o desenvolvimento de IA que respeite os direitos e valores democráticos, promovendo uma inovação tecnológica responsável e sustentável;

**Conformidade com Regulamentações:** Garantir que o desenvolvimento e uso da IA estejam em conformidade com as regulamentações vigentes, incluindo leis de proteção de dados e direitos individuais, o marco civil da internet, a lei geral de proteção aos dados e as cláusulas pétreas da constituição;

**Conscientização:** Criar e distribuir materiais que esclareçam os riscos e ofereçam recomendações para o uso racional das ferramentas de IA.

# 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

### 1 Resumo do artigo escolhido

Artigo escolhido: AHMAD et al Requirements practices and gaps when engineering Artificial Intelligence systems.

### 1.1 Objetivo do estudo

O estudo tem como objetivo analisar as práticas de requisitos e as lacunas existentes no desenvolvimento de sistemas de Inteligência Artificial (IA)centrados no ser humano. Ele busca  entender como as práticas de engenharia de requisitos estão sendo aplicadas no contexto da IA, especialmente em sistemas que interagem com os seres humanos, e como essas práticas podem ser melhoradas para atender às necessidades e expectativas dos usuários.

### 1.2 Problema, oportunidade ou situação que levou ao estudo

A necessidade deste estudo surgiu devido à crescente importância de desenvolver sistemas de IA que sejam eficazes na interação com os seres humanos. A engenharia de requisitos, tradicionalmente usada no desenvolvimento de sistemas de software, precisa ser adaptada para lidar com as complexidades e desafios únicos da IA centrada no ser humano. As lacunas na aplicação das práticas de requisitos podem levar a falhas na criação de sistemas de IA que atendam adequadamente às necessidades dos usuários, resultando em uma experiência de usuário insatisfatória e uma adoção limitada desses sistemas.

### 1.3 Metodologia usada para obter e analisar as informações do estudo

Os autores adotaram uma metodologia qualitativa, realizando uma revisão sistemática da literatura e entrevistas com especialistas da área de engenharia de requisitos e desenvolvimento de IA. Eles examinaram as práticas atuais, identificando as lacunas na aplicação da engenharia de requisitos em sistemas de IA. Além disso, realizaram um estudo de caso para analisar como essas práticas são implementadas em cenários reais de desenvolvimento de sistemas de IA centrados no ser humano.

### 1.4 Resultados obtidos pelo estudo

Os principais resultados do estudo incluem a identificação de várias lacunas nas práticas de engenharia de requisitos aplicadas ao desenvolvimento de sistemas de IA centrados no ser humano. Entre as lacunas destacadas estão a falta de foco nas necessidades emocionais e cognitivas dos usuários, a dificuldade em integrar requisitos éticos e sociais nas fases iniciais do desenvolvimento e a falta de uma abordagem centrada no usuário para a coleta de requisitos. Os autores propõem uma série de melhorias, incluindo a adoção de abordagens mais colaborativas e interdisciplinares, além de uma melhor integração de feedbacks dos usuários e stakeholders ao longo do ciclo de vida do desenvolvimento de IA.

# 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.

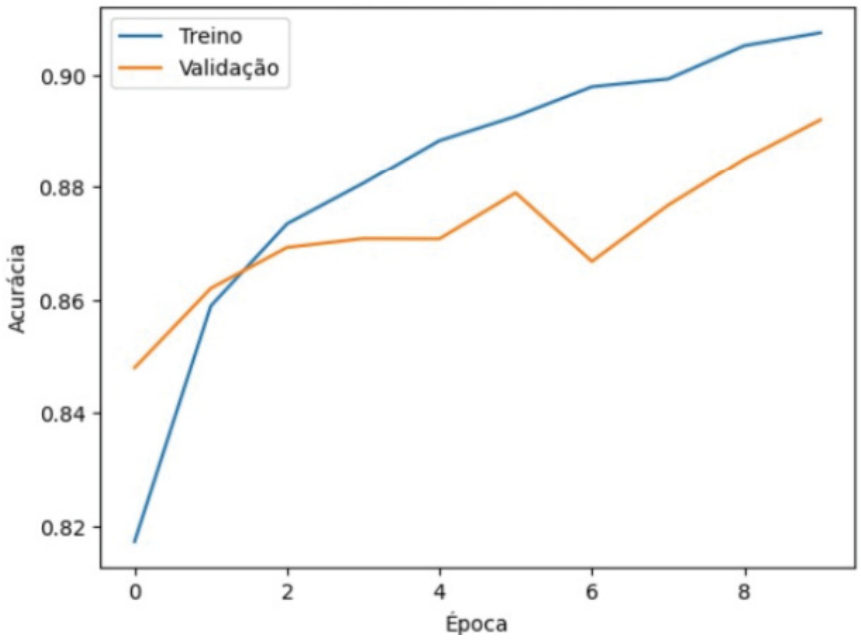
```
url = "https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-cat_on_snow.jpg"
```

```
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

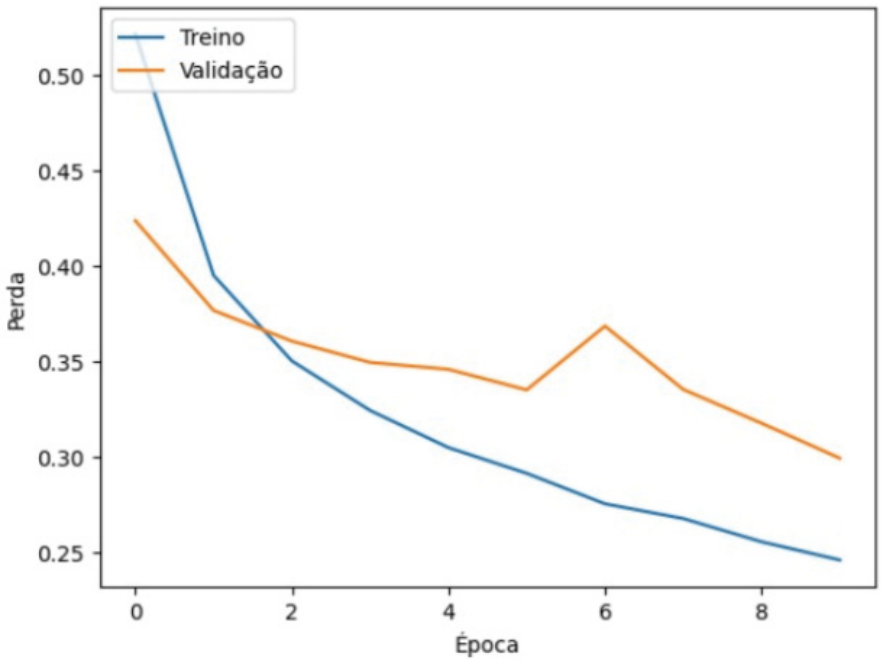
### 1 Classificação (RNA)

Figura 1: Gráfico de Acurácia do modelo



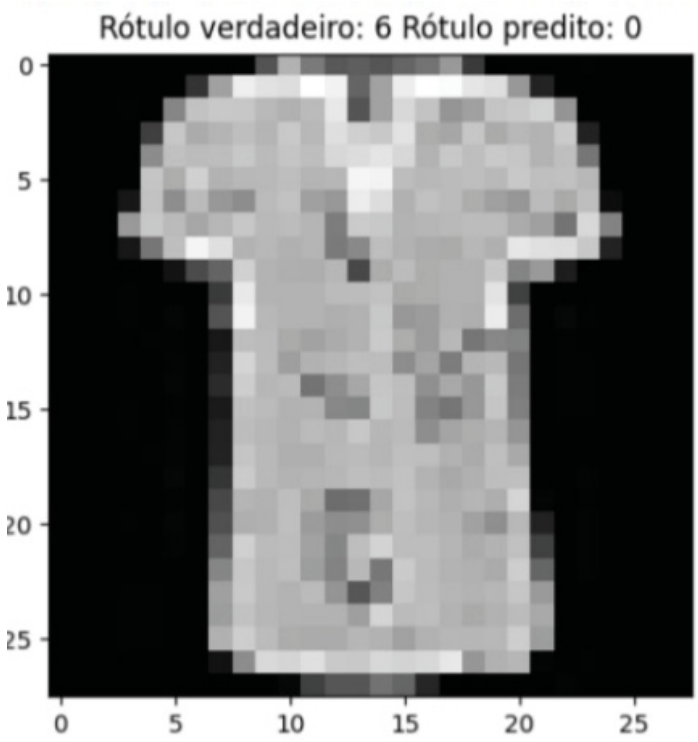

Fonte: O autor (2025)

Figura2: Gráfico de Perda do modelo




Fonte: O autor (2025)

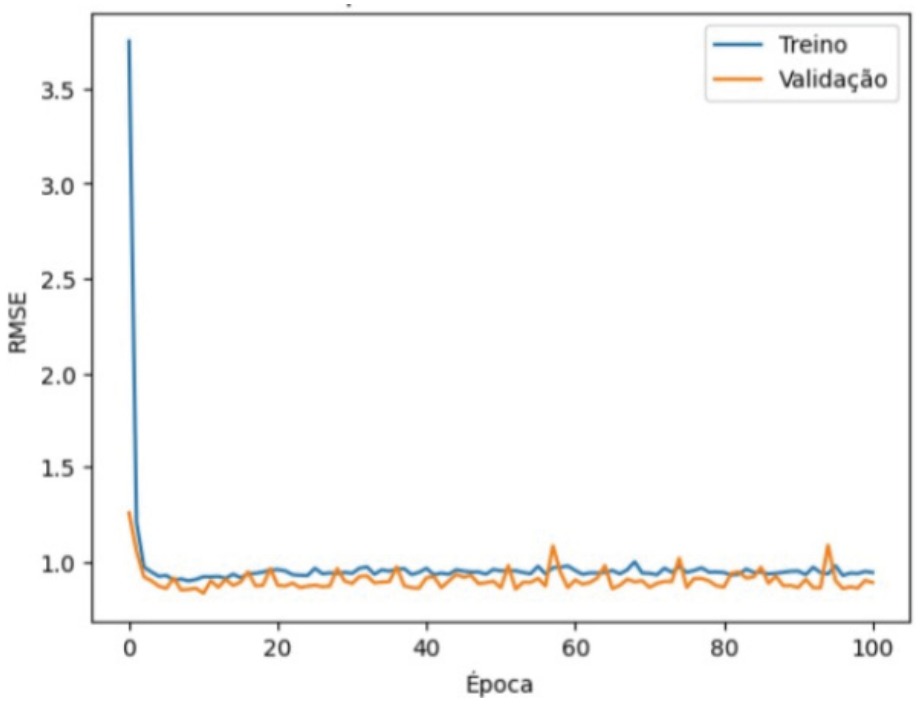
Figura 3: Classificação errada




Fonte: O autor (2025)

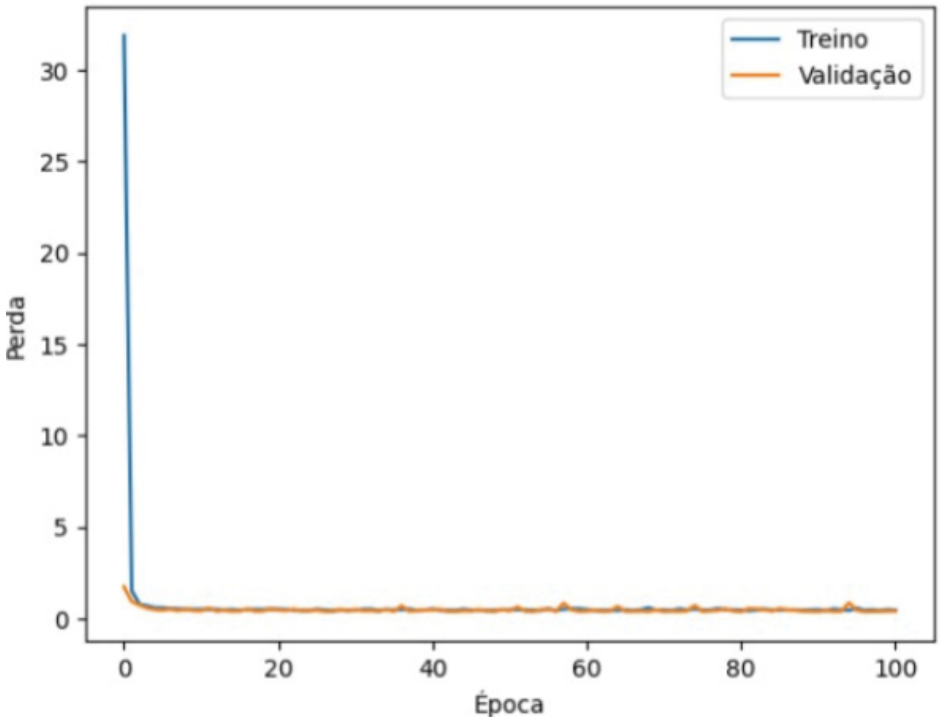
## 2 Regressão (RNA)

Figura 4: Gráfico de erro quadrático médio do modelo



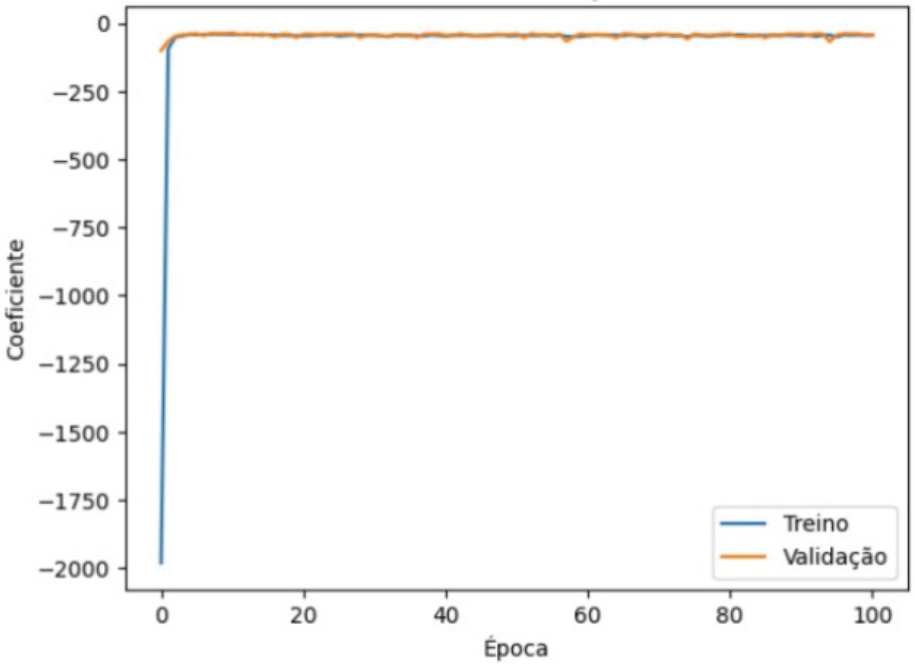

Fonte: O autor (2025)

Figura 5: Gráfico de perda do modelo



Fonte: O autor (2025)

Figura 6: Gráfico de coeficiente do modelo




Fonte: O autor (2025)

O modelo de previsão da qualidade do vinho obteve os seguintes resultados:

1. MSE de 0.3936: Isso significa que, em média, a diferença entre as previsões do modelo e os valores reais de qualidade, quando elevada ao quadrado, é de 0.4038. Quanto menor esse valor, melhor, então 0.4038 é um erro razoável, mas há espaço para melhorias.

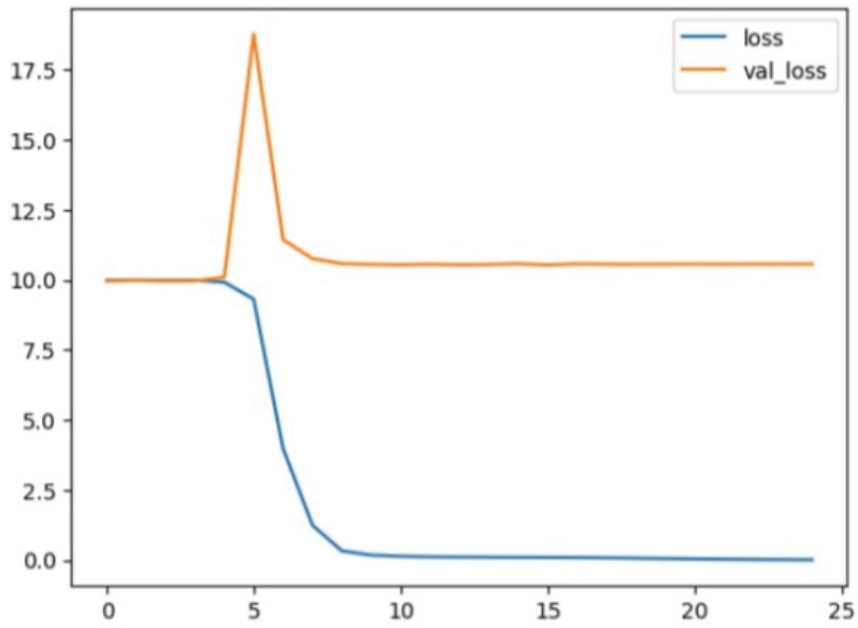
2. RMSE de 0.6274: Este é o erro médio do modelo, mas em unidades diretas de qualidade do vinho. Ou seja, em média, o modelo erra por cerca de 0.64 unidades na escala de qualidade. Novamente, não é um erro gigantesco, mas também não é extremamente preciso.

3. $R^2$ de 0.3422: Isso quer dizer que o modelo consegue explicar cerca de 40% da variação na qualidade do vinho. Ou seja, ele tem uma boa capacidade explicativa, mas ainda há uma grande parte da variação que não é explicada pelas características do vinho usadas no modelo.

Em resumo, o modelo está razoavelmente bom, mas há margem para melhorar. Ele consegue capturar uma parte significativa da variação da qualidade do vinho, mas não de forma perfeita, e ainda com um erro médio considerável.

## 3 Sistemas de Recomendação

Figura 7: Gráfico de perda do modelo




Fonte: O autor (2025)

Figura 8: Exemplo de recomendação de livro

```
3406/3406 ━━━━━━━━━━━━━━━━━━━━ 12s 4ms/step
Recomendação: Livro -  2412  /  1.1802464 *
```

```
# Identificação do título sugerido para usuário
print( df[df['ISBN'] == books[idx]].values[0])
```

```
[2412 'The Perfect Storm: A True Story of Men Against the Sea'
 'Sebastian Junger' 1997 'Little Brown and Company' 10202 10]
```

Fonte: O autor (2025)

## 4 Deepdream

Figura 9: Imagem onírica obtida por Main Loop

Fonte: O autor (2025)

Figura 10: Imagem onírica obtida ao levar modelo até oitava

Fonte: O autor (2025)

## Linhas de código

```
#Bibliotecas

import tensorflow as tf
import matplotlib.pyplot as plt
```

```
import numpy as np

from sklearn.model_selection import train_test_splita
from sklearn.preprocessing import StandardScaler
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

# Carregando base https://www.tensorflow.org/datasets/catalog/fashion_mnist?hl=pt-br

data = tf.keras.datasets.fashion_mnist.load_data()

(x_train, y_train), (x_test, y_test) = data

x_data = np.concatenate([x_train, x_test])
y_data = np.concatenate([y_train, y_test])

# Split  75% treinamento e 25% teste
x_train, x_test, y_train, y_test = train_test_split(
    x_data, y_data, test_size=0.3, random_state=42
)


# Número de linhas e colunas
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)

#Exemplo dos dados
print(x_train)

#Normalização dos pixels

x_train = x_train / 255.0
x_test = x_test / 255.0

#Criação do modelo
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')
])


#Otimização do modelo com gradiente descendente, avaliando com acurácia
#OBS: A função de perda escolhida é mais adequada a problemas de classificação, quando o índice é inteiro e representa a classe


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

# Treinamento com 10 épocas

history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

#Checando resultados
print("Pontuação de treinamento: ", model.evaluate(x_train,y_train))
print("Pontuação de teste: ", model.evaluate(x_test, y_test))



# Gráficos durante o treinamento

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Acurácia do Modelo')
plt.ylabel('Acurácia')
plt.xlabel('Época')
plt.legend(['Treino', 'Validação'], loc='upper left')
plt.show()
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Perda do modelo')
plt.ylabel('Perda')
plt.xlabel('Época')
plt.legend(['Treino', 'Validação'], loc='upper left')
plt.show()

#Exemplo de predição do modelo
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)

# Demonstrando uma predição errada
misclassified = np.where(y_pred != y_test)[0]
i = np.random.choice(misclassified)

plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("Rótulo verdadeiro: %s Rótulo predito: %s" % (y_test[i], y_pred[i]))

# Bibliotecas
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.python.keras import backend
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt

#Importação dos dados
```

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

#Traduzindo colunas para entender melhor os dados
df = pd.DataFrame(data)

df.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
]

print(df)

# Separação dos dados, removendo primeira coluna
X = df.iloc[1:, :-1].values.astype(float)
Y = df.iloc[1:, -1].values.astype(float)

print(X)

# Separação para treinamento e teste do modelo
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)

# Definição do modelo

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)

# Funções de avaliação da regressão

def rmse(y_true, y_pred):
  return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))

def r2(y_true, y_pred):
  media = backend.mean(y_true)
  num   = backend.sum(backend.square(y_true - y_pred))
  den   = backend.sum(backend.square(y_true - media))
  return (1.0 - num/den)

#Definição do modelo

optimizer=tf.keras.optimizers.Adam(learning_rate=0.05)
# optimizer=tf.keras.optimizers.SGD(learning_rate=0.2, momentum=0.5)
# optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
              loss="mse",
              metrics=[rmse, r2])

# Callback para parar treinamento
early_stop = tf.keras.callbacks.EarlyStopping(
                        monitor='val_loss',
                        patience=20,
                        restore_best_weights=True)

# Épocas de treinamento com callback
history = model.fit(x_train, y_train,
              epochs=1500,
              validation_data=(x_test, y_test),
              callbacks=[early_stop])

# Resultados do treinamento e teste
print("Resultado treinamento: ", model.evaluate(x_train, y_train))
```

```
print("Resultado teste: ", model.evaluate(x_test, y_test))

# Gráficos durante o treinamento e teste

plt.plot(history.history['rmse'])
plt.plot(history.history['val_rmse'])
plt.title('Erro quadrático médio do modelo')
plt.ylabel('RMSE')
plt.xlabel('Época')
plt.legend(['Treino', 'Validação'], loc='upper right')
plt.show()


plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Perda do modelo')
plt.ylabel('Perda')
plt.xlabel('Época')
plt.legend(['Treino', 'Validação'], loc='upper right')
plt.show()


plt.plot(history.history['r2'])
plt.plot(history.history['val_r2'])
plt.title('Coeficiente de determinação do modelo')
plt.ylabel('Coeficiente')
plt.xlabel('Época')
plt.legend(['Treino', 'Validação'], loc='lower right')
plt.show()

# Predição com o modelo

y_pred = model.predict(x_test).flatten()
print(y_pred)

# Avaliando modelo treinado
mse  = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
```

```
r2   = r2_score(y_test, y_pred)

print("mse     = ", mse)
print("rmse    = ", rmse)
print("r2      = ", r2)

#Bibliotecas

import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Embedding, Flatten, Concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD, Adam

from sklearn.utils import shuffle

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

# Importação da base.csv (via ação do usuario)
from google.colab import files
import io
uploaded = files.upload()

#Carregando base importada (usamos o on bad lines para evitar falhas)
import io
filename = next(iter(uploaded))
df = pd.read_csv(filename,delimiter=",", on_bad_lines="skip")
df

#Embeddings (ISBN e ID_usuário precisam ser categóricas)

df.ISBN = pd.Categorical(df.ISBN)
df['ISBN'] = df.ISBN.cat.codes

df.ID_usuario = pd.Categorical(df.ID_usuario)
df['ID_usuario'] = df.ID_usuario.cat.codes
```

```
# Dimensões
N = len(set(df.ISBN))
M = len(set(df.ID_usuario))

# dimensão do embedding (tentar outros)
K = 10

# Livro
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

# Usuário
m = Input(shape=(1,))
m_emb = Embedding(M, K)(m)  # saída : num_samples, 1, K
m_emb = Flatten()(m_emb)    # saída : num_samples, K

x = Concatenate()([u_emb, m_emb])

x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

model = Model(inputs=[u, m], outputs=x)

# Criação do modelo
model.compile(
   loss="mse",
   optimizer=SGD(learning_rate=0.08, momentum=0.9)
)

# Separando os dados para treino e teste (considerando apenas 3 itens da tabela)

ISBN, ID_usuario, Notas = shuffle(df.ISBN, df.ID_usuario, df.Notas)

Ntrain = int(0.8 * len(Notas)) # separar os dados 80% x 20%
```

```
train_ISBN = ISBN[:Ntrain]
train_ID_usuario = ID_usuario[:Ntrain]
train_Notas = Notas[:Ntrain]

test_ISBN = ISBN[Ntrain:]
test_ID_usuario = ID_usuario[Ntrain:]
test_Notas = Notas[Ntrain:]

avg_Notas = train_Notas.mean()
train_Notas = train_Notas - avg_Notas
test_Notas = test_Notas - avg_Notas


# Treinamento do modelo em épocas
epochs = 25

history = model.fit(
   x=[train_ISBN, train_ID_usuario],
   y=train_Notas,
   epochs=epochs,
   batch_size=1024,
   verbose=2, # não imprime o progresso
   validation_data=([test_ISBN, test_ID_usuario], test_Notas)
)

# Gráfico de perda
plt.plot(history.history["loss"], label="loss")
plt.plot(history.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

# Gerar o array com o usuário único
# repete a quantidade de livros
input_usuario = np.repeat(a=29888, repeats=N)
books = np.array(list(set(ISBN)))

preds = model.predict( [input_usuario, books] )
```

```
        # descentraliza as predições
        rat = preds.flatten() + avg_Notas

        # índice da maior nota
        idx = np.argmax(rat)

        print("Recomendação: Livro - ", books[idx], " / ", rat[idx] , "*")


        # Identificação do título sugerido para usuário
        print( df[df['ISBN'] == books[idx]].values[0])

        #Bibliotecas

        import tensorflow as tf
        import numpy as np
        import matplotlib as mpl
        import IPython.display as display
        import PIL.Image

        # Imagem de referência
        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>'))

#Modelo de base já treinado
base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')

# Maximizando as ativações das camadas
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]

# Criação do modelo
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)

def calc_loss(img, model):
  # Passe a imagem pelo modelo para recuperar as ativações.
  # Converte a imagem em um batch de tamanho 1.
  img_batch = tf.expand_dims(img, axis=0)
  layer_activations = model(img_batch)
  if len(layer_activations) == 1:
    layer_activations = [layer_activations]

  losses = []
  for act in layer_activations:
    loss = tf.math.reduce_mean(act)
    losses.append(loss)

  return  tf.reduce_sum(losses)
```

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

  @tf.function(
      input_signature=(
        tf.TensorSpec(shape=[None,None,3], dtype=tf.float32),
        tf.TensorSpec(shape=[], dtype=tf.int32),
        tf.TensorSpec(shape=[], dtype=tf.float32),)
  )
  def __call__(self, img, steps, step_size):
      print("Tracing")
      loss = tf.constant(0.0)

      for n in tf.range(steps):
        with tf.GradientTape() as tape:
          # Gradientes relativos a img
          tape.watch(img)
          loss = calc_loss(img, self.model)

        # Calculo do gradiente da perda em relação aos pixels da imagem de entrada.
        gradients = tape.gradient(loss, img)

        # Normalizacao dos gradintes
        gradients /= tf.math.reduce_std(gradients) + 1e-8

        # Na subida gradiente, a "perda" é maximizada.
            # Você pode atualizar a imagem adicionando diretamente os gradientes (porque eles têm o mesmo formato!)
        img = img + gradients*step_size
        img = tf.clip_by_value(img, -1, 1)

      return loss, img

deepdream = DeepDream(dream_model)

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)

import time
start = time.time()

OCTAVE_SCALE = 1.30

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[:-1]
float_base_shape = tf.cast(base_shape, tf.float32)
```

```
for n in range(-2, 3):
  new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

  img = tf.image.resize(img, new_shape).numpy()

  img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```

# APÊNDICE 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**)

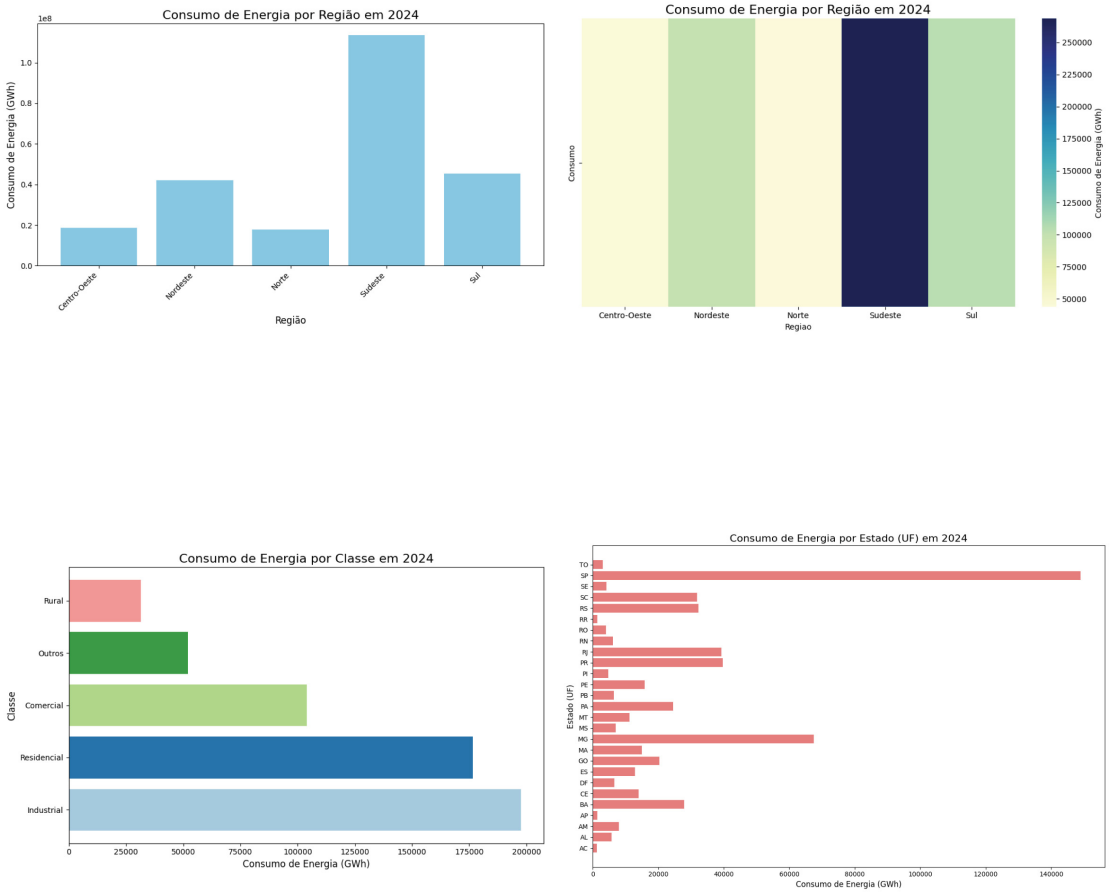
## B – RESOLUÇÃO

### 1 Conjunto de Dados Utilizado

O objetivo desta análise é compreender o comportamento do consumo de energia elétrica no Brasil até o ano de 2024. Para tanto, foram utilizados dados extraídos do portal de Dados Abertos da Empresa de Pesquisa Energética (EPE). As principais fontes de dados utilizadas incluem:

- Consumo Mensal de Energia Elétrica
- Disponível no Portal de Dados Abertos da EPE.
- Abrange informações sobre o consumo de energia elétrica por classe (residencial, industrial, comercial, etc.), por região e por estado (UF).
- Os dados foram coletados mensalmente e atualizados pela equipe da EPE com base nas informações fornecidas pelo Sistema de Acompanhamento do Mercado (SAM) e pelas reuniões da Comissão Permanente de Análise e Acompanhamento do Mercado de Energia Elétrica (COPAM).

Figura 11: Análise do consumo de energia no Brasil em 2024: Distribuição por região, estado e classe.




Fonte: O autor (2025)

## 2 Ferramentas Utilizadas para Análise e Visualização

A análise e visualização dos dados foi realizada utilizando a linguagem Python e suas respectivas bibliotecas, destacando-se:

- Pandas: Utilizada para o tratamento e manipulação dos dados tabulares, permitindo agregações, filtragens e cálculos estatísticos necessários para a etapa exploratória.
- Matplotlib: Aplicada para a construção de gráficos de barras, linhas e mapas de calor, proporcionando controle detalhado sobre o layout e personalização das visualizações.
- Seaborn: Empregada para aprimorar os gráficos estatísticos, facilitando a criação de visualizações mais sofisticadas, como distribuições e correlações entre variáveis.

A combinação dessas ferramentas possibilitou não apenas a organização e limpeza do conjunto de dados, mas também a criação de representações visuais claras e informativas, capazes de evidenciar padrões no consumo de energia elétrica em diversas dimensões (tempo, região, estado e classe).

## 3 Público-Alvo e Objetivo da Visualização

O público-alvo desta visualização abrange profissionais do setor energético, gestores públicos, analistas de dados e outros stakeholders envolvidos com a análise do consumo de energia elétrica no Brasil. O objetivo é proporcionar uma visão clara acerca das variações do consumo de energia por classe, região e estado, além de identificar tendências ao longo do tempo.

## 4 Visualizações de Dados

### 4.1 Consumo de Energia por Região em 2024

- Gráfico de Barras: Exibe o consumo de energia por região, destacando a região Sudeste como a maior consumidora, seguida pela região Nordeste.
- Mapa de Calor: Apresenta a variação do consumo de energia elétrica entre as diferentes regiões do Brasil.

### 4.2. Consumo de Energia por Classe em 2024

- Gráfico de Barras Horizontais: Mostra a classe Industrial como a maior consumidora de energia, seguida pela classe Residencial. As classes Comercial e Rural, embora com menores índices de consumo, ainda representam setores relevantes.

### 4.3. Consumo de Energia por Estado (UF) em 2024

- Gráfico de Barras: Destaca os estados com maior consumo de energia, com São Paulo (SP) e Minas Gerais (MG) liderando a demanda elétrica.

**4.4. Consumo de Energia por Ano (2005–2025)**

• Gráfico de Barras: Apresenta a evolução do consumo de energia ao longo dos anos, evidenciando flutuações entre períodos de crescimento e declínios. Os anos mais recentes (2021–2024) mostram um aumento significativo, enquanto anos como 2009, 2015, 2016 e 2020 apresentaram quedas no consumo.

## 5. Descrição da Narrativa/Storytelling

**5.1. Consumo por Região (2024)**

O gráfico de consumo por região revela que o Sudeste concentra o maior consumo de energia elétrica, em virtude de sua forte presença industrial e da elevada densidade populacional. Este cenário indica que a região exige maior atenção no planejamento energético, a fim de evitar sobrecargas e gargalos na rede elétrica. Em segundo lugar, o Nordeste mostra um crescimento econômico expressivo, o que implica no aumento da demanda e reforça a necessidade de políticas públicas que incentivem a expansão da oferta de energia e a diversificação das fontes.

**5.2. Consumo por Classe (2024)**

A visualização demonstra que a classe Industrial lidera o consumo, seguida de perto pela classe Residencial. Este padrão reforça dois pontos importantes:

• O peso significativo da indústria como principal consumidora, o que demanda políticas voltadas para a melhoria da eficiência energética e a adoção de tecnologias limpas.

• O aumento contínuo do consumo residencial, impulsionado pela urbanização e pela maior disponibilidade de eletrodomésticos e eletrônicos.

Embora o consumo nas classes Comercial e Rural seja relativamente menor, essas áreas não devem ser negligenciadas, uma vez que também podem ser alvo de políticas de eficiência energética específicas.

**5.3. Consumo por Estado (UF) (2024)**

O gráfico que mostra o consumo por estado destaca São Paulo (SP) e Minas Gerais (MG) como os maiores consumidores de energia, o que reflete a alta concentração industrial e populacional nesses estados. Para os planejadores e gestores do setor energético, isso implica na necessidade de priorizar investimentos em infraestrutura, além de promover soluções como geração distribuída e o uso de energias renováveis nessas regiões.

**5.4. Consumo por Ano (2005–2025)**

O gráfico histórico de consumo de energia elétrica revela uma trajetória de crescimento consistente, mas com quedas significativas em anos como 2009, 2015, 2016 e 2020, períodos esses associados a crises econômicas ou a eventos excepcionais, como a pandemia de COVID-19. A retomada observada entre 2021 e 2024 indica um aumento da demanda, o que reforça a necessidade de um planejamento energético flexível e adaptável, capaz de responder a choques externos e variações de mercado.

## 6. Ações Possíveis

### 6.1. Planejamento de Infraestrutura no Sudeste

Recomenda-se o investimento na expansão e modernização da rede elétrica na região Sudeste, a fim de reduzir os riscos de sobrecarga e garantir a continuidade do fornecimento de energia elétrica.

### 6.2. Fomento à Eficiência Energética nas Indústrias

A adoção de tecnologias mais eficientes deve ser incentivada para reduzir desperdícios de energia, além de apoiar o investimento em fontes de energia limpa. A criação de programas de subsídios e financiamentos para empresas que implementem soluções sustentáveis pode acelerar essa transição.

### 6.3. Monitoramento das Regiões e Estados em Expansão

É fundamental o acompanhamento de estados como Minas Gerais e regiões como o Nordeste, que apresentam crescimento expressivo no consumo de energia elétrica. Esse monitoramento permitirá prever demandas futuras e direcionar investimentos estratégicos.

### 6.4. Expansão da Capacidade de Geração de Energia

É imperativo ampliar a capacidade de geração de energia no Brasil, diversificando a matriz energética com fontes renováveis, como solar, eólica e biomassa, garantindo um sistema resiliente e adaptável a crises e flutuações econômicas.

### 6.5. Educação Energética para Residências

Promover campanhas de conscientização sobre o uso eficiente da energia elétrica nas residências, incentivar a troca de eletrodomésticos antigos por modelos mais eficientes e apoiar a instalação de sistemas de geração distribuída, como painéis solares, são ações cruciais para reduzir o consumo.

## 7. Conclusão

A análise do consumo de energia elétrica no Brasil revela um cenário de crescimento contínuo, porém concentrado, caracterizado por grandes desigualdades regionais e setoriais. Os gráficos apresentados destacam o Sudeste e os estados de São Paulo e Minas Gerais como os principais polos consumidores, refletindo a elevada concentração populacional e industrial. Ao mesmo tempo, o Nordeste ganha relevância, demonstrando expansão econômica e aumento da demanda.

O setor industrial é o maior responsável pelo consumo de energia elétrica, mas o setor residencial também apresenta um aumento substancial no consumo, o que destaca a necessidade de políticas públicas voltadas para a eficiência energética e o uso sustentável da energia. A análise histórica entre os anos de 2005 e 2025 indica uma tendência de alta no consumo, com exceções em anos de crise econômica ou eventos excepcionais, como a pandemia.

Portanto, a narrativa dos dados evidencia a importância de investimentos estratégicos em infraestrutura, eficiência energética e diversificação da matriz energética para garantir a sustentabilidade do fornecimento de energia elétrica, bem como a competitividade econômica, a qualidade de vida e a preservação ambiental no Brasil.

# 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.

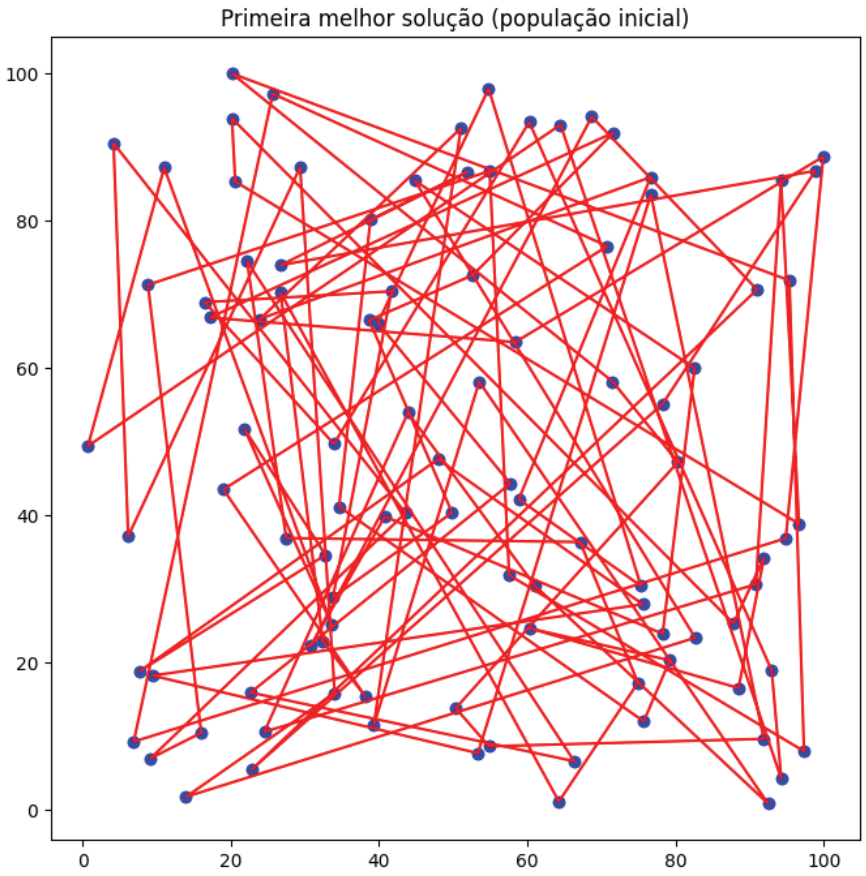
O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

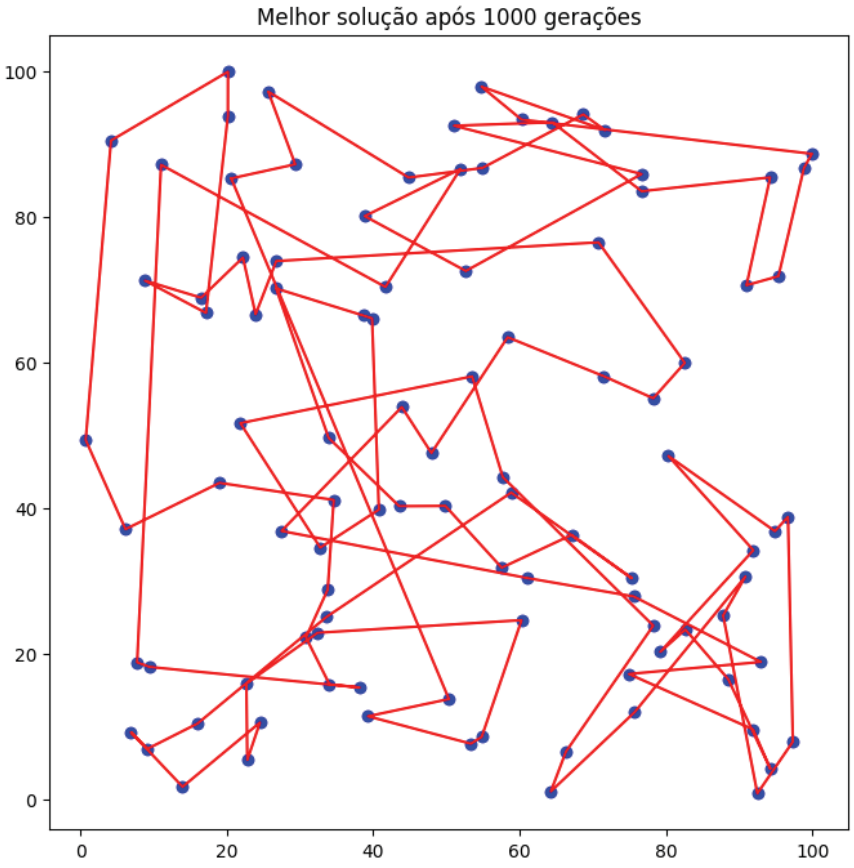
O PDF deverá conter o código-fonte e as imagens obtidas.

## B – RESOLUÇÃO

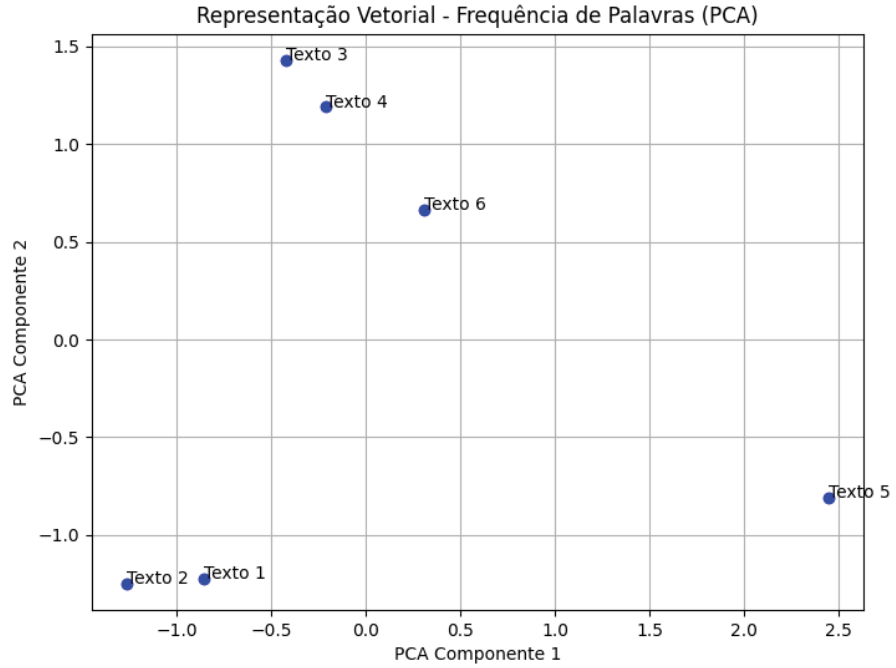
```
## Biliotecas
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import permutations

## Parâmetros
# Parâmetros Iniciais
NUM_CIDADES = 100
POPULACAO_SIZE = 100
GERACOES = 1000
MUTACAO_RATE = 0.01
CROSSOVER_RATE = 0.90

cidades = np.random.rand(NUM_CIDADES, 2) * 100

## Funções de apoio

# Calcula a distância euclidiana (comprimento do vetor) total percorrida no
caminho
def calcular_distancia(caminho):
    return sum(np.linalg.norm(cidades[caminho[i]] - cidades[caminho[i+1]])
for i in range(len(caminho)-1))


def criar_populacao():
    return [np.random.permutation(range(1, NUM_CIDADES)).tolist() for _ in
range(POPULACAO_SIZE)]

# Avalia a população com base na distância total do percurso
def avaliar_populacao(populacao):
    return sorted(populacao, key=lambda ind: calcular_distancia([0] + ind +
[0]))

# Seleciona um indivíduo por torneio
def selecao_torneio(populacao):
    torneio = random.sample(populacao, 5)
    return min(torneio, key=lambda ind: calcular_distancia([0] + ind +
[0]))
```

```
# Aplica crossover genético nos pais para gerar dois filhos
def crossover(pai1, pai2):
    tamanho = len(pai1)
    ponto1, ponto2 = sorted(random.sample(range(tamanho), 2))
    meio = pai1[ponto1:ponto2]
    filho1 = meio + [gene for gene in pai2 if gene not in meio]
    filho2 = meio + [gene for gene in pai1 if gene not in meio]
    return filho1, filho2

# Aplica mutação trocando duas cidades de posição
def mutacao(individuo):
    if random.random() < MUTACAO_RATE:
        i, j = random.sample(range(len(individuo)), 2)
individuo[i], individuo[j] = individuo[j], individuo[i]
    return individuo

# Imprime o percurso das cidades
def mostra_solucao(melhor_solucao, titulo):
    caminho = [0] + melhor_solucao + [0]
    plt.figure(figsize=(8, 8))
    plt.scatter(cidades[:, 0], cidades[:, 1], c='blue')
    plt.plot(cidades[caminho, 0], cidades[caminho, 1], 'r-', lw=1.5)
    plt.title(titulo)
    plt.show()

## Algoritmo e resultado

def main():

    populacao = criar_populacao()
    melhor_solucao = avaliar_populacao(populacao)[0]
    mostra_solucao(melhor_solucao, "Primeira melhor solução (população
inicial)")

    for _ in range(GERACOES):
        nova_populacao = []
        while len(nova_populacao) < POPULACAO_SIZE * CROSSOVER_RATE:
            pai1, pai2 = selecao_torneio(populacao),
selecao_torneio(populacao)
            filho1, filho2 = crossover(pai1, pai2)
            nova_populacao.extend([filho1, filho2])

        while len(nova_populacao) < POPULACAO_SIZE:
            nova_populacao.append(mutacao(selecao_torneio(populacao)))

        populacao = avaliar_populacao(nova_populacao)[:POPULACAO_SIZE]
        melhor_solucao = populacao[0]

    mostra_solucao(melhor_solucao, "Melhor solução após 1000 gerações")


main()
```


Primeira melhor solução (população inicial)
0
20
40
60
80
100

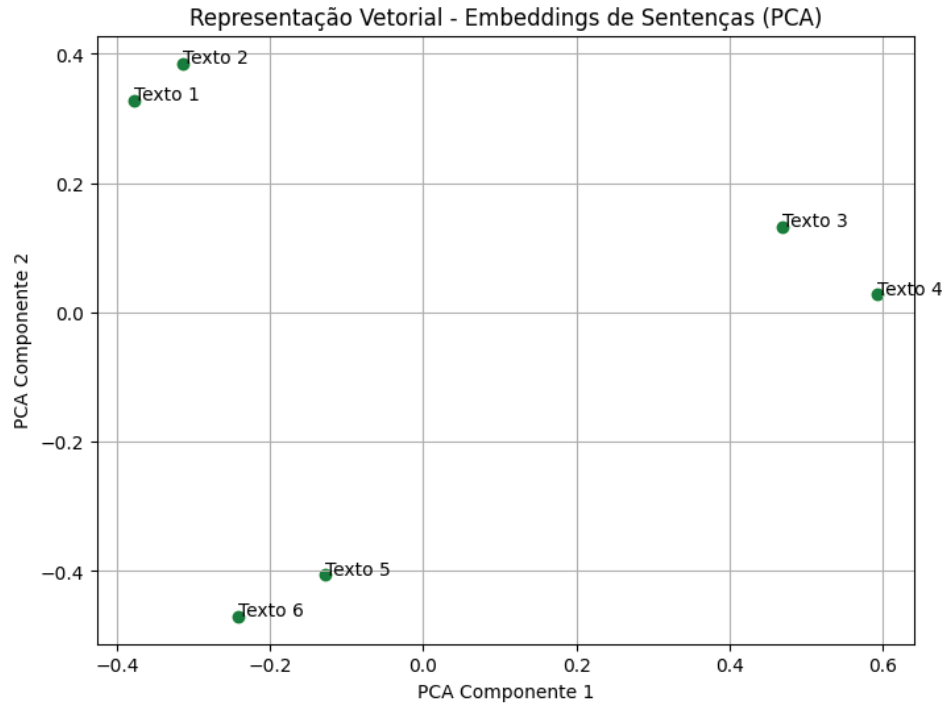

Melhor solução após 1000 gerações
0
20
40
60
80
100

```
## Bibliotecas

import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer
from sentence_transformers import SentenceTransformer

## Texto base


textos = [
    "O cachorro correu pelo parque atrás da bola",
    "O cão brincou no parque com sua bola favorita",
    "O gato dormia tranquilamente no sofá da sala",
    "O felino estava descansando calmamente no sofá",
    "Um pássaro voou rapidamente para a árvore mais alta",
    "O drone pousou cuidadosamente no topo da árvore"
]

## Modelo por frequência de palavras, com visualização usando PCA


vectorizer = CountVectorizer()
X_palavras = vectorizer.fit_transform(textos).toarray()
# PCA para visualização (reduzindo a 2 dimensões)
pca_palavras = PCA(n_components=2)
X_pca_palavras = pca_palavras.fit_transform(X_palavras)

# Plot da representação baseada em frequência de palavras
plt.figure(figsize=(8, 6))
plt.scatter(X_pca_palavras[:,0], X_pca_palavras[:,1], color='blue')

for i, texto in enumerate(textos):
    plt.annotate(f'Texto {i+1}', (X_pca_palavras[i,0],
X_pca_palavras[i,1]))

plt.title('Representação Vetorial - Frequência de Palavras (PCA)')
plt.xlabel('PCA Componente 1')
plt.ylabel('PCA Componente 2')
plt.grid(True)
plt.show()
```




```
### Matrix de similaridade usando coseno

# Similaridade por cosseno
sim_matrix = cosine_similarity(X_pca_palavras)

print("Matriz de similaridade por cosseno entre textos:\n")
print(sim_matrix)

### Avaliação dos resultados similares

pares_similares = np.argsort(-sim_matrix, axis=1)[:, 1]

print("\nTextos mais similares entre si:")
for idx, par in enumerate(pares_similares):
    print(f"Texto {idx+1} é mais próximo do Texto {par+1}")

Textos mais similares entre si:
Texto 1 é mais próximo do Texto 2
Texto 2 é mais próximo do Texto 1
Texto 3 é mais próximo do Texto 4
Texto 4 é mais próximo do Texto 3
Texto 5 é mais próximo do Texto 6
Texto 6 é mais próximo do Texto 4

## Modelo por embedding e visualização usando PCA

# Usando modelo de sentence transformer https://huggingface.co/sentence-
transformers/all-MiniLM-L6-v2
modelo_embed = SentenceTransformer('all-MiniLM-L6-v2')
X_embeddings = modelo_embed.encode(textos)

# PCA para visualização (reduzindo a 2 dimensões)
pca_embed = PCA(n_components=2)
X_pca_embed = pca_embed.fit_transform(X_embeddings)

# Plot da representação baseada em embeddings de sentenças
plt.figure(figsize=(8, 6))
plt.scatter(X_pca_embed[:,0], X_pca_embed[:,1], color='green')
```

```
for i, texto in enumerate(textos):
    plt.annotate(f'Texto {i+1}', (X_pca_embed[i,0], X_pca_embed[i,1]))

plt.title('Representação Vetorial - Embeddings de Sentenças (PCA)')
plt.xlabel('PCA Componente 1')
plt.ylabel('PCA Componente 2')
plt.grid(True)
plt.show()
```




```
### Matrix de similaridade usando coseno

# Similaridade por cosseno
sim_matrix = cosine_similarity(X_pca_embed)

print("Matriz de similaridade por cosseno entre textos:\n")
print(sim_matrix)

### Avaliação dos resultados similares
pares_similares = np.argsort(-sim_matrix, axis=1)[:, 1]

print("\nTextos mais similares entre si:")
for idx, par in enumerate(pares_similares):
    print(f"Texto {idx+1} é mais próximo do Texto {par+1}")


Textos mais similares entre si:
Texto 1 é mais próximo do Texto 2
Texto 2 é mais próximo do Texto 1
Texto 3 é mais próximo do Texto 4
Texto 4 é mais próximo do Texto 3
Texto 5 é mais próximo do Texto 6
Texto 6 é mais próximo do Texto 5
```