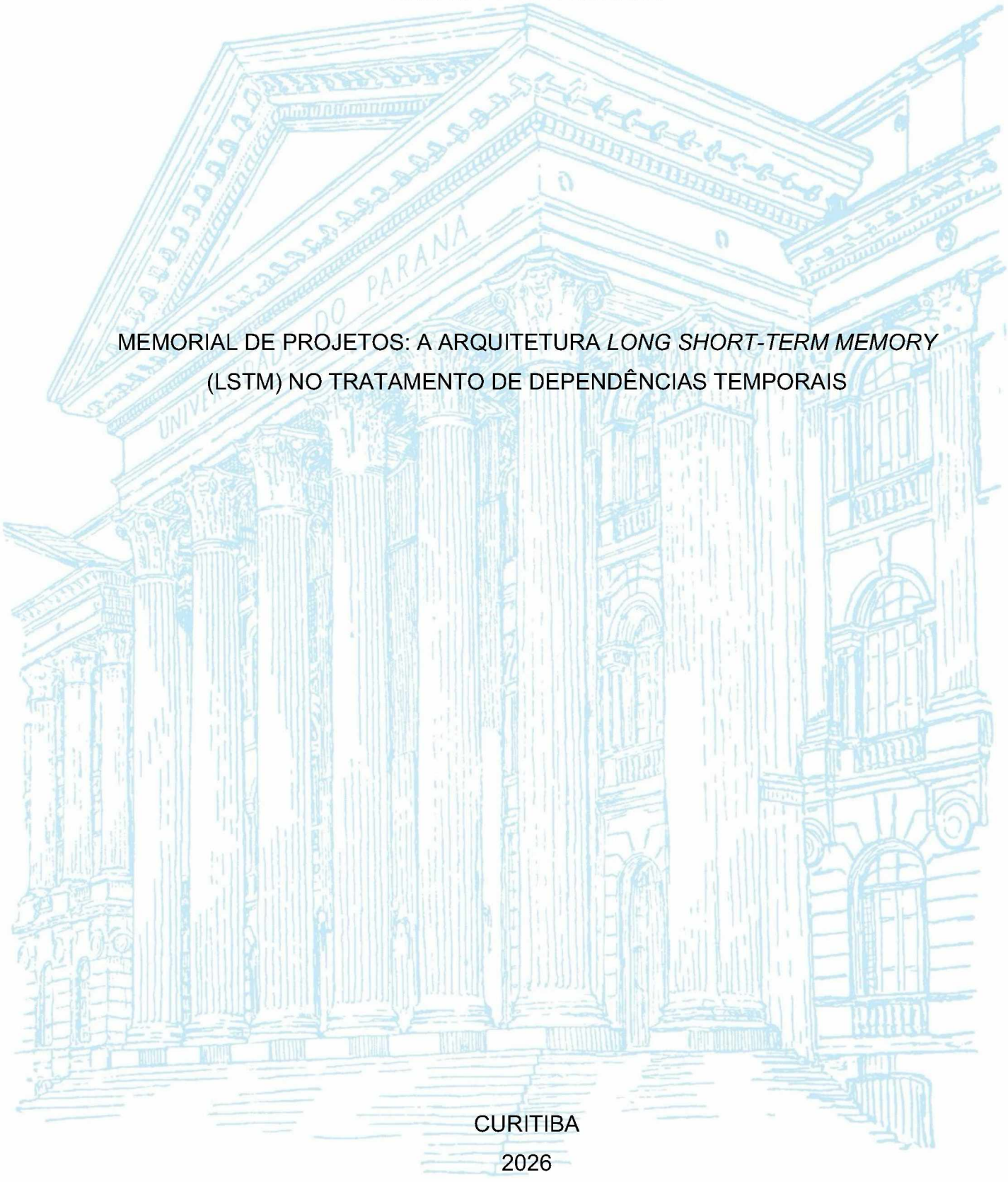
UNIVERSIDADE FEDERAL DO PARANÁ

BRUNO PIEPER BUNHAK



MEMORIAL DE PROJETOS: A ARQUITETURA *LONG SHORT-TERM MEMORY* (LSTM) NO TRATAMENTO DE DEPENDÊNCIAS TEMPORAIS

CURITIBA

2026

BRUNO PIEPER BUNHAK

MEMORIAL DE PROJETOS: A ARQUITETURA *LONG SHORT-TERM MEMORY* (LSTM) NO TRATAMENTO DE DEPENDÊNCIAS TEMPORAIS

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

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2026

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **BRUNO PIEPER BUNHAK**, intitulada: **MEMORIAL DE PROJETOS: A ARQUITETURA *LONG SHORT-TERM MEMORY* (LSTM) NO TRATAMENTO DE DEPENDÊNCIAS TEMPORAIS**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

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

Curitiba, 07 de Fevereiro de 2026.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAZER ANTHOM NIZER ROJAS MONTAÑO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

A arquitetura *Long Short-Term Memory* (LSTM), também conhecida como memória de longo e curto prazo, constitui uma extensão das redes neurais recorrentes desenvolvida com o propósito de mitigar os problemas de desvanecimento e explosão do gradiente no treinamento de sequências temporais longas. Este parecer técnico apresenta uma análise conceitual da LSTM, descrevendo seus fundamentos teóricos essenciais e sua relevância no processamento de dados sequenciais. São abordados os principais elementos da arquitetura, com destaque para a célula de memória e para os mecanismos de controle baseados em portas de entrada, esquecimento e saída, responsáveis pela regulação do fluxo de informações ao longo do tempo. Ressalta-se o papel do *constant error carousel* na preservação do gradiente e na aprendizagem de dependências temporais de longo alcance. O texto também discute as limitações da LSTM, incluindo a elevada complexidade computacional, a necessidade de grandes volumes de dados para treinamento adequado, as restrições de paralelização decorrentes do processamento sequencial e as dificuldades de generalização para valores extremos fora da distribuição de treinamento. Conclui-se que, embora amplamente utilizada, a adoção da LSTM deve considerar a disponibilidade de dados, os recursos computacionais e as características das dependências temporais envolvidas, sendo, em alguns cenários, recomendável a avaliação de arquiteturas alternativas.

Palavras-chave: memória de longo e curto prazo; redes neurais recorrentes; aprendizado profundo; dependências temporais; aprendizado sequencial.

ABSTRACT

The Long Short-Term Memory (LSTM) architecture is an extension of recurrent neural networks designed to address the vanishing and exploding gradient problems in the training of long temporal sequences. This technical report presents a conceptual analysis of LSTM, outlining its main theoretical foundations and its relevance in sequential data processing. The core components of the architecture are discussed, with emphasis on the memory cell and the gate-based control mechanisms, namely the input, forget, and output gates, which regulate the flow of information over time. The role of the constant error carousel in preserving gradients and enabling the learning of long-term temporal dependencies is highlighted. The report also examines the main limitations of LSTM, including high computational complexity, the requirement for large datasets for effective training, limited parallelization due to sequential processing, and challenges in generalizing to extreme values outside the training data distribution. It is concluded that, although widely adopted, the use of LSTM should be evaluated considering data availability, computational resources, and the nature of the temporal dependencies involved.

Keywords: long short-term memory; recurrent neural networks; deep learning; temporal dependencies; sequential learning.

SUMÁRIO

1 PARECER TÉCNICO

1.1 CONTEXTUALIZAÇÃO HISTÓRICA

As redes neurais recorrentes (RNNs) surgiram como uma abordagem promissora para a modelagem de dados sequenciais, permitindo que informações de passos temporais anteriores influenciem o processamento de entradas subsequentes. Entretanto, tais redes passaram a apresentar um problema fundamental durante o treinamento por retropropagação ao longo do tempo: o gradiente tendia a desaparecer ou explodir ao ser propagado por múltiplos passos temporais, impedindo o aprendizado eficiente de dependências de longo prazo (Hochreiter; Schmidhuber, 1997).

Em 1997, Sepp Hochreiter e Jürgen Schmidhuber propuseram a arquitetura *Long Short-Term Memory* (LSTM) como solução para esse desafio. Segundo os autores, "aprender a armazenar informações ao longo de intervalos de tempo estendidos por meio da retropropagação recorrente leva muito tempo, principalmente devido ao fluxo de erro insuficiente e decrescente durante a retropropagação." (Hochreiter; Schmidhuber, 1997, p. 1735). A LSTM foi projetada especificamente para manter um fluxo de erro constante através de um mecanismo denominado *constant error carousel*, permitindo que informações relevantes sejam preservadas ao longo de centenas ou milhares de passos temporais.

1.2 FUNDAMENTOS TEÓRICOS

A arquitetura LSTM distingue-se das redes recorrentes tradicionais pela presença de uma célula de memória (*cell state*) que mantém informações ao longo do tempo e de três portas principais que regulam o fluxo de informação: a porta de entrada (*input gate*), a porta de esquecimento (*forget gate*) e a porta de saída (*output gate*). Essa estrutura foi concebida para superar limitações estruturais das RNNs clássicas no tratamento de dependências temporais de longo prazo, particularmente aquelas associadas às dificuldades de treinamento em sequências extensas (Skansi, 2018).

A porta de entrada decide quais informações da entrada atual e do estado oculto anterior serão incorporadas à célula de memória. A porta de esquecimento

determina quais partes do estado anterior da célula devem ser descartadas, permitindo que a rede se adapte a mudanças no padrão dos dados. A porta de saída controla qual parte do estado da célula será exposta como saída no estado oculto atual. Essas portas operam de forma coordenada para garantir que informações relevantes sejam mantidas enquanto informações obsoletas sejam descartadas.

O mecanismo fundamental que permite à LSTM mitigar o problema do desvanecimento do gradiente é o *constant error carousel*. Este mecanismo mantém o gradiente aproximadamente constante ao longo de múltiplos passos temporais, evitando que ele se atenue exponencialmente. A célula de memória atua como um "carrossel" que preserva o erro, permitindo que a rede aprenda dependências que se estendem por longos intervalos temporais (Hochreiter; Schmidhuber, 1997).

O processo de atualização da célula de memória ocorre através da interação coordenada das três portas: a porta de esquecimento define quais informações previamente armazenadas devem ser mantidas ou descartadas, a porta de entrada seleciona quais novas informações serão incorporadas ao estado da memória, e a porta de saída regula como o conteúdo atual da célula será exposto à próxima camada da rede. Essa organização permite que a arquitetura aprenda, de forma adaptativa, quando reter, atualizar ou eliminar informações ao longo do tempo, em função do padrão dos dados. A utilização de mecanismos de controle baseados em portas contribui para o gerenciamento seletivo da memória e para a mitigação de limitações presentes em redes recorrentes tradicionais, resultando em melhor desempenho no aprendizado de dependências temporais de longo prazo (Goodfellow; Bengio; Courville, 2016).

1.3 LIMITAÇÕES E DESAFIOS

Apesar de seus avanços, a arquitetura LSTM apresenta limitações que devem ser consideradas em aplicações práticas. A presença de múltiplos mecanismos internos de controle aumenta a complexidade computacional do modelo, resultando em maior custo de treinamento, especialmente em sequências longas. Esse aumento de custo pode comprometer a eficiência e a escalabilidade do modelo em ambientes com restrições de recursos computacionais (Sejnowski, 2018).

A necessidade de grandes volumes de dados para treinamento adequado constitui outra limitação importante. A LSTM requer conjuntos de dados extensos para

aprender padrões significativos e generalizar adequadamente. Em domínios com dados limitados ou fragmentados, o desempenho da LSTM pode ser comprometido, levando a sobreajuste ou generalização insuficiente. Modelos LSTM profundos são particularmente suscetíveis a problemas de sobreajuste quando treinados com conjuntos de dados pequenos, exigindo técnicas de regularização avançadas para mitigar esse comportamento (Chollet, 2021).

A LSTM também apresenta limitações específicas na modelagem de valores extremos que estão fora da distribuição dos dados de treinamento. A saturação das funções de ativação sigmoide e tangente hiperbólica em estados de célula extremos limita a capacidade da rede de responder adequadamente a eventos raros ou extremos. Mesmo com ajustes arquiteturais ou ampliação do volume de dados, a generalização para extremos não é totalmente garantida, representando um desafio fundamental da arquitetura.

De modo geral, embora a arquitetura LSTM tenha representado um avanço significativo na modelagem de dependências temporais em dados sequenciais, ela apresenta limitações decorrentes de seu processamento inerentemente sequencial. Essa característica dificulta a paralelização das operações e aumenta o custo computacional do treinamento, especialmente em sequências longas. Além disso, mesmo com mecanismos projetados para preservar informações ao longo do tempo, a captura eficiente de dependências de longo alcance permanece um desafio prático, motivando a investigação de abordagens alternativas para o processamento de sequências complexas (Goodfellow; Bengio; Courville, 2016).

1.4 CONSIDERAÇÕES FINAIS

A arquitetura *Long Short-Term Memory* representa um marco importante no desenvolvimento de redes neurais recorrentes ao oferecer uma solução eficaz para o problema do desvanecimento do gradiente. Sua ampla adoção atesta sua utilidade prática. Contudo, o uso da LSTM deve ser avaliado considerando-se a disponibilidade de dados, os recursos computacionais e a natureza das dependências temporais. Em cenários com dados limitados ou dependências extremamente longas, arquiteturas alternativas ou modelos híbridos podem ser mais apropriados. Direções futuras incluem variantes mais eficientes, integração com mecanismos de atenção e modelos híbridos que combinam conhecimentos de domínio com aprendizado de máquina.

REFERÊNCIAS

CHOLLET, F. **Deep Learning with Python**. 2. ed. Shelter Island, NY: Manning Publications Co., 2021.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA: The MIT Press, 2016.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, nov. 1997.

SEJNOWSKI, T. J. **The deep learning revolution**. Cambridge, MA: The MIT Press, 2018.

SKANSI, S. **Introduction to deep learning: From logical calculus to artificial intelligence**. 1. ed. Cham, Suíça: Springer International Publishing, 2018.

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA

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.

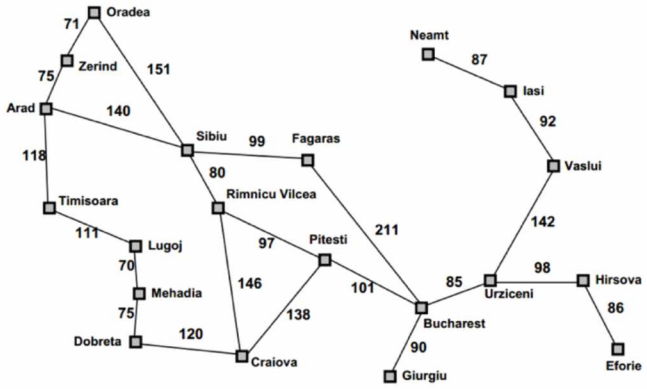
2 Busca Heurística

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

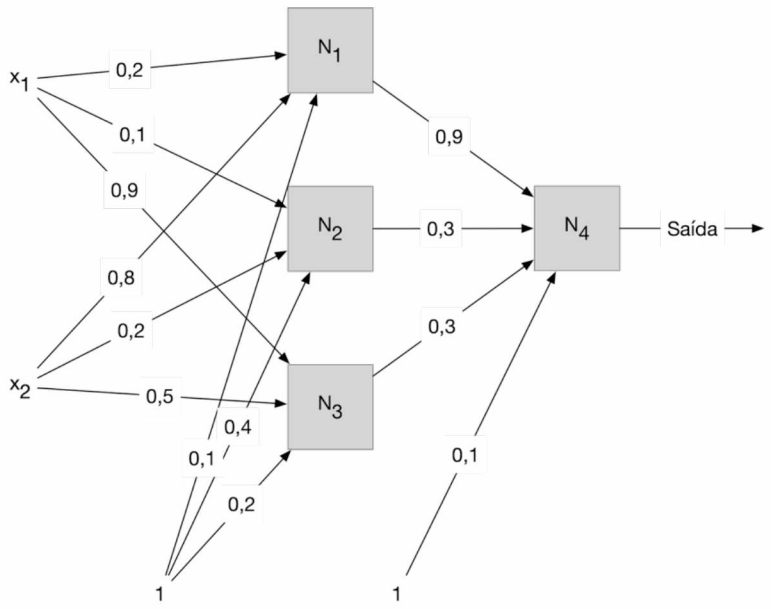
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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) (6,25 pontos) Pergunte ao ChatGPT o que é Inteligência Artificial e cole aqui o resultado.

Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e máquinas capazes de realizar tarefas que normalmente exigiriam inteligência humana. Essas tarefas incluem reconhecimento de fala, compreensão de linguagem natural, tomada de decisão, aprendizado, visão computacional, entre outras.

A IA busca criar algoritmos e modelos que possam simular certos aspectos da inteligência humana, como raciocínio, resolução de problemas, aprendizado e adaptação ao ambiente. Essa área abrange uma variedade de técnicas, incluindo aprendizado de máquina, redes neurais, processamento de linguagem natural, visão computacional e muito mais.

O objetivo final da inteligência artificial é criar sistemas que possam realizar tarefas de forma autônoma e inteligente, sem intervenção humana direta.

b) (6,25 pontos) Dada essa resposta do ChatGPT, classifique usando as 4 abordagens vistas em sala. Explique o porquê.

Agir como humanos:

A resposta dada pelo ChatGPT se enquadra bastante na abordagem “Agir como humanos”, ao citar o desenvolvimento de máquinas que executam funções que exigem inteligência humana, semelhante a definição de Kurzweil (1990). Também enquadra-se nesta abordagem ao mencionar a compreensão de linguagem natural e a aprendizagem.

Pensar como os humanos:

Apesar de não falar em implementar fielmente o processo de pensamento, a definição do ChatGPT se enquadra parcialmente na abordagem de "pensar como humanos" quando fala em "criar algoritmos e modelos que possam simular certos aspectos da inteligência humana, como raciocínio, resolução de problemas, aprendizado e adaptação ao ambiente", pois está alinhada com a definição de Bellman (1978), que aborda a IA como a automatização destes tipo de atividades.

Pensar racionalmente:

A definição do ChatGPT não se enquadra nesta abordagem, pois não menciona modelar o processo de raciocínio ou a busca de um resultado logicamente correto.

Agir racionalmente:

Por não mencionar a busca do melhor resultado possível, a definição dada pelo ChatGPT não se enquadra nesta abordagem.

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.

O ChatGPT é um modelo de linguagem de grande porte (LLM) desenvolvido pela OpenAI, treinado em um enorme conjunto de dados de texto e código. Essa ferramenta é capaz de gerar respostas textuais complexas e coerentes a partir de uma ampla gama de prompts e perguntas, abrindo um leque de possibilidades para interação humano-computador. Ele analisa a entrada feita pelo usuário, com base nessa informação ele ativa diferentes modelos neurais para lidar com diferentes tipos de problema.

A forma como o ChatGPT gera as respostas é, de certa forma, prevendo as próximas palavras a serem escritas. Através dos dados de treinamento, ele identificou padrões nos textos produzidos por humanos, e se tornou capaz de produzir novos textos similares. Com base no *prompt* do usuário (e no próprio texto que vai sendo gerado), o ChatGPT busca em seu "conhecimento" a direção para a qual o texto deve continuar.

Referências:

ChatGPT: o que é e como usar? Veja o guia completo do chatbot da OpenAI: https://www.techtudo.com.br/guia/2023/03/chatgpt-o-que-e-e-como-usar-veja-o-guia-completo-do-chatbot-da-openai-edsoftwares.ghtml

What Is ChatGPT Doing … and Why Does It Work?: https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

d) (6,25 pontos) Entendendo o que é o ChatGPT, classifique o próprio ChatGPT usando as 4 abordagens vistas em sala. Explique o porquê

Pensar como os humanos: Não se enquadra nesta abordagem, por não replicar processos cognitivos reais.

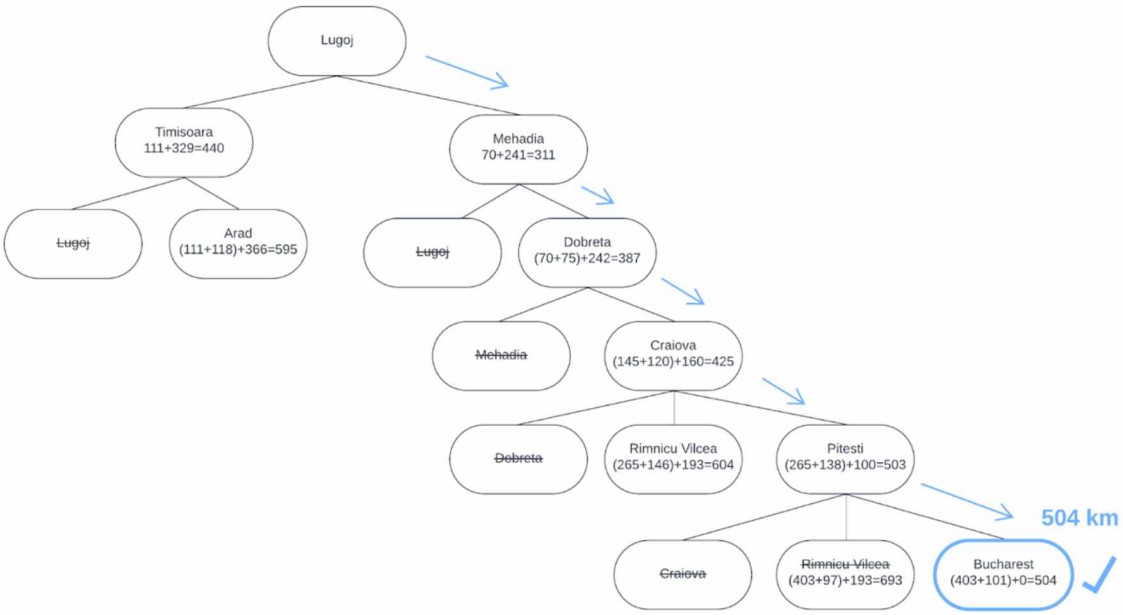
Pensar racionalmente: Utiliza algoritmos e modelos estatísticos para gerar respostas lógicas e coerentes com base em seu treinamento.

Agir como os humanos: Destaca-se aqui, gerando interações textuais indistinguíveis das humanas em muitos contextos.

Agir racionalmente: Como esta abordagem é mais ampla, e engloba técnicas das outras abordagens, o ChatGPT se enquadra nela, pois além da geração de textos bastante semelhantes aos de humanos e da utilização de modelos para gerar respostas lógicas, ele também busca maximizar relevância e utilidade das respostas, seguindo princípios de otimização.

2 - Busca Heurística

a)



3 - Lógica

a)

uvas caem: p

raposa as come: q

uvas maduras: r

Buscar: qp (A raposa come as uvas se e somente se as uvas caem)

P1: pq

P2: qr

P3: rp

P4: rp - Equivalência Implicação - P3

P5: qp - Silogismo Hipotético - P2, P4

P6: (qp)(pq) - Conjunção - P5, P1

P7: qp - Equivalência Bicondicional - P6

4 - Redes Neurais Artificiais

a)

N1 Entrada: (1 0,1)+(-30,2)+(10,8) =0,3

N1 Saída: 0,3

b)

N2 Entrada: (10,4)+(-30,1)+(10,2)=0,3

N2 Saída: 0,3

c)

N3 Entrada: (10,2)+(-30,9)+(10,5)=-2

N3 Saída: -2

d)

N4 Entrada: (10,1)+(0,30,9)+(0,30,3)+(-20,3)=-0,14

N4 Saída: tanh(-0,14)= -0,1391

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

a)

Código

```
dados = pd.read_csv("precos_carros_brasil.csv", low_memory=False)
```

b)

Código

```
dados.isna().any()
```

Resultado

```
year_of_reference     True
month_of_reference    True
fipe_code             True
authentication        True
brand                 True
model                 True
fuel                  True
gear                  True
engine_size           True
year_model            True
avg_price_brl         True
dtype: bool
```

Código

```
dados.dropna(inplace=True)
```

c)

Código

```
dados.duplicated().sum()
```

Resultado

```
2
```

Código

```
dados.drop_duplicates(inplace=True)
```

d)

Código

```
numericas_cols = [col for col in dados.columns if dados[col].dtype !=
"object"]
categoricas_cols = [col for col in dados.columns if dados[col].dtype
== "object"]
dados[numericas_cols].describe()
```

Resultado

```
   year_of_reference     year_model  avg_price_brl
count      202295.000000  202295.000000  202295.000000
mean         2021.564695    2011.271514   52756.765713
std             0.571904       6.376241   51628.912116
min          2021.000000    2000.000000    6647.000000
25%          2021.000000    2006.000000   22855.000000
50%          2022.000000    2012.000000   38027.000000
75%          2022.000000    2016.000000   64064.000000
max          2023.000000    2023.000000  979358.000000
```

e)

Código

```
dados["model"].value_counts()
```

Resultado

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

Código

```
dados["brand"].value_counts()
```

Resultado

```
brand
Fiat               44962
VW - VolksWagen    44312
GM - Chevrolet     38590
Ford               33150
Renault            29191
Nissan             12090
Name: count, dtype: int64
```

f)

Foram identificadas 6 marcas de carros e 2112 modelos diferentes. A base de dados inicial contem 65.245 valores NaN 2 valores duplicados, apos o tratamento

desses dados a base de dados contem 202.295 linhas. É notável que a fabricante que mais tem veículos na base é a Fiat, seguida da VW, e GM em terceiro lugar. A fabricante com a menor quantidade é a Nissan.

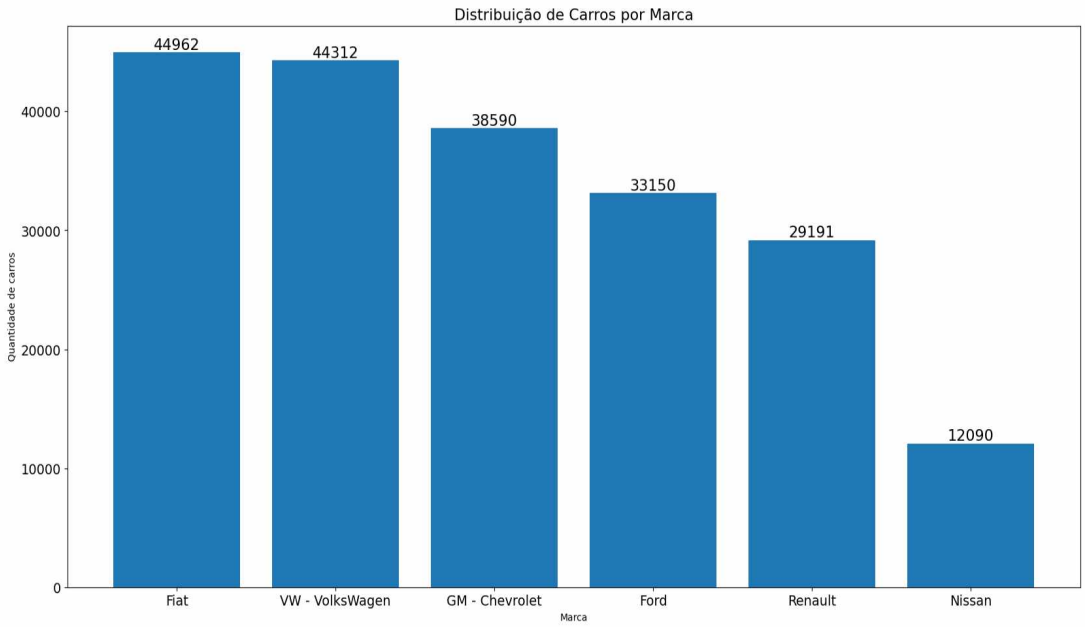
2 Visualização dos dados

a)

Código

```
plt.figure(figsize=(20, 10))
grafico_distribuicao_marca = plt.bar(
    dados["brand"].value_counts().index,
dados["brand"].value_counts()
)
plt.title("Distribuição de Carros por Marca", fontsize=16)
plt.ylabel("Quantidade de carros")
plt.xlabel("Marca")
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.bar_label(grafico_distribuicao_marca, label_type="edge", size=16)
plt.show()
```

Resultado



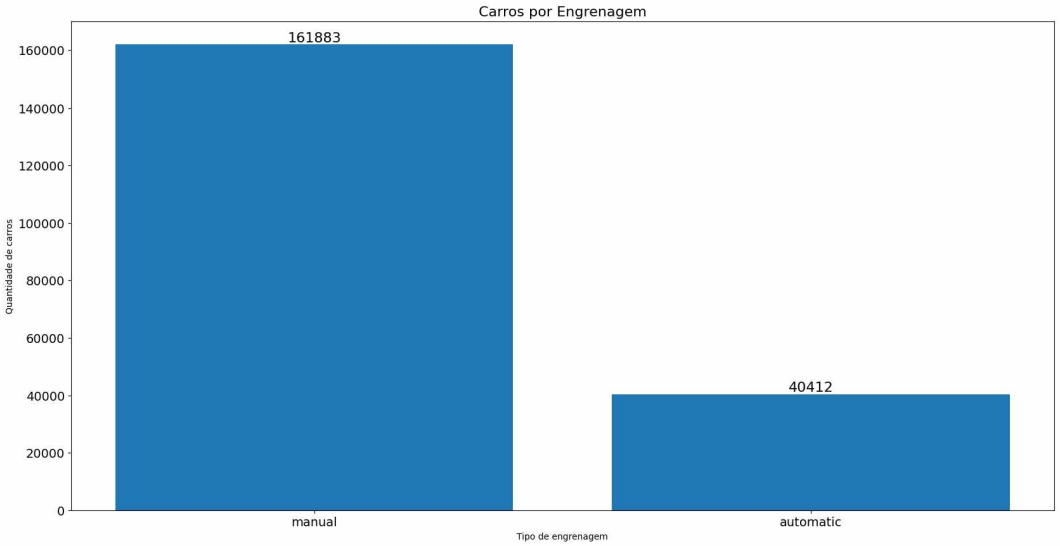
b)

Código

```
plt.figure(figsize=(20, 10))
grafico_distribuicao_engrenagem = plt.bar(
    dados["gear"].unique(), dados["gear"].value_counts()
)
plt.title("Carros por Engrenagem", fontsize=16)
```

```
plt.ylabel("Quantidade de carros")
plt.xlabel("Tipo de engrenagem")
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.bar_label(grafico_distribuicao_engrenagem, label_type="edge",
size=16)
plt.show()
```

Resultado



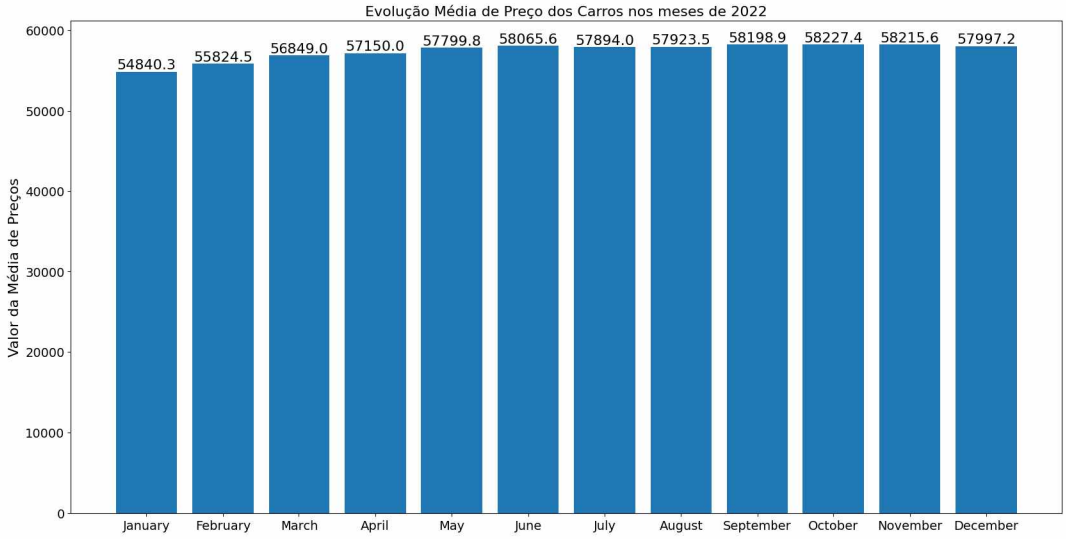
c)

Código

```
dados_2022 = dados.loc[dados["year_of_reference"] == 2022]

media_preco_mes = (
    dados_2022.groupby(["month_of_reference"])["avg_price_brl"]
    .mean()
    .reset_index(name="mean_avg_price_brl")
)

media_preco_mes["month_of_reference"] = pd.Categorical(
    media_preco_mes["month_of_reference"],
    categories=[
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December",
    ],
    ordered=True,
)
```

```
media_preco_mes.sort_values("month_of_reference", inplace=True,
ignore_index=True)

media_preco_mes

plt.figure(figsize=(20, 10))
grafico_media_preco_mes_2022_bara = plt.bar(
    media_preco_mes["month_of_reference"],
media_preco_mes["mean_avg_price_brl"]
)
plt.title("Evolução Média de Preço dos Carros nos meses de 2022",
fontsize=16)
plt.ylabel("Valor da Média de Preços", fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.bar_label(
    grafico_media_preco_mes_2022_bara, fmt="%.01f", size=16,
label_type="edge"
)
plt.show()
```

Resultado



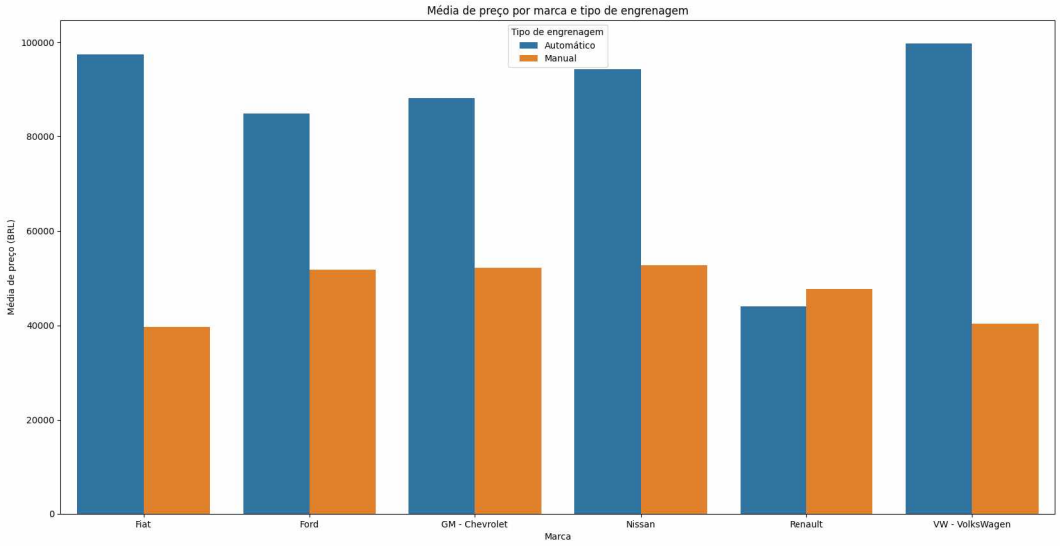
d)

Código

```
preco_por_maraca_engrenagem = (
    dados.groupby(["brand", "gear"])["avg_price_brl"]
    .mean()
    .reset_index(name="mean_avg_price_brl")
)

grafico_preco_por_maraca_engrenagem = sns.barplot(
    data=preco_por_maraca_engrenagem,
    x="brand",
    y="mean_avg_price_brl",
    hue="gear",
)
grafico_preco_por_maraca_engrenagem.set_title(
    "Média de preço por marca e tipo de engrenagem"
)
```

```
grafico_preco_por_maraca_engrenagem.set_ylabel("Média de preço
(BRL)")
grafico_preco_por_maraca_engrenagem.set_xlabel("Marca")
grafico_preco_por_maraca_engrenagem.legend(

grafico_preco_por_maraca_engrenagem.get_legend_handles_labels()[0],
    ["Automático", "Manual"],
    title="Tipo de engrenagem",
)
grafico_preco_por_maraca_engrenagem.figure.set_size_inches(20, 10)
```

Resultado



e)

Os preços de carros automáticos são, em geral, maiores que os preços de carros manuais. A única marca onde isso não ocorreu foi a Renault, onde a média dos preços é semelhante, inclusive sendo um pouco mais elevada nos carros manuais.

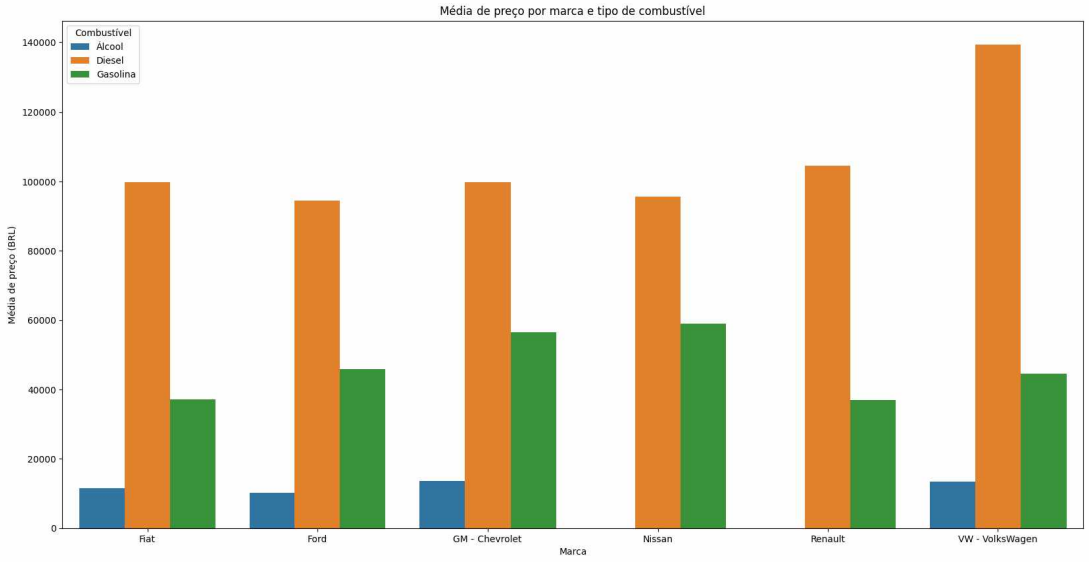
f)

Código

```
preco_por_maraca_combustivel = (
    dados.groupby(["brand", "fuel"])["avg_price_brl"]
    .mean()
    .reset_index(name="mean_avg_price_brl")
)

grafico_preco_por_maraca_combustivel = sns.barplot(
    data=preco_por_maraca_combustivel,
    x="brand",
    y="mean_avg_price_brl",
    hue="fuel",
)
grafico_preco_por_maraca_combustivel.set_title(
    "Média de preço por marca e tipo de combustível"
)
grafico_preco_por_maraca_combustivel.set_ylabel("Média de preço
(BRL)")
```

```
grafico_preco_por_maraca_combustivel.set_xlabel("Marca")
grafico_preco_por_maraca_combustivel.legend(

grafico_preco_por_maraca_combustivel.get_legend_handles_labels()[0],
    ["Álcool", "Diesel", "Gasolina"],
    title="Combustível",
)
grafico_preco_por_maraca_combustivel.figure.set_size_inches(20, 10)
```

Resultado



g)

Os veículos movidos a diesel apresentam um preço médio mais elevado do que os demais tipos de combustíveis. Os veículos movidos a álcool apresentam um preço médio menor do que os demais tipos de combustíveis. Os veículos movidos a gasolina apresentam um preço médio entre os veículos movidos a diesel e a álcool.

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

a)

Código

```
dados_num = dados.filter(
    [
        "brand",
        "model",
        "gear",
        "fuel",
        "engine_size",
        "year_model",
        "year_of_reference",
```

```
        "month_of_reference",
        "avg_price_brl",
    ]
)
dados_num["brand"] = LabelEncoder().fit_transform(dados_num["brand"])
dados_num["model"] = LabelEncoder().fit_transform(dados_num["model"])
dados_num["gear"] = LabelEncoder().fit_transform(dados_num["gear"])
dados_num["fuel"] = LabelEncoder().fit_transform(dados_num["fuel"])
dados_num["month_of_reference"] = LabelEncoder().fit_transform(
    dados_num["month_of_reference"]
)
dados_num["engine_size"] = dados["engine_size"].str.replace(",",
".").astype(float)

x = dados_num.drop("avg_price_brl", axis=1)
y = dados_num["avg_price_brl"]

y.head()
```

Resultado

```
0     9162.0
1     8832.0
2     8388.0
3     8453.0
4    12525.0
Name: avg_price_brl, dtype: float64
```

b)

Código

```
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.25, random_state=42
)
```

c)

Código

```
model_rf = RandomForestRegressor()
model_rf.fit(x_train, y_train)

model_rf = RandomForestRegressor()
model_rf.fit(x_train, y_train)
```

d)

Código

```
valores_preditos_rf = model_rf.predict(x_test)
valores_preditos_xgb = model_xgb.predict(x_test)
```

e)

Código

```
feature_importance_rf = pd.DataFrame(
    model_rf.feature_importances_, index=x_train.columns,
columns=["importance"]
```

```
).sort_values("importance", ascending=False)
feature_importance_rf
```

Resultado

```
importance
engine_size           0.449773
year_model            0.390370
model                 0.059092
gear                  0.033169
fuel                  0.032056
brand                 0.017997
year_of_reference     0.012330
month_of_reference    0.005213
```

Código

```
feature_importance_xgb = pd.DataFrame(
    model_xgb.feature_importances_, index=x_train.columns,
columns=["importance"]
).sort_values("importance", ascending=False)
feature_importance_xgb
```

Resultado

```
importance
engine_size           0.438108
year_model            0.200562
fuel                  0.146407
gear                  0.113473
brand                 0.055805
model                 0.022404
year_of_reference     0.018362
month_of_reference    0.004878
```

f)

O ano do carro (year_model) e o tamanho do motor (engine_size) se mostraram as variáveis mais importantes na predição do valor dos automóveis. O tipo de combustível acabou não se mostrando tão importante, o que é curioso pois a média de preço dos veículos a diesel era superior às outras. O ano e mes de referência dos dados foram as variáveis de menos e importância.

g)

Código

```
mse_rf = mean_squared_error(y_test, valores_preditos_rf)
mse_rf
```

Resultado

```
11681843.441155115
```

Código

```
mae_rf = mean_absolute_error(y_test, valores_preditos_rf)
mae_rf
```

Resultado

```
1732.1978164799061
```

Código

```
r2_rf = r2_score(y_test, valores_preditos_rf)
r2_rf
```

Resultado

```
0.995659336124847
```

Código

```
mse_xgb = mean_squared_error(y_test, valores_preditos_xgb)
mse_xgb
```

Resultado

```
29555221.170828193
```

Código

```
mae_xgb = mean_absolute_error(y_test, valores_preditos_xgb)
mae_xgb
```

Resultado

```
3173.461879807312
```

Código

```
r2_xgb = r2_score(y_test, valores_preditos_xgb)
r2_xgb
```

Resultado

```
0.98901806196046
```

h)

O modelo Random Forest apresentou melhor desempenho em todas as métricas avaliadas: menor MSE (11.681.843,44), menor MAE (1.732,12) e R^2 mais próximo de 1 (0,9957). Portanto, o Random Forest teve um desempenho superior ao XGBoost na previsão com base nas métricas utilizadas.

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

    - O modelo alométrico é dado por: Volume = b0 + b1 * dap^2 * Ht

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

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

    - Coeficiente de determinação: R^2

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

onde y_i é o valor observado, $\widehat{y_i}$ é o valor predito e $\bar{y}$ 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

Pesquisa com Dados de Satélite (Satellite)

Verificando as matrizes de confusão, o melhor resultado foi obtido com o método SVM, com 87,69% de precisão.

Código

```
# carregar pacotes
library("mlbench")
library("caret")

# 1. Carregue a base de dados Satellite
data("Satellite")
Satellite <- Satellite[c("x.17", "x.18", "x.19", "x.20", "classes")]

# 2. Crie partições contendo 80% para treino e 20% para teste
indices <- createDataPartition(Satellite$classes, p=0.8, list=FALSE)
treino <- Satellite[indices,]
teste <- Satellite[-indices, ]

# 3. Treine modelos RandomForest, SVM e RNA para predição destes
dados.
rf <- train(classes~., data=treino, method="rf")
svm <- train(classes~., data=treino, method="svmRadial")
rna <- train(classes~., data=treino, method="nnet")

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.
matrizesConfusao.rf <- confusionMatrix(predicoes.rf, teste$classes)
matrizesConfusao.svm <- confusionMatrix(predicoes.svm, teste$classes)
matrizesConfusao.rna <- confusionMatrix(predicoes.rna, teste$classes)

matrizesConfusao.rf
matrizesConfusao.svm
matrizesConfusao.rna
```

Resultado

```
> matrizesConfusao.rf
Confusion Matrix and Statistics

                     Reference
Prediction            red soil cotton crop grey soil damp grey soil
vegetation stubble very damp grey soil
  red soil                 296           1         5              2
8                    0
  cotton crop                0         123         0              0
2                    1
  grey soil                  3           0       238             29
1                   10
  damp grey soil             1           0        21             62
2                   40
  vegetation stubble         6          10         0              0
120                    8
  very damp grey soil        0           6         7             32
8                   242

Overall Statistics

               Accuracy : 0.8419
                 95% CI : (0.8208, 0.8614)
    No Information Rate : 0.2383
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8047

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: red soil Class: cotton crop Class: grey
soil Class: damp grey soil Class: vegetation stubble
Sensitivity                   0.9673            0.87857
0.8782               0.49600                   0.85106
Specificity                   0.9836            0.99738
0.9576               0.94478                   0.97900
Pos Pred Value                0.9487            0.97619
0.8470               0.49206                   0.83333
Neg Pred Value                0.9897            0.98532
0.9671               0.94560                   0.98158
Prevalence                    0.2383            0.10903
0.2111               0.09735                   0.10981
Detection Rate                0.2305            0.09579
0.1854               0.04829                   0.09346
```

```
Detection Prevalence          0.2430            0.09813
0.2188               0.09813                   0.11215
Balanced Accuracy             0.9755            0.93797
0.9179               0.72039                   0.91503
                     Class: very damp grey soil
Sensitivity                                0.8040
Specificity                                0.9461
Pos Pred Value                             0.8203
Neg Pred Value                             0.9403
Prevalence                                 0.2344
Detection Rate                             0.1885
Detection Prevalence                       0.2298
Balanced Accuracy                          0.8750

> matrizesConfusao.svm
Confusion Matrix and Statistics

                     Reference
Prediction            red soil cotton crop grey soil damp grey soil
vegetation stubble very damp grey soil
  red soil                 298           1         4               2
7                    0
  cotton crop                1         120         0               0
4                    0
  grey soil                  4           0       260              29
1                   12
  damp grey soil             0           1         7              69
2                   32
  vegetation stubble         3          14         0               2
117                  3
  very damp grey soil        0           4         0              23
10                 254

Overall Statistics

               Accuracy : 0.8707
                 95% CI : (0.8511, 0.8886)
    No Information Rate : 0.2383
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8399

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: red soil Class: cotton crop Class: grey
soil Class: damp grey soil Class: vegetation stubble
Sensitivity                   0.9739            0.85714
0.9594               0.55200                   0.82979
Specificity                   0.9857            0.99563
0.9546               0.96376                   0.98075
Pos Pred Value                0.9551            0.96000
0.8497               0.62162                   0.84173
Neg Pred Value                0.9918            0.98274
0.9888               0.95226                   0.97904
Prevalence                    0.2383            0.10903
0.2111               0.09735                   0.10981
Detection Rate                0.2321            0.09346
0.2025               0.05374                   0.09112
```

```
Detection Prevalence          0.2430            0.09735
0.2383               0.08645                  0.10826
Balanced Accuracy             0.9798            0.92639
0.9570               0.75788                  0.90527
                     Class: very damp grey soil
Sensitivity                               0.8439
Specificity                               0.9624
Pos Pred Value                            0.8729
Neg Pred Value                            0.9527
Prevalence                                0.2344
Detection Rate                            0.1978
Detection Prevalence                      0.2266
Balanced Accuracy                         0.9031

> matrizesConfusao.rna
Confusion Matrix and Statistics

                     Reference
Prediction            red soil cotton crop grey soil damp grey soil
vegetation stubble very damp grey soil
  red soil                 291           5         3               1
15                    0
  cotton crop                7         126         0               0
50                    0
  grey soil                  6           0       266              61
1                    23
  damp grey soil             0           0         0               0
0                     1
  vegetation stubble         1           2         0               0
34                    0
  very damp grey soil        1           7         2              63
41                   277

Overall Statistics

               Accuracy : 0.7741
                 95% CI : (0.7503, 0.7968)
    No Information Rate : 0.2383
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7151

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: red soil Class: cotton crop Class: grey
soil Class: damp grey soil Class: vegetation stubble
Sensitivity                   0.9510            0.90000
0.9815              0.0000000                  0.24113
Specificity                   0.9755            0.95017
0.9102              0.9991372                  0.99738
Pos Pred Value                0.9238            0.68852
0.7451              0.0000000                  0.91892
Neg Pred Value                0.9845            0.98728
0.9946              0.9025721                  0.91419
Prevalence                    0.2383            0.10903
0.2111              0.0973520                  0.10981
Detection Rate                0.2266            0.09813
0.2072              0.0000000                  0.02648
```

```
Detection Prevalence          0.2453            0.14252
0.2780             0.0007788                    0.02882
Balanced Accuracy             0.9632            0.92509
0.9459             0.4995686                    0.61926
                     Class: very damp grey soil
Sensitivity                                0.9203
Specificity                                0.8840
Pos Pred Value                             0.7084
Neg Pred Value                             0.9731
Prevalence                                 0.2344
Detection Rate                             0.2157
Detection Prevalence                       0.3045
Balanced Accuracy                          0.9021
```

.

Estimativa de Volumes de Árvores

O melhor resultado foi obtido com o modelo RandomForest, isso pôde ser observado nas três métricas utilizadas. Seu R^2 foi de 0.85 (sendo o mais próximo do valor "1" entre os modelos utilizados), o Syx foi de 0.1445, e o Syx% de 11.07 (o mais próximo de 0 entre os modelos).

Obs.: durante a análise dos resultados, foi verificado que os dados preditos pelo método nnet foram iguais para todos os registros. Provavelmente seria necessário um tratamento dos dados para melhor funcionamento com este método.

Código

```
# carregar pacotes
library("caret")

# 1. Carregar o arquivo Volumes.csv
(http://www.razer.net.br/datasets/Volumes.csv)
df <- read.csv("http://www.razer.net.br/datasets/Volumes.csv",
sep=";", dec=",")

# 2. Eliminar a coluna NR, que só apresenta um número sequencial
df$NR <- NULL

# 3. Criar partição de dados: treinamento 80%, teste 20%
indices <- createDataPartition(df$VOL, p=0.8, list=FALSE)
treino <- df[indices,]
teste <- df[-indices, ]

# 4. Usando o pacote "caret", treinar os modelos: Random Forest (rf),
SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de
SPURR
rf <- train(VOL~., data=treino, method="rf")
svm <- train(VOL~., data=treino, method="svmRadial")
rna <- caret::train(VOL~., data=treino, method="nnet")
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, treino, 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
# Coeficiente de determinação: R²
func_r2 <- function(predicoes, observacoes) {
  res <- sum( (observacoes - predicoes) ^ 2  )
  tot <- sum( (observacoes - mean(observacoes)) ^2 )
  return (1 - ( res / tot ))
}


# Erro padrão da estimativa: Syx
func_Syx <- function(predicoes, observacoes) {
  res <- sum( (observacoes - predicoes) ^ 2 )
  gl <- length(predicoes) - 2
  return (sqrt (res / gl) )
}

# Syx%:
func_Syx_p <- function(predicoes, observacoes) {
  syx <- func_Syx(predicoes, observacoes)
  return ( (syx / mean(observacoes)) * 100 )
}

# 7. Escolha o melhor modelo.
r2.rf <- func_r2(predicoes.rf, teste$VOL)
r2.svm <- func_r2(predicoes.svm, teste$VOL)
r2.rna <- func_r2(predicoes.rna, teste$VOL)
r2.alom <- func_r2(predicoes.alom, teste$VOL)
cat("R² RF: ", r2.rf, "\n")
cat("R² SVM: ", r2.svm, "\n")
cat("R² RNA: ", r2.rna, "\n")
cat("R² ALOM: ", r2.alom, "\n")
# O R² da rna ficou negativo, verificando os dados preditos,
verificamos que todos os valores foram gerados como "1". Ao pesquisar
o assunto, a normalização dos dados poderia corrigir essa situação.
predicoes.rna

syx.rf <- func_Syx(predicoes.rf, teste$VOL)
syx.svm <- func_Syx(predicoes.svm, teste$VOL)
syx.rna <- func_Syx(predicoes.rna, teste$VOL)
syx.alom <- func_Syx(predicoes.alom, teste$VOL)
cat("Syx RF: ", syx.rf, "\n")
cat("Syx SVM: ", syx.svm, "\n")
cat("Syx RNA: ", syx.rna, "\n")
cat("Syx ALOM: ", syx.alom, "\n")

syxp.rf <- func_Syx_p(predicoes.rf, teste$VOL)
syxp.svm <- func_Syx_p(predicoes.svm, teste$VOL)
syxp.rna <- func_Syx_p(predicoes.rna, teste$VOL)
syxp.alom <- func_Syx_p(predicoes.alom, teste$VOL)
cat("Syx% RF: ", syxp.rf, "\n")
cat("Syx% SVM: ", syxp.svm, "\n")
cat("Syx% RNA: ", syxp.rna, "\n")
cat("Syx% ALOM: ", syxp.alom, "\n")
```

Resultado

```
> cat("R² RF: ", r2.rf, "\n")
R² RF:  0.8580739
> cat("R² SVM: ", r2.svm, "\n")
R² SVM:  0.7963183
> cat("R² RNA: ", r2.rna, "\n")
R² RNA:  -0.7244946
> cat("R² ALOM: ", r2.alom, "\n")
R² ALOM:  0.8263134

> predicoes.rna
 1  2  7  9 14 19 20 27 32 37 38 40 42 44 46 49 54 58 77 98
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1

> cat("Syx RF: ", syx.rf, "\n")
Syx RF:  0.1423097
> cat("Syx SVM: ", syx.svm, "\n")
Syx SVM:  0.1704823
> cat("Syx RNA: ", syx.rna, "\n")
Syx RNA:  0.49606
> cat("Syx ALOM: ", syx.alom, "\n")
Syx ALOM:  0.1574296
```

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

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

B – RESOLUÇÃO

1 Gráficos e tabelas

Código

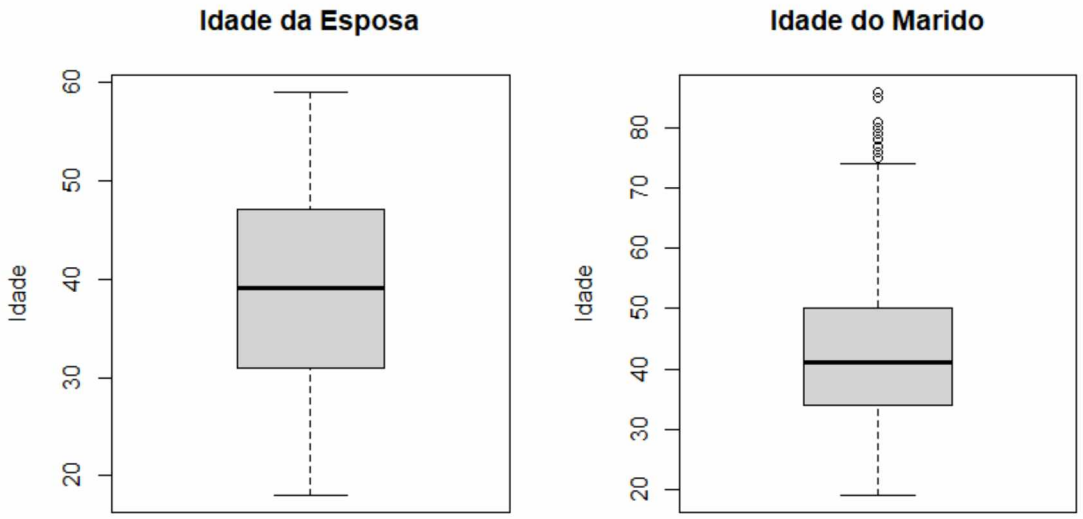
```
install.packages("modeest")
library(dplyr)
library(modeest)
```

```
load('salarios.RData')
```

Box-Plot

Código

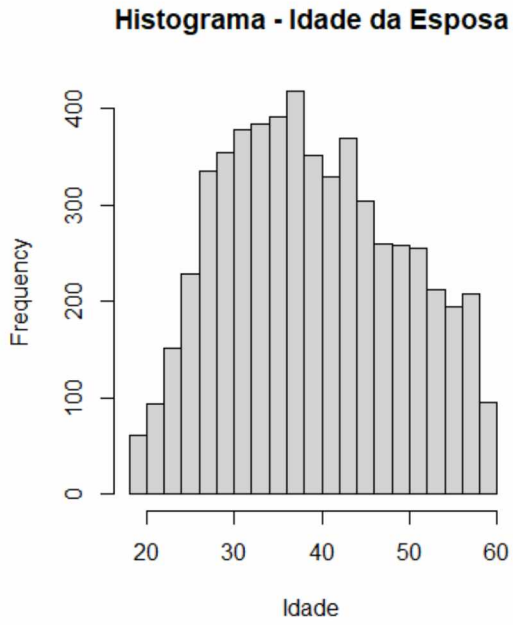
```
> boxplot(salarios$age, main = "Idade da Esposa", ylab = "Idade")
> boxplot(salarios$husage, main = "Idade do Marido", ylab = "Idade")
```

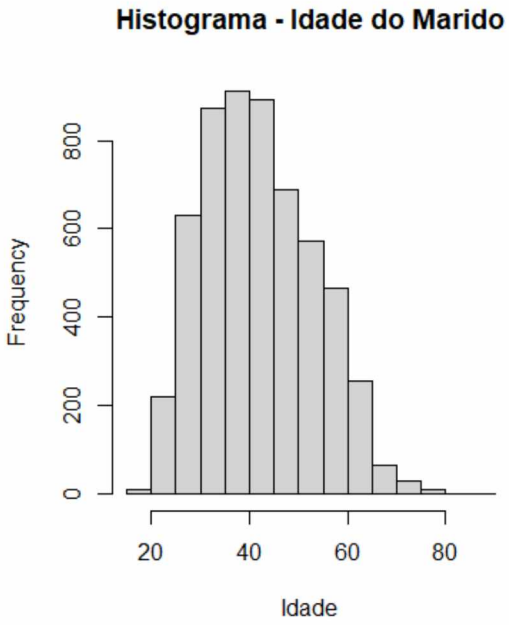


Histograma

Código

```
> hist(salarios$age, main = "Histograma - Idade da Esposa", xlab =
"Idade", breaks = 20)
> hist(salarios$husage, main = "Histograma - Idade do Marido", xlab =
"Idade", breaks = 20)
```





Analisando os gráficos gerados, podemos verificar, que as idades dos maridos (husage) se distribuem em uma faixa maior de valores, com alguns registros chegando na casa dos 80 anos. Também podemos verificar, no box-plot, que a mediana das duas variáveis está próxima de 40.

Tabela de frequências - “age”

Código

```
> table_age <- fdt(salarios$age)
> print(table_age)
```

Resultado

```
    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 frequências - “husage”

Código

```
> table_husage <- fdt(salarios$husage)
> print(table_husage)
```

Resultado

```
    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
```

2 Medidas de posição e dispersão

Código

```
> mean_age <- mean(salarios$age)
> median_age <- median(salarios$age)
> mode_age <- mfv(salarios$age)
> cat("Idade da Esposa:", "\n",
+     "Média: ", mean_age, "\n",
+     "Mediana: ", median_age, "\n",
+     "Moda: ", mode_age, "\n")
```

Resultado

```
Idade da Esposa:
 Média:  39.42758
 Mediana:  39
 Moda:  37
```

Código

```
> mean_husage <- mean(salarios$husage)
> median_husage <- median(salarios$husage)
> mode_husage <- mfv(salarios$husage)
> cat("Idade do Marido:", "\n",
+     "Média: ", mean_husage, "\n",
+     "Mediana: ", median_husage, "\n",
+     "Moda: ", mode_husage, "\n")
```

Resultado

```
Idade do Marido:
 Média:  42.45296
 Mediana:  41
 Moda:  44
```

Código

```
> mean_husage / mean_age
```

Resultado

```
[1] 1.076733
```

A idade média dos maridos é 7,67% superior à das esposas.

Código

```
> median_husage / median_age
```

Resultado

```
[1] 1.051282
```

A mediana da idade dos maridos é 5,12% superior à das esposas

Código

```
> mode_husage / mode_age
```

Resultado

```
[1] 1.189189
```

A moda da idade dos maridos é 18,92% superior à das esposas.

Código

```
> variance_age <- var(salarios$age)
> sd_age <- sd(salarios$age)
> cv_age <- sd_age / mean_age * 100
> variance_husage <- var(salarios$husage)
> sd_husage <- sd(salarios$husage)
> cv_husage <- sd_husage / mean_husage * 100
> cat("Idade da Esposa:\n")
> cat("Variância:", variance_age, "\n")
> cat("Desvio Padrão:", sd_age, "\n")
> cat("Coeficiente de Variação:", cv_age, "%\n\n")
> cat("Idade do Marido:\n")
> cat("Variância:", variance_husage, "\n")
> cat("Desvio Padrão:", sd_husage, "\n")
> cat("Coeficiente de Variação:", cv_husage, "%\n")
```

Resultado

```
Idade da Esposa:
Variância: 99.75234
Desvio Padrão: 9.98761
Coeficiente de Variação: 25.33153 %

Idade do Marido:
```

```
Variância: 126.0717
Desvio Padrão: 11.22817
Coeficiente de Variação: 26.44849 %
```

Código

```
> ((variance_husage / variance_age)-1) * 100
```

Resultado

```
[1] 26.38473
```

Portanto, a variância da idade dos maridos é 26,38% superior à da idade das esposas

Código

```
> ((sd_husage / sd_age)-1) * 100
```

Resultado

```
[1] 12.42096
```

O desvio padrão da idade dos maridos é 12,42% superior ao desvio padrão da idade das esposas.

Quanto aos coeficientes de variação, que ficaram em 25,33% e 26,45%, podemos dizer que existe média dispersão entre os valores destas variáveis nessa base de dados.

3 Testes paramétricos ou não paramétricos

Código

```
# Vamos fazer checagens preliminares para verificar as exigencias
# do teste: Amostras independentes, normalidade e homogeneidade
# das variancias entre grupos

# Premissa 1: As duas amostras sao independentes?
# Sim, pois os são dados dos maridos e esposas, e não dados do mesmo
indivíduo em dois momentos.
# Nao se trata de uma amostra ou grupos emparelhados.

# Premissa 2: Os dados de cada amostra/grupo possuem distribuicao
#             normal?
# Vamos usar o teste de normalidade com o seguinte
# teste de hipoteses:

# - H0: os dados sao normalmente distribuidos
# - Ha: os dados nao sao normalmente distribuidos

# Primeiro vamos fazer o teste de normalidade para a
# Idade das esposas
> ad.test(salarios$age)
```

Resultado

```
      Anderson-Darling normality test
data:  salarios$age
A = 31.828, p-value < 0.00000000000000022
```

P-value < 0.00000000000000022 (ou seja, <0.05), então rejeitamos H0 e identificamos que os dados não são normalmente distribuídos.

Código

```
# Agora vamos fazer o teste de normalidade para a
# Idade dos maridos
> ad.test(salarios$husage)
```

Resultado

```
      Anderson-Darling normality test

data:  salarios$husage
A = 28.176, p-value < 0.00000000000000022
```

P-value < 0.00000000000000022 (ou seja, <0.05), então rejeitamos H0 e identificamos que os dados não são normalmente distribuídos.

Conclusão: não é possível usar o unpaired test (paramétrico), pois não foi atendida a premissa de que os dados do grupo possuem distribuição normal.

Portanto, vamos utilizar o teste não paramétrico:

Código

```
# Vamos fazer o teste se a idade mediana dos maridos eh
# estatisticamente igual a idade mediana das mulheres

# Hipoteses do teste:

# H0: A idade mediana dos maridos eh igual estatisticamente a idade
mediana das mulheres;
# Ha: A idade mediana dos maridos nao eh igual estatisticamente a
idade mediana das mulheres

> res <- wilcox.test(salarios$age, salarios$husage,
+                    exact = FALSE, conf.int=TRUE)
> res
```

Resultado

```
      Wilcoxon rank sum test with continuity correction
```

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

Com base no resultado do teste de Wilcoxon, rejeitamos H0, pois p-value é menor que 0,05. Portanto, concluímos que a idade mediana dos maridos é estatisticamente diferente da idade mediana das esposas. O intervalo de confiança da diferença entre as medianas está entre -3 e -2, com uma diferença da mediana de -2,999966, ou seja, a mediana da idade das mulheres aproximadamente 3 anos inferior à dos maridos.

APÊNDICE E - ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

Código

```
> # Carregar os pacotes necessários
> library(caret)
> library(glmnet)
> library(knitr)
> library(rmarkdown)
> # Carregar a base de dados
> load("trabalhosalarios.RData")
> # Dividir os dados em conjunto de treinamento e teste
> set.seed(123) # para reprodutibilidade
> trainIndex =
sample(1:nrow(trabalhosalarios),0.8*nrow(trabalhosalarios))
```

```
> train <- trabalhosalarios[trainIndex, ]
> test <- trabalhosalarios[-trainIndex, ]
> # Verificando as dimensões dos conjuntos de treino e teste
> # Vamos padronizar as variaveis
> # Vamos criar um objeto com as variaveis para padronizar
> # As variaveis binarias nao sao padronizadas
> cols = c('husage', 'husearns', 'huseduc', 'hushrs', 'earns',
+         'age', 'educ', 'exper')
> # Padronizando a base de treinamento e teste
> pre_proc_val <- preProcess(train[,cols],
+                            method = c("center", "scale"))
> train[,cols] = predict(pre_proc_val, train[,cols])
> test[,cols] = predict(pre_proc_val, test[,cols])
```

Regressão Ridge

Código

```
> # Objeto com as variaveis que serão usadas
> cols_reg = names(trabalhosalarios)
> # Vamos gerar variaveis dummies para organizar os datasets
> # em objetos tipo matriz
> # Estamos interessados em estimar o salario-hora(log) (lwage)
> dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+
+                      husblck+hushisp+hushrs+kidge6+earns+age+
+                      black+educ+hispanic+union+exper+kidlt6,
+                      data = trabalhosalarios[,cols_reg])
> train_dummies = predict(dummies, newdata = train[,cols_reg])
> test_dummies = predict(dummies, newdata = test[,cols_reg])
> x = as.matrix(train_dummies)
> y_train = train$lwage
> x_test = as.matrix(test_dummies)
> y_test = test$lwage
> lambdas <- 10^seq(2, -3, by = -.1)
> # Calculando o lambda:
> ridge_lamb <- cv.glmnet(x, y_train, alpha = 0,
+                         lambda = lambdas)
> # Vamos ver qual o lambda otimo
> best_lambda_ridge <- ridge_lamb$lambda.min
> # Estimando o modelo Ridge
> ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
+                    family = 'gaussian',
+                    lambda = best_lambda_ridge)
> # Vamos ver o resultado (valores) da estimativa
> # (coeficientes)
> ridge_reg[["beta"]]
```

Resultado

```
16 x 1 sparse Matrix of class "dgCMatrix"
                    s0
husage    0.006397754
husunion  0.005280570
husearns  0.039217775
huseduc   0.013724222
husblck  -0.003691159
hushisp   0.013267092
hushrs   -0.020264678
kidge6    0.024676722
```

```
earns     0.369863911
age       0.016272185
black    -0.017860325
educ      0.058767989
hispanic -0.052478381
union     0.071476614
exper    -0.003305692
kidlt6    0.073278976
```

Código

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

Código

```
> # Predicao e avaliacao nos dados de treinamento:
> predictions_train <- predict(ridge_reg,
+                              s = best_lambda_ridge,
+                              newx = x)
> # As metricas da base de treinamento sao:
> eval_results(y_train, predictions_train, train)
```

Resultado

```
       RMSE   Rsquare
1 0.2930866 0.6867948
```

Código

```
> # Predicao e avaliacao nos dados de teste:
> predictions_test <- predict(ridge_reg,
+                             s = best_lambda_ridge,
+                             newx = x_test)
> # As metricas da base de teste sao:
> eval_results(y_test, predictions_test, test)
```

Resultado

```
      RMSE   Rsquare
1 0.281537 0.7009349
```

Código

```
> # Fazendo uma predição com o modelo
> our_pred.husage = (40-pre_proc_val[["mean"]][["husage"]])/
+  pre_proc_val[["std"]][["husage"]]
> our_pred.husunion = 0
> our_pred.husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
+  pre_proc_val[["std"]][["husearns"]]
```

```
> our_pred.huseduc = (13-pre_proc_val[["mean"]][["huseduc"]])/
+  pre_proc_val[["std"]][["huseduc"]]
> our_pred.husblck = 1
> our_pred.hushisp = 0
> our_pred.hushrs = (40-pre_proc_val[["mean"]][["hushrs"]])/
+  pre_proc_val[["std"]][["hushrs"]]
> our_pred.kidge6 = 1
> our_pred.age = (38-pre_proc_val[["mean"]][["age"]])/
+  pre_proc_val[["std"]][["age"]]
> our_pred.black = 0
> our_pred.educ = (13-pre_proc_val[["mean"]][["educ"]])/
+  pre_proc_val[["std"]][["educ"]]
> our_pred.hispanic = 1
> our_pred.union = 0
> our_pred.exper = (18-pre_proc_val[["mean"]][["exper"]])/
+  pre_proc_val[["std"]][["exper"]]
> our_pred.kidlt6 = 1
> # Definido este valor, pois sem a variável ocorre erro na predição
> our_pred.earns = (360-pre_proc_val[["mean"]][["earns"]])/
+  pre_proc_val[["std"]][["earns"]]

> our_pred = as.matrix(data.frame(husage=our_pred.husage,
+                                 husunion=our_pred.husunion,
+                                 husearns=our_pred.husearns,
+                                 huseduc=our_pred.huseduc,
+                                 husblck=our_pred.husblck,
+                                 hushisp=our_pred.hushisp,
+                                 hushrs=our_pred.hushrs,
+                                 kidge6=our_pred.kidge6,
+                                 earns=our_pred.earns,
+                                 age=our_pred.age,
+                                 black=our_pred.black,
+                                 educ=our_pred.educ,
+                                 hispanic=our_pred.hispanic,
+                                 union=our_pred.union,
+                                 exper=our_pred.exper,
+                                 kidlt6=our_pred.kidlt6))

> # Fazendo a predicao:
> predict_our_ridge <- predict(ridge_reg,
+                              s = best_lambda_ridge,
+                              newx = our_pred)
> # O resultado da predicao eh:
> predict_our_ridge
```

Resultado

```
          s1
[1,] 2.173516
```

Código

```
> # Encontrando o valor final (antilog)
> hrwage_pred_ridge = exp(predict_our_ridge)
> hrwage_pred_ridge
```

Resultado

```
         s1
[1,] 8.78913
```

Código

```
> # Intervalo de confianca:
> n <- nrow(train) # tamanho da amostra
> m <- predict_our_ridge # valor medio predito
> s <- sd(train$lwage) # desvio padrao
> dam <- s/sqrt(n) # distribuicao da amostragem da media
> CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
> CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior
> # Os valores sao:
> exp(CIlwr_ridge)
> exp(CIupr_ridge)
```

Resultado

```
         s1
[1,] 8.5925
          s1
[1,] 8.990259
```

Com os valores usados na predição, o salário-hora predito para a esposa foi de US$ 8.79, podendo variar de US$ 8.59 a US$ 8.99

Regressão Lasso

Código

```
> lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
+                         lambda = lambdas,
+                         standardize = TRUE, nfolds = 5)
> best_lambda_lasso <- lasso_lamb$lambda.min
> lasso_model <- glmnet(x, y_train, alpha = 1,
+                       lambda = best_lambda_lasso,
+                       standardize = TRUE)
```

Código

```
> predictions_train <- predict(lasso_model,
+                              s = best_lambda_lasso,
+                              newx = x)
> eval_results(y_train, predictions_train, train)
       RMSE  Rsquare
```

Resultado

```
1 0.2931455 0.686669
```

Código

```
> predictions_test <- predict(lasso_model,
+                             s = best_lambda_lasso,
+                             newx = x_test)
> eval_results(y_test, predictions_test, test)
```

Resultado

```
       RMSE   Rsquare
1 0.2797465 0.7047266
```

Código

```
> # Fazendo a predicao (dataframe com os valores ja foi definido
anteriormente)
> predict_our_lasso <- predict(lasso_model,
+                              s = best_lambda_lasso,
+                              newx = our_pred)
> # Encontrando o valor final (antilog)
> hrwage_pred_lasso = exp(predict_our_lasso)
> # Intervalo de confianca:
> n <- nrow(train) # tamanho da amostra
> m <- predict_our_lasso # valor medio predito
> s <- sd(train$lwage) # desvio padrao
> dam <- s/sqrt(n) # distribuicao da amostragem da media
> CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
> CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior
> # Os valores sao:
> hrwage_pred_lasso
> exp(CIlwr_ridge)
> exp(CIupr_ridge)
```

Resultado

```
           s1
[1,] 8.831578
           s1
[1,] 8.633999
           s1
[1,] 9.033679
```

Com os valores usados na predição, o salário-hora predito para a esposa foi de US$ 8.83, podendo variar de US$ 8.63 a US$ 9.03

Elasticnet

Código

```
> train_cont <- trainControl(method = "repeatedcv",
+                            number = 10,
+                            repeats = 5,
+                            search = "random",
+                            verboseIter = TRUE)
elastic_reg <- train(lwage ~ husage+husearns+huseduc+hushrs+
                       earns+age+educ+husblck+hushisp+
                       kidge6+black+hispanic+union+kidlt6,
                     data = train,
                     method = "glmnet",
                     tuneLength = 10,
                     trControl = train_cont)
```

Código

```
> predictions_train <- predict(elastic_reg, x)
> # As metricas na base de treino sao:
> eval_results(y_train, predictions_train, train)
```

Resultado

```
      RMSE   Rsquare
1 0.292994 0.6869926
```

Código

```
> # Vamos fazer as predicoes na base de teste
> predictions_test <- predict(elastic_reg, x_test)
> # As metricas de performance na base de teste sao:
> eval_results(y_test, predictions_test, test)
```

Resultado

```
       RMSE   Rsquare
1 0.2800212 0.7041465
```

Código

```
> # Fazendo a predicao neste modelo:
> predict_our_elastic <- predict(elastic_reg,our_pred)
> # Encontrando o valor final (antilog)
> hrwage_pred_elast = exp(predict_our_elastic)
> # Intervalo de confianca:
> n <- nrow(train) # tamanho da amostra
> m <- predict_our_elastic # valor medio predito
> s <- sd(train$lwage) # desvio padrao
> dam <- s/sqrt(n) # distribuicao da amostragem da media
> CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
> CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior
> # Os valores sao:
> hrwage_pred_elast
> exp(CIlwr_ridge)
> exp(CIupr_ridge)
```

Resultado

```
[1] 8.890571
[1] 8.691672
[1] 9.094022
```

Com os valores usados na predição, o salário-hora predito para a esposa foi de US$ 8.89, podendo variar de US$ 8.69 a US$ 9.09

Comparação entre os modelos

Podemos agora fazer a comparação entre os modelos para tirar algumas conclusões.

Regressão	RMSE (treino)	R^2 (treino)	RMSE (teste)	R^2 (teste)
Ridge	0.2930866	0.6867948	0.281537	0.7009349
Lasso	0.2931455	0.686669	0.2797465	0.7047266
ElasticNet	0.292994	0.6869926	0.2800212	0.7041465

Ao analisar as métricas dos diferentes modelos, percebemos que nos três casos os valores ficaram muito próximos, com diferenças mínimas entre eles. Sendo necessária a seleção de um deles, poderia ser selecionado o ElasticNet, que obteve os melhores números na base de treino e o segundo melhor na base de teste. Porém, como a diferença observada é muito pequena, qualquer um dos modelos, neste caso, deverá apresentar resultados semelhantemente corretos.

Predição e intervalos de confiança:

Na tabela a seguir podemos ver os valores preditos de salário-hora, bem como o intervalo de confiança, exibindo o valor mínimo e máximo:

Regressão	Valor predito	Intervalo de confiança
Ridge	US$ 8.79	US$ 8.59 a US$ 8.99
Lasso	US$ 8.83	US$ 8.63 a US$ 9.03
ElasticNet	US$ 8.89	US$ 8.69 a US$ 9.09

APÊNDICE F - ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

1 Construção de Características: Identificador automático de idioma

Identificador automático de idioma

Problema: Dados um texto de entrada, é possível identificar em qual língua o texto está escrito?

Entrada: "texto qualquer"

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

O processo de Reconhecimento de Padrões

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)

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.",
]
```

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

Código

```
import random
import pandas as pd

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)

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

Resultado

```
0          1
0     Estoy estudiando para mis exámenes.   espanhol
1                 Ella canta hermosamente.   espanhol
2                        Have a great day!     inglês
3           I'm learning to play the guitar.     inglês
4             Me gustaría ir de vacaciones.   espanhol
..                                      ...        ...
87    Could you pass me the salt, please?     inglês
88  O restaurante tem uma vista incrível.  português
89          O trânsito está terrível hoje.  português
90              I'm studying for my exams.     inglês
91          ¿Cómo estuvo tu fin de semana?   espanhol

[92 rows x 2 columns]
```

Construção dos atributos

Esse é o coração desse trabalho e que deverá ser desenvolvido por vocês. Pensem em como podemos "medir" cadas frase/sentença e extrair características que melhorem o resultado do processo de identificação. Após a criação de cada novo atributo, execute as etapas seguintes e registre as métricas da matriz de confusão. Principalmente acurácia e a precisão.

Código

```
# 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)


# conta a quantidade de w e y na frase
def funcao2(frase):
    w = frase.lower().count("w")
    y = frase.lower().count("y")
    return w + y


# conta a quantidade de acentos em vogais na frase
def funcao3(frase):
    acentos = (
        frase.lower().count("á")
        + frase.lower().count("é")
        + frase.lower().count("í")
        + frase.lower().count("ó")
        + frase.lower().count("ú")
    )
    acentos = (
        acentos
        + frase.lower().count("ã")
        + frase.lower().count("õ")
        + frase.lower().count("ê")
        + frase.lower().count("ô")
    )
    return acentos


def extraiCaracteristicas(frase):
```

```
    # frase é um vetor [ 'texto', 'lingua' ]
    texto = frase[0]
    pattern_regex = re.compile("[^\w+]", re.UNICODE)
    texto = re.sub(pattern_regex, " ", texto)
    # print(texto)
    caracteristica1 = tamanhoMedioFrases(texto)
    caracteristica2 = funcao2(texto)
    caracteristica3 = funcao3(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
```

Resultado

```
0  1  2          3
0   6.000000  1  1   espanhol
1   7.000000  0  0   espanhol
2   3.250000  1  0     inglês
3   3.571429  1  0     inglês
4   4.800000  0  1   espanhol
..       ... .. ..        ...
87  3.857143  1  0     inglês
88  5.166667  0  1  português
89  5.000000  0  2  português
90  3.333333  2  0     inglês
91  3.833333  0  1   espanhol

[92 rows x 4 columns]
```

Treinando o modelo com SVM

Separando o conjunto de treinamento do conjunto de testes

Código

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


# from sklearn.metrics import confusion_matrix
```

```
vet = np.array(padroes)
classes = vet[:, -1]  # classes = [p[-1] for p in padroes]
# print(classes)
padroes_sem_classe = vet[:, 0:-1]
# print(padroes_sem_classe)
X_train, X_test, y_train, y_test = train_test_split(
    padroes_sem_classe, classes, test_size=0.25, stratify=classes
)
```

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

Código

```
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))
```

Resultado

```
Acurácia nos dados de treinamento: 75.36%
[[ 9  5  8]
 [ 2 20  1]
 [ 1  0 23]]
              precision    recall  f1-score   support

    espanhol       0.75      0.41      0.53        22
      inglês       0.80      0.87      0.83        23
   português       0.72      0.96      0.82        24

    accuracy                           0.75        69
   macro avg       0.76      0.75      0.73        69
weighted avg       0.76      0.75      0.73        69

métricas mais confiáveis
```

```
[[1 3 4]
 [0 6 1]
 [0 1 7]]
              precision    recall  f1-score   support

    espanhol       1.00      0.12      0.22         8
      inglês       0.60      0.86      0.71         7
   português       0.58      0.88      0.70         8

    accuracy                           0.61        23
   macro avg       0.73      0.62      0.54        23
weighted avg       0.73      0.61      0.54        23
```

2 Melhore uma base de dados ruim

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

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

Base utilizada: Spambase - https://archive.ics.uci.edu/dataset/94/spambase

Definição das colunas existentes na base

Código

```
colunas = [
    "word_freq_make",
    "word_freq_address",
    "word_freq_all",
    "word_freq_3d",
    "word_freq_our",
    "word_freq_over",
    "word_freq_remove",
    "word_freq_internet",
    "word_freq_order",
    "word_freq_mail",
    "word_freq_receive",
    "word_freq_will",
    "word_freq_people",
    "word_freq_report",
```

```
    "word_freq_addresses",
    "word_freq_free",
    "word_freq_business",
    "word_freq_email",
    "word_freq_you",
    "word_freq_credit",
    "word_freq_your",
    "word_freq_font",
    "word_freq_000",
    "word_freq_money",
    "word_freq_hp",
    "word_freq_hpl",
    "word_freq_george",
    "word_freq_650",
    "word_freq_lab",
    "word_freq_labs",
    "word_freq_telnet",
    "word_freq_857",
    "word_freq_data",
    "word_freq_415",
    "word_freq_85",
    "word_freq_technology",
    "word_freq_1999",
    "word_freq_parts",
    "word_freq_pm",
    "word_freq_direct",
    "word_freq_cs",
    "word_freq_meeting",
    "word_freq_original",
    "word_freq_project",
    "word_freq_re",
    "word_freq_edu",
    "word_freq_table",
    "word_freq_conference",
    "char_freq_;",
    "char_freq_(",
    "char_freq_[",
    "char_freq_!",
    "char_freq_$",
    "char_freq_#",
    "capital_run_length_average",
    "capital_run_length_longest",
    "capital_run_length_total",
    "spam",
]
```

Obtém o arquivo diretamente da web

Código

```
import requests, zipfile, io
import pandas as pd

r =
requests.get("https://archive.ics.uci.edu/static/public/94/spambase.z
ip")
z = zipfile.ZipFile(io.BytesIO(r.content))
z.namelist()
dadosfp = z.open("spambase.data")
dados = dadosfp.read()
```

```
base_dados = pd.read_csv(io.BytesIO(dados), header=None,
names=colunas, lineterminator="\n", na_values="?")
```

Separar a classe dos atributos

Código

```
X = base_dados.iloc[:, :-1]
X_orig = X.copy()
Y = base_dados["spam"]
Y_orig = base_dados["spam"]
```

Verificar se existem colunas correlacionadas

Código

```
matriz_correlacao = X.corr()

colunas_alta_correlacao = []
for i in range(len(matriz_correlacao.columns)):
    for j in range(i + 1, len(matriz_correlacao.columns)):
        if abs(matriz_correlacao.iloc[i, j]) > 0.9:
            colunas_alta_correlacao.append(
                (matriz_correlacao.columns[i],
matriz_correlacao.columns[j]))

print("Colunas correlacionadas:")
for col1, col2 in colunas_alta_correlacao:
    print(f"{col1} and {col2}")
```

Resultado

```
Colunas correlacionadas:
word_freq_857 and word_freq_415
```

Removendo uma das colunas correlacionadas

Código

```
X.drop("word_freq_857", axis=1, inplace=True)
```

Normalização

Código

```
from sklearn.preprocessing import minmax_scale
X = pd.DataFrame(minmax_scale(X))
```

A próxima seção trata da construção do modelo, dos testes e das métricas da matriz de confusão

Código

```
from sklearn.model_selection import train_test_split

# com os dados originais
```

```
X_orig_train, X_orig_test, y_orig_train, y_orig_test =
train_test_split(
    X_orig, Y_orig, test_size=0.25, stratify=Y_orig, random_state=10
)

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

Treina o modelo com base nos dados originais (SVM)

Código

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

treinador = svm.SVC()  # algoritmo escolhido

modelo_orig = treinador.fit(X_orig_train, y_orig_train)

# predição com os mesmos dados usados para testar
print("Matriz de confusão - com os dados ORIGINAIS usados para
TESTES")
y2_orig_pred = modelo_orig.predict(X_orig_test)
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
print(cm_orig_test)
print(classification_report(y_orig_test, y2_orig_pred))
```

Resultado

```
Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[615  82]
 [268 186]]
              precision    recall  f1-score   support

           0       0.70      0.88      0.78       697
           1       0.69      0.41      0.52       454

    accuracy                           0.70      1151
   macro avg       0.70      0.65      0.65      1151
weighted avg       0.70      0.70      0.67      1151
```

Como os dados ficam após os processos de tratamento dos dados?

Código

```
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)

# predição com os mesmos dados usados para testar
print("Matriz de confusão - com os dados TRATADOS usados para
TESTES")
```

```
y2_pred = modelo.predict(X_test)
cm_test = confusion_matrix(y_test, y2_pred)
print(cm_test)
print(classification_report(y_test, y2_pred))
```

Resultado

```
Matriz de confusão - com os dados TRATADOS usados para TESTES
[[663  34]
 [ 54 400]]
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       697
           1       0.92      0.88      0.90       454

    accuracy                           0.92      1151
   macro avg       0.92      0.92      0.92      1151
weighted avg       0.92      0.92      0.92      1151
```

APÊNDICE G - APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

Seed utilizado: 202400

Classificação

Para o experimento de Classificação:

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

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	C=100 Sigma=0,015	0,8503	Reference Prediction bus opel saab van bus 42 0 0 0 opel 0 28 7 1 saab 0 14 35 1 van 1 0 1 37
RNA – CV	size=21 decay=0,7	0,8323	Reference Prediction bus opel saab van bus 40 2 0 0 opel 0 28 9 0 saab 2 11 33 1 van 1 1 1 38
SVM – Hold-out	C=1 Sigma=0,06667012	0,7844	Reference Prediction bus opel saab van bus 41 0 0 0 opel 0 24 13 0 saab 0 17 28 1 van 2 1 2 38

RF – CV	mtry=5	0,7665	Reference Prediction bus opel saab van bus 42 0 0 0 opel 0 24 15 0 saab 0 17 25 2 van 1 1 3 37
RF – Hold-out	mtry=10	0,7545	Reference Prediction bus opel saab van bus 42 0 0 0 opel 0 25 17 1 saab 0 16 23 2 van 1 1 3 36
KNN	k=1	0,6168	Reference Prediction bus opel saab van bus 36 2 3 1 opel 1 18 21 3 saab 4 20 17 3 van 2 2 2 32
RNA – Hold-out	size=5 decay=0,1	0,485	Reference Prediction bus opel saab van bus 42 40 42 1 opel 1 1 0 0 saab 0 0 0 0 van 0 1 1 38

Novos Casos

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
1	100	40	96	150	60	9	160	40	25	150	180	390	190	73	5	18	189	195	saab
2	93	52	81	145	65	10	153	41	16	140	177	335	160	75	8	13	188	200	van
3	90	43	99	190	70	11	199	36	20	160	210	655	205	78	12	11	187	192	bus

Código

```
library("caret")

set.seed(202400)

dados <- read.csv("./dados/06 - Veículos/6 - Veiculos - Dados.csv")
dados$a <- NULL

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

ctrl <- trainControl(method = "cv", number = 10)
gridSvmCV = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))

svmCV <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl,
tuneGrid=gridSvmCV)
svmCV

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

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – CV	size=11 decay=0,4	0,7908	Reference Prediction neg pos neg 87 19 pos 13 34
SVM – CV	C=2 Sigma=0,01	0,7843	Reference Prediction neg pos neg 87 20 pos 13 33
RNA – Hold-out	size=3 decay=0,1	0,7778	Reference Prediction neg pos neg 85 19 pos 15 34
SVM – Hold-out	C=0,5 Sigma=0,1449667	0,7778	Reference Prediction neg pos neg 86 20 pos 14 33
RF – Hold-out	mtry=2	0,7778	Reference Prediction neg pos neg 87 21 pos 13 32
RF – CV	mtry=2	0,7712	Reference Prediction neg pos neg 86 21 pos 14 32
KNN	k=9	0,7647	Reference Prediction neg pos neg 85 21 pos 15 32

Novos Casos

	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
1	5	120	70	12	0	34.6	0.527	45	pos
2	3	93	68	36	83	28.6	0.451	34	neg
3	6	150	65	4	0	25.3	0.522	36	neg

Código

```
library("caret")

set.seed(202400)

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

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

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

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

rnaCV <- train(diabetes~., data=treino, method="nnet", trace=FALSE, trControl=ctrl,
tuneGrid=gridRnaCV, maxit=2000)
rnaCV

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

Regressão

Para o experimento de Regressão:

Ordenar por R2 descendente, ou seja, a técnica de melhor R2 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 R2 (criar um arquivo com novos casos à sua escolha)
- O Gráfico de Resíduos para a técnica/parâmetro de maior R2
- A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

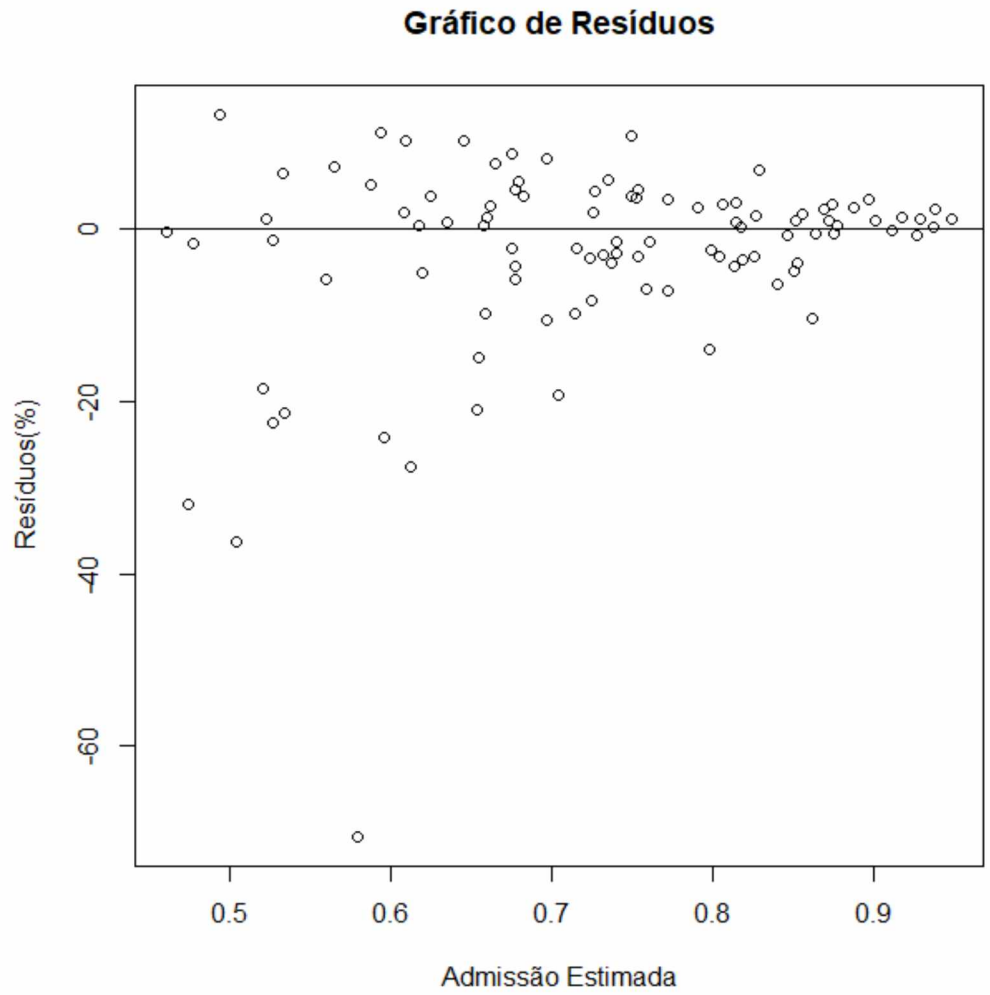
Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV	C=2 Sigma= 0,01	0,8576092	0,5451641	0,05673658	0,05506989	0,03948337
RF – Hold-out	mtry=2	0,8304361	0,5949125	0,05589008	0,06009524	0,04293873
RF – CV	mtry=2	0,8292982	0,5969053	0,05583056	0,06029654	0,04278709
RNA – CV	size=9 decay=0,1	0,822511	0,6086562	0,05306953	0,06148356	0,04558092
SVM – Hold-out	C=1 Sigma= 0,201492	0,8079223	0,6331766	0,05438461	0,06396049	0,04449268
RNA – Hold-out	size=5 decay=0,1	0,8032007	0,6409116	0,05182541	0,06474185	0,04748661
KNN	K=9	0,7832191	0,6726619	0,05741561	0,06794912	0,05113379

Novos Casos

	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	ChanceOfAdmit
1	340	120	5	4.0	4.3	9.5	0	0.9135143
2	331	110	3	4.5	4.6	8.3	1	0.7700531
3	323	108	4	3.5	3.8	8.7	1	0.7865742

Gráfico de Resíduos



Código

```
library("caret")
library("Metrics")

set.seed(202400)

dados <- read.csv("./dados/09 - Admissão/9 - Admissao - Dados.csv")
dados$num <- NULL

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

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

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

syx <- function(observado, predito) {
  return (sqrt((sum((predito-observado)^2) / (length(teste) -
(ncol(teste)-1)))))
}

pearson <- function(observado, predito) {
  mediaReais <- mean(teste$ChanceOfAdmit)
  mediaEstimados <- mean(predito)

  num <- sum((teste-mediaReais)*(predito-mediaEstimados))
  den <- sqrt(sum((teste-mediaReais)^2)) * sqrt(sum((predito-
mediaEstimados)^2))

  return (num / den)
}

metricas <- function(predito) {
  print("R2")
  print(r2(teste$ChanceOfAdmit, predito))

  print("SYX")
  print(syx(teste$ChanceOfAdmit, predito))

  print("PEARSON")
  print(pearson(teste$ChanceOfAdmit, predito))

  print("RMSE")
  print(rmse(teste$ChanceOfAdmit, predito))

  print("MAE")
  print(mae(teste$ChanceOfAdmit, predito))
}

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

svmCV <- train(ChanceOfAdmit~., data=treino, method="svmRadial",
trControl=ctrl, tuneGrid=gridSvmCV)
svmCV

predict.svmCV <- predict(svmCV, teste)
metricas(predict.svmCV)
```

Biomassa

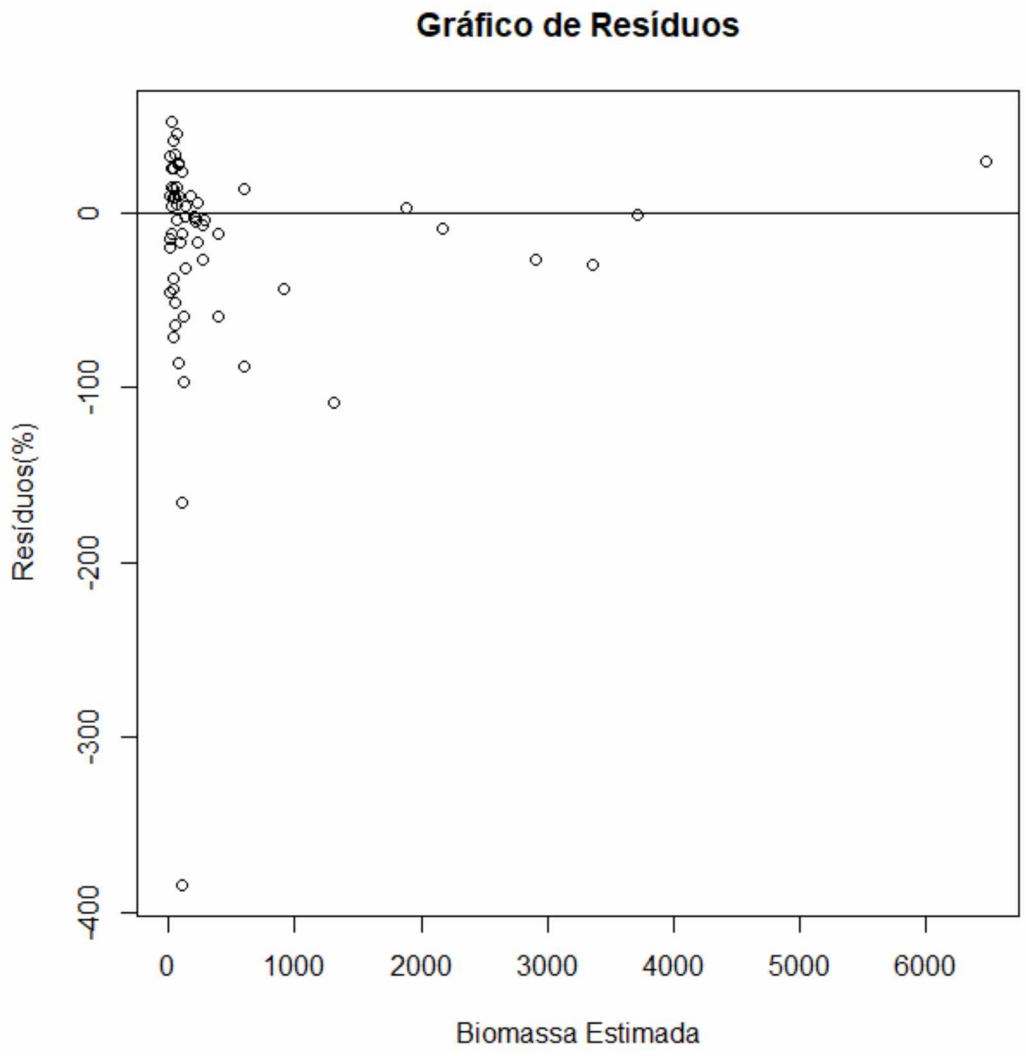
Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – CV	mtry=2	0,9155052	2986,813	0,837266	385,596	112,6905
RNA – CV	size=10 decay=0,1	0,908381	3110,182	0,8334981	401,5228	125,549
RF – Hold-out	mtry=2	0,8994709	3257,909	0,8316976	420,5943	117,8495
SVM – CV	C=100 Sigma= 0,01	0,8725528	3668,245	0,8137989	473,5684	159,3758

KNN	K=1	0,8113007	4463,528	0,8015866	576,239	152,3858
SVM – Hold-out	C=1 Sigma= 1,144223	0,4505385	7616,611	0,6136272	983,3002	236,1239
RNA – Hold-out	size=5 decay=0,1	-1,259574	15445,66	0,3510669	1994,026	518,2475

Novos Casos:

	dap	h	Me	biomassa
1	6.0	6.0	1.20	10.12790
2	7.0	4.8	1.06	11.45959
3	7.3	5.0	1.00	11.99973

Gráfico de Resíduos:



Código

```
library("caret")
library("Metrics")

set.seed(202400)

dados <- read.csv("./dados/05 - Biomassa/5 - Biomassa - Dados.csv")

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

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

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

syx <- function(observado, predito) {
  return (sqrt((sum((predito-observado)^2) / (length(teste) -
(ncol(teste)-1)))))
}

pearson <- function(observado, predito) {
  mediaReais <- mean(teste$biomassa)
  mediaEstimados <- mean(predito)

  num <- sum((teste-mediaReais)*(predito-mediaEstimados))
  den <- sqrt(sum((teste-mediaReais)^2)) * sqrt(sum((predito-
mediaEstimados)^2))

  return (num / den)
}

metricas <- function(predito) {
  print("R2")
  print(r2(teste$biomassa, predito))

  print("SYX")
  print(syx(teste$biomassa, predito))

  print("PEARSON")
  print(pearson(teste$biomassa, predito))

  print("RMSE")
  print(rmse(teste$biomassa, predito))

  print("MAE")
  print(mae(teste$biomassa, predito))
}

gridRfCV = expand.grid(mtry=c(2, 5, 7, 9))

rfCV <- train(biomassa~., data=treino, method="rf", trControl=ctrl,
tuneGrid=gridRfCV)
rfCV

predict.rfCV <- predict(rfCV, teste)
metricas(predict.rfCV)
```

Agrupamento

Veículo

Lista de Clusters gerados.
10 primeiras linhas do arquivo com o cluster correspondente:
Usa 10 clusters no experimento.
Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos

Lista de Clusters gerados:

```
Cluster means:
       Comp     Circ     DCirc    RadRa PrAxisRa    MaxLRa   ScatRa    Elong PrAxisRect MaxLRect ScVarMaxis
1  89.96000 39.17333  75.70667 168.4267 65.49333  9.400000 149.6933 44.17333   19.00000 135.4533   173.8800
2 107.25000 55.54167 103.41667 190.2500 55.79167  5.791667 248.7083 26.87500   27.08333 168.0833   271.2917
3  91.04950 45.54455  80.14851 157.9505 63.07921 10.019802 156.2574 43.01980   19.56436 152.1881   176.2475
4  93.47143 41.32857  80.14286 188.7857 66.95714  9.142857 165.7857 39.61429   20.14286 139.4286   192.6857
5 104.12409 53.65693 102.67153 201.8759 62.42336 10.474453 217.1825 30.70073   24.42336 169.2847   227.1022
6  85.66667 36.74074  57.31481 122.4444 56.18519  5.444444 120.8148 55.72222   17.22222 128.4074   140.3148
7  95.98507 45.95522  90.64179 196.2537 64.34328  8.313433 182.4179 35.89552   21.49254 148.6716   203.2687
8 102.14103 49.70513 100.51282 204.2051 63.66667  9.371795 200.6667 32.80769   22.94872 156.5897   218.5897
9  85.61069 43.19847  70.41221 138.4962 58.93130  7.198473 148.4885 45.29771   18.93893 143.6794   170.0305
10 88.88073 38.88991  66.82569 138.2477 57.80734  7.266055 134.4128 49.91743   18.02752 135.4128   156.9083
   ScVarmaxis    RaGyr SkewMaxis Skewmaxis Kurtmaxis KurtMaxis   HollRa
1    335.9200 142.2400  69.18667  4.720000 14.373333  193.4667 200.4133
2    910.5417 247.1667  83.62500  6.500000 14.208333  182.6667 183.6250
3    360.3069 178.3564  73.11881  7.108911  9.267327  187.3960 195.5248
4    414.8286 155.5714  69.82857  5.400000 15.000000  194.3143 200.6857
5    697.2482 212.8613  71.91971  7.583942 15.875912  187.9124 197.3942
6    215.5000 134.9444  74.96296  7.000000 12.185185  185.9444 190.8333
7    502.8060 179.2239  68.01493  6.238806 12.970149  193.7761 200.3433
8    603.9103 199.7308  69.38462  6.461538 14.756410  191.1538 199.0000
9    324.9847 173.2214  77.76336  6.000000  9.137405  183.6412 188.7863
10   267.5505 142.8165  71.34862  6.091743 11.045872  189.7156 194.9266
```

10 primeiras linhas do arquivo com o cluster correspondente:

1. 3
2. 9
3. 8
4. 1
5. 3
6. 2
7. 3
8. 10
9. 6
10. 7

Código

```
set.seed(202400)

dados <- read.csv("./dados/06 - Veículos/6 - Veiculos - Dados.csv")
dados$a <- NULL
dados$tipo <- NULL

km.res=kmeans(dados, 10)
print(km.res)

resultado <- cbind(dados, km.res$cluster)
resultado
```

Regras de associação

Musculação

Regras geradas com uma configuração de Suporte e Confiança.
Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos

Suporte: 0,33
Confiança: 0,9

```
    lhs                     rhs         support   confidence coverage  lift     count
[1] {Afundo}             => {Gemeos}    0.3461538 1.0000000  0.3461538 1.529412  9
[2] {AgachamentoSmith}   => {Extensor}  0.3461538 0.9000000  0.3846154 1.800000  9
[3] {Esteira}            => {Extensor}  0.4230769 0.9166667  0.4615385 1.833333 11
[4] {Extensor}           => {Bicicleta} 0.4615385 0.9230769  0.5000000 1.714286 12
[5] {Esteira, Extensor}  => {Bicicleta} 0.3846154 0.9090909  0.4230769 1.688312 10
[6] {Bicicleta, Esteira} => {Extensor}  0.3846154 1.0000000  0.3846154 2.000000 10
```

Código

```
library(arules)

set.seed(202400)

dados <- read.transactions("./dados/12 - Regras de Associacao -
Praticas/12 - Regras de Associacao - Praticas - 2 - Musculacao/2 -
Musculacao - Dados.csv", format="basket",sep=";")
inspect(dados)

rules <- apriori(dados, parameter = list(supp = 0.33, conf = 0.9,
target = "rules"))
inspect(rules)
```

APÊNDICE H - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

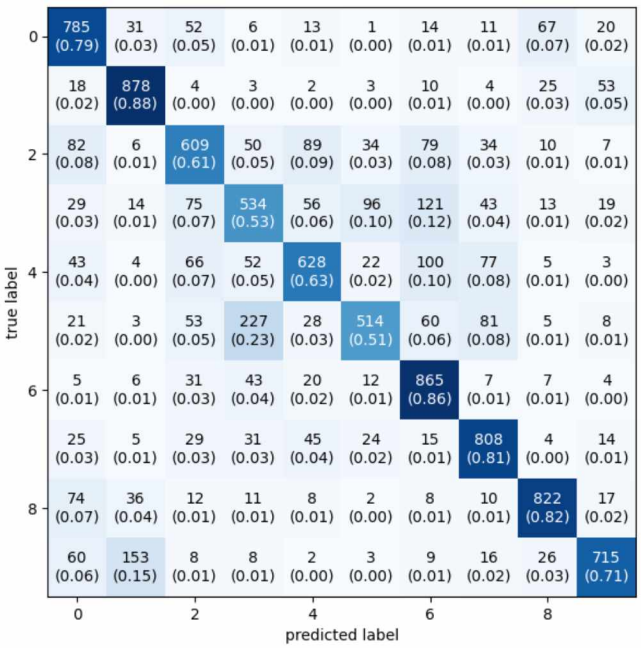
1 Classificação de Imagens (CNN)

Código

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

# Carga da base
cifar10 = tf.keras.datasets.cifar10

# Dados de treino e teste
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Normalização dos dados
# Imagens em pixels de 0 - 255 -> dividir por 255.0 transforma em 0 -
1
x_train, x_test = x_train / 255.0, x_test / 255.0

y_train, y_test = y_train.flatten(), y_test.flatten()

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

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

model = Model(i, x)

# 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)

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

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

Resultado



2 Detector de SPAM (RNN)

Código

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

# carrega e arruma a base
#!wget http://www.razer.net.br/datasets/spam.csv
!gdown --id 1dUTrZwPJiqu0TQPYsfm2fsF1wm1n1Rq_

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

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

# Acerta o tamanho das sequências (padding)
data_train = pad_sequences(sequences_train) # usa o tamanho da maior
seq.
T = data_train.shape[1]

data_test = pad_sequences(sequences_test, maxlen=T)

# 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,))
x = Embedding(V+1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation="sigmoid")(x)

model = Model(i, x)

# Compilar e treinar 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))

# 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")
```

Resultado

```
[[0.6188728]]
SPAM
```

3 Gerador de Dígitos Fake (GAN)

Código

```
# 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

# 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

# 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)

# Checkpoints
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint =
tf.train.Checkpoint(generator_optimizer=generator_optimizer,

discriminator_optimizer=discriminator_optimizer,
                                 generator=generator,
                                 discriminator=discriminator)

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

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

# Função que faz um passo de treinamento
@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_discriminato
r, 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)

    display.clear_output(wait=True)
    generate_and_save_images(generator, epoch + 1, seed)

    if (epoch + 1) % 15 == 0:
      checkpoint.save(file_prefix = checkpoint_prefix)
    print ('Time for epoch {} is {} sec'.format(epoch + 1,
time.time() - start))

  display.clear_output(wait=True)
```

```
  generate_and_save_images(generator, epochs, seed)

# Gerar e salvar imagens
def generate_and_save_images(model, epoch, test_input):
  predictions = model(test_input, training=False)
  fig = plt.figure(figsize=(4, 4))

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

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

# Treinar o modelo e restaurar o último ponto de verificação
train(train_dataset, EPOCHS)
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))
```

Resultado

4 Tradutor de Textos (Transformer)

Código

```
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)

# 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']

# 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/{mode
l_name}.zip", cache_dir='.', cache_subdir='', extract=True)

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

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

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

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

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

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

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

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

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

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

  # newaxis, aumenta a dimensão [] -> [ [] ]
```

```
  pos_encoding = angle_rads[np.newaxis, ...]
  return tf.cast(pos_encoding, dtype=tf.float32)

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

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

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

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

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

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

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

  # divide por sqrt(d_k)
  scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)

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

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

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

  return output, attention_weights

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

```
    assert d_model % self.num_heads == 0

    self.depth = d_model // self.num_heads

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

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

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

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

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

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

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

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

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

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

    return output, attention_weights

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

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

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

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

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

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

    ffn_output = self.ffn(out1) # (batch_size, input_seq_len,
d_model)
    ffn_output = self.dropout2(ffn_output, training=training)
    out2 = self.layernorm2(out1 + ffn_output) # (batch_size,
input_seq_len, d_model)

    return out2

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

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

    self.ffn = point_wise_feed_forward_network(d_model, dff)

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

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

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

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

    # (batch_size, target_seq_len, d_model)
```

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

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

    return out3, attn_weights_block1, attn_weights_block2

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

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

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

    for i in range(self.num_layers):
      x = self.enc_layers[i](x, training, mask)

    return x # (batch_size, input_seq_len, d_model)

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

    self.d_model = d_model
    self.num_layers = num_layers

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

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

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

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

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

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

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

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

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

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

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

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

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

    return final_output, attention_weights

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

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

    # Used in the 1st attention block in the decoder.
```

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

    return enc_padding_mask, look_ahead_mask, dec_padding_mask

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

# Otimizador

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

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

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

# Função de Perda e Métrica de Acurácia (mascarados)
loss_object =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)
  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask
  return tf.reduce_sum(loss_)/tf.reduce_sum(mask)

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

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

transformer = Transformer(
    num_layers=num_layers,
```

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

# Checkpoint
checkpoint_path = "./checkpoints/train"

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

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

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

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

for epoch in range(EPOCHS):
  start = time.time()
  train_loss.reset_state()
  train_accuracy.reset_state()
  # inp -> portuguese, tar -> english
  for (batch, (inp, tar)) in enumerate(train_batches):
    train_step(inp, tar)
    if batch % 50 == 0:
      print(f'Epoch {epoch + 1} Batch {batch} Loss
{train_loss.result():.4f} Accuracy {train_accuracy.result():.4f}')

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

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

```
class Translator(tf.Module):
  def __init__(self, tokenizers, transformer):
    self.tokenizers = tokenizers
    self.transformer = transformer
  def __call__(self, sentence, max_length=20):
    # input sentence is portuguese, hence adding the start and end
token
    assert isinstance(sentence, tf.Tensor)
    if len(sentence.shape) == 0:
      sentence = sentence[tf.newaxis]
    sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
    encoder_input = sentence
    # as the target is english, the first token to the transformer
should be the
    # english start token.
    start_end = self.tokenizers.en.tokenize([''])[0]
    start = start_end[0][tf.newaxis]
    end = start_end[1][tf.newaxis]
    output_array = tf.TensorArray(dtype=tf.int64, size=0,
dynamic_size=True)
    output_array = output_array.write(0, start)

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

# Efetuar uma tradução
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}')
```

Resultado

```
Prediction     : b'i read about trifuges in the encyclopedia .'
```

APÊNDICE I - BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

1. Estudo de Caso

Sabe-se que a enorme produção de dados, hoje conhecido popularmente como o novo petróleo, pode gerar inúmeros insights e informações valiosas que a mente humana não consegue processar e assim obter conhecimentos sobre os mais variados assuntos. Pensando desta forma, um hospital pode extrair, aprender e gerar conteúdo muito mais positivo do que aquele gerado em anos de pesquisas e convenções das quais os médicos são convidados a participar. Mas como o hospital pode conhecer melhor os tratamentos feitos aos pacientes? Como é mensurado? Existe um feedback do paciente dizendo esse remédio funcionou? Ou o seu 'não retorno' consta como resultado válido?

Essas respostas todas podem ser respondidas utilizando Big Data. A coleta de dados pelos médicos em grande quantidade, incluindo registros de sangue, qualidade de vida, exames, histórico de tratamentos, medicamentos receitados, acompanhamentos, pós-consulta, comentários em redes sociais... todas essas informações analisadas geram uma base de dados rica e sólida (5Vs), sendo um potencial ferramenta preditiva para auxiliar a melhora dos tratamentos convencionais.

A solução está em utilizar Machine Learning. É possível prever crises de saúde, ataques cardíacos ou infecções; com base em sinais vitais e histórico médico e familiar é possível prever, com mais eficiência, futuras doenças; e até personalizar os tratamentos.

Com isso, a precisão dos diagnósticos pode reduzir as readmissões (retornos) hospitalares e ser mais assertivo quanto as prescrições, exames e acompanhamentos. Aplicado a rede pública de saúde (SUS), essa aplicação pode trazer vários benefícios desde um melhor controle da saúde pública, até um menor custo e consumo de insumos hospitalares.

2. Dados Processados

Para implementar uma solução de Big Data em um hospital, especialmente utilizando Machine Learning para melhorar tratamentos e prever crises de saúde, são necessárias diversas ferramentas e tecnologias. Desta forma, os dados a serem processados segundo as fontes, caracterização e objetivos das ferramentas aplicadas são:

Nome, idade, sexo, peso

Tipo de Dados: Estruturados

Vs: Volume e Velocidade no acesso

Histórico de doenças e Tratamentos

Tipo de Dados: Estruturados, Temporais e Interação

Vs: Volume e Veracidade

Registros de exames de sangue

Tipo de Dados: Estruturados e Temporais

Vs: Volume e Velocidade no acesso

Exames de Ressonância e Raio-X

Tipo de Dados: Estruturados e Temporais

Vs: Volume e Velocidade no acesso

Relatório de Consultas

Tipo de Dados: Estruturados e Interação

Vs: Volume e Velocidade no acesso

Medicamentos Receitados

Tipo de Dados: Estruturados

Vs: Volume

Notas Médicas

Tipo de Dados: Estruturados e Semi-Estruturados

Vs: Volume e Veracidade

Prescrições (Receitas)

Tipo de Dados: Estruturados e Semi-Estruturados

Vs: Volume e Velocidade

Relatório de Consultas

Tipo de Dados: Semi-Estruturados

Vs: Volume, Velocidade e Veracidade

Pós-consulta - Comentários em Mídias Sociais

Tipo de Dados: Não-Estruturados e Interação

Vs: Volume, Velocidade, Variedade, Veracidade e Valor

Imagens de Ressonância e Raio-X

Tipo de Dados: Não-Estruturados

Vs: Volume, Variedade e Valor

Sensores diversos - Dados de equipamentos médicos (monitores, elétrons)

Tipo de Dados: Não-Estruturados e Temporais

Vs: Volume, Variedade e Valor

Transcrições de conversas – Médico → Paciente

Tipo de Dados: Estruturados, Semi-Estruturados, Temporais e Interação

Vs: Variedade e Valor

3. Ferramentas Objetivos para Alcance

Sistemas de Registro Eletrônico de Saúde (EHR): Ferramentas como Epic, Cerner, ou sistemas personalizados de EHR, que coletam e armazenam dados de pacientes de maneira estruturada.
Plataformas de Big Data: Soluções como Hadoop e Spark para processar e analisar grandes volumes de dados. O Hadoop é útil para armazenar grandes quantidades de dados distribuídos, enquanto o Spark oferece processamento em tempo real e análise de dados com suporte para Machine Learning.
Bancos de Dados NoSQL: Ferramentas como MongoDB ou Cassandra são essenciais para armazenar dados não estruturados e semiestruturados, como notas médicas, relatórios de exames e históricos de tratamentos.
Plataformas de Machine Learning: Bibliotecas e frameworks como TensorFlow, Scikit-Learn, ou PyTorch são fundamentais para desenvolver e treinar modelos preditivos que auxiliem na personalização de tratamentos e na previsão de crises de saúde.
Ferramentas de Visualização: Soluções como Tableau, Power BI ou mesmo dashboards customizados em Python ou R para apresentar os dados analisados de forma compreensível e acessível aos médicos e administradores.

Ferramentas de Integração e ETL: Apache Nifi ou Talend podem ser usadas para extrair, transformar e carregar dados de diversas fontes, integrando dados de EHRs, sistemas de laboratório, e outros repositórios.

4. Da Modelagem de Dados

A modelagem de dados em um hospital utilizando Big Data pode variar dependendo do tipo de dados e do objetivo da análise.

Modelo Relacional: Pode ser usado para armazenar informações estruturadas como registros de pacientes, tratamentos prescritos, e resultados de exames. Bancos de dados relacionais como MySQL ou PostgreSQL são úteis para consultas complexas e integrações com outros sistemas hospitalares
Modelo Chave-Valor: Adequado para armazenar informações de acesso rápido, como consultas recentes dos pacientes ou dados de sensores médicos em tempo real. Ferramentas como Redis são frequentemente utilizadas para esse tipo de modelagem.
Modelo Documento: Ideal para armazenar registros médicos eletrônicos, relatórios de consultas, e anotações clínicas, onde cada documento pode representar um paciente com todas as suas informações de saúde. MongoDB é uma escolha comum para esse tipo de modelagem.
Modelo Colunar: Utilizado para grandes volumes de dados analíticos, como dados históricos de pacientes, para identificar padrões e tendências. Apache Cassandra é uma opção popular para essa modelagem, especialmente em análises de larga escala.

Modelo de Grafo: Pode ser usado para mapear relações complexas entre sintomas, doenças, tratamentos e pacientes, facilitando a análise de interações e coocorrências. Neo4j é um exemplo de banco de dados de grafos que pode ser utilizado nesse contexto.

Essas ferramentas, caracterizações, Vs e modelagens formam a base para a implementação eficaz de Big Data em um ambiente hospitalar, com o objetivo de melhorar a qualidade dos cuidados de saúde e a eficiência operacional.

APÊNDICE J - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

1 Extração de Características

Código

```
# 1. Carregue a base de dados de Treino
!unzip Train_Warwick.zip

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

import os
import shutil
import random

def split_dataset(dataset_dir, train_dir, val_dir, split_ratio=0.8):
  """
  Divide o conjunto de dados em conjuntos de treino e validação,
garantindo que os pacientes
  não sejam divididos entre os dois conjuntos.

  Args:
    dataset_dir: Caminho do diretório do conjunto de dados.
    train_dir: Caminho do diretório para o conjunto de treino.
    val_dir: Caminho do diretório para o conjunto de validação.
    split_ratio: Razão da divisão entre treino e validação (padrão
0.8).
  """

  patients = [set(),set(),set(),set()]
  for root, _, files in os.walk(dataset_dir):
    for file in files:
      if file.endswith(".png"):
        patient_id = file.split("_")[0]
        patients[int(root.split("/")[1])].add(patient_id)

  for patient_set in patients:
    patients_list = list(patient_set)
    random.shuffle(patients_list)

    num_train_patients = int(len(patients_list) * split_ratio)
```

```
    train_patients = patients_list[:num_train_patients]
    val_patients = patients_list[num_train_patients:]

    for root, _, files in os.walk(dataset_dir):
      for file in files:
        if file.endswith(".png"):
          patient_id = file.split("_")[0]
          src_path = os.path.join(root, file)

          if patient_id in train_patients:
            dst_dir = os.path.join(train_dir, os.path.basename(root))
            if not os.path.exists(dst_dir):
              os.makedirs(dst_dir)
            dst_path = os.path.join(dst_dir, file)
            shutil.copy(src_path, dst_path)
          elif patient_id in val_patients:
            dst_dir = os.path.join(val_dir, os.path.basename(root))
            if not os.path.exists(dst_dir):
              os.makedirs(dst_dir)
            dst_path = os.path.join(dst_dir, file)
            shutil.copy(src_path, dst_path)


dataset_dir = "Train_4cls_amostra"
train_dir = "train"
val_dir = "val"

split_dataset(dataset_dir, train_dir, val_dir)

# 3.1 Extraia características utilizando LBP
import cv2
import numpy as np
import pandas as pd
import os
from skimage.feature import local_binary_pattern

def extract_lbp_features(image_path):
  """
  Extrai as características LBP de uma imagem.

  Args:
    image_path: Caminho para a imagem.

  Returns:
    Um vetor de características LBP.
  """
  image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
  radius = 3
  n_points = 8 * radius
  lbp = local_binary_pattern(image, n_points, radius,
method="uniform")
  hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, n_points +
3), range=(0, n_points + 2))
  return hist


def generate_lbp_csv(data_dir, csv_path):
  """
  Gera um arquivo CSV contendo as características LBP para as imagens
em um diretório.
```

```
  Args:
    data_dir: Caminho para o diretório contendo as imagens.
    csv_path: Caminho para o arquivo CSV de saída.
  """

  rows = []
  for root, _, files in os.walk(data_dir):
    for file in files:
      if file.endswith(".png"):
        image_path = os.path.join(root, file)
        lbp_features = extract_lbp_features(image_path)
        rows.append([image_path] + list(lbp_features))

  df = pd.DataFrame(rows, columns=["filename"] + [f"lbp_{i}" for i in
range(len(rows[0]) - 1)])
  df.to_csv(csv_path, index=False)


train_data_dir = "train"
train_csv_path = "train_lbp_features.csv"
generate_lbp_csv(train_data_dir, train_csv_path)

val_data_dir = "val"
val_csv_path = "val_lbp_features.csv"
generate_lbp_csv(val_data_dir, val_csv_path)

# 3.2 Extraia características utilizando CNN VGG16
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16,
preprocess_input
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
import pandas as pd
import os

def extract_vgg16_features(image_path, model):
  """
  Extrai as características VGG16 de uma imagem.

  Args:
    image_path: Caminho para a imagem.
    model: Modelo VGG16 pré-treinado.

  Returns:
    Um vetor de características VGG16.
  """
  img = load_img(image_path, target_size=(224, 224))
  img_array = img_to_array(img)
  img_array = np.expand_dims(img_array, axis=0)
  img_array = preprocess_input(img_array)
  features = model.predict(img_array)
  return features.flatten()


def generate_vgg16_csv(data_dir, csv_path):
  """
  Gera um arquivo CSV contendo as características VGG16 para as
imagens em um diretório.

  Args:
    data_dir: Caminho para o diretório contendo as imagens.
```

```
    csv_path: Caminho para o arquivo CSV de saída.
  """

  # Carregar o modelo VGG16 pré-treinado sem a última camada
  base_model = VGG16(weights='imagenet', include_top=False,
pooling='avg')

  rows = []
  for root, _, files in os.walk(data_dir):
    for file in files:
      if file.endswith(".png"):
        image_path = os.path.join(root, file)
        vgg16_features = extract_vgg16_features(image_path,
base_model)
        rows.append([image_path] + list(vgg16_features))

  df = pd.DataFrame(rows, columns=["filename"] + [f"vgg16_{i}" for i
in range(len(rows[0]) - 1)])
  df.to_csv(csv_path, index=False)


# Exemplo de uso:
train_data_dir = "train"
train_csv_path = "train_vgg16_features.csv"
generate_vgg16_csv(train_data_dir, train_csv_path)

val_data_dir = "val"
val_csv_path = "val_vgg16_features.csv"
generate_vgg16_csv(val_data_dir, val_csv_path)

# 4. Treine modelos Random Forest, SVM e RNA para predição dos dados
extraídos.
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Carregar os dados de treinamento e validação
train_lbp_df = pd.read_csv("train_lbp_features.csv")
val_lbp_df = pd.read_csv("val_lbp_features.csv")

train_vgg16_df = pd.read_csv("train_vgg16_features.csv")
val_vgg16_df = pd.read_csv("val_vgg16_features.csv")

# Função para extrair rótulos das imagens
def extract_labels(df):
  labels = []
  for filename in df["filename"]:
    label = filename.split("/")[1]  # Extraindo o rótulo do nome do
arquivo
    labels.append(label)
  return labels

train_labels = extract_labels(train_lbp_df)
val_labels = extract_labels(val_lbp_df)

# Separar as características das imagens dos rótulos
X_train_lbp = train_lbp_df.drop("filename", axis=1)
X_val_lbp = val_lbp_df.drop("filename", axis=1)
```

```
X_train_vgg16 = train_vgg16_df.drop("filename", axis=1)
X_val_vgg16 = val_vgg16_df.drop("filename", axis=1)


# Treinamento dos modelos

# Random Forest
rf_model = RandomForestClassifier()
rf_model.fit(X_train_lbp, train_labels)
rf_predictions = rf_model.predict(X_val_lbp)
rf_accuracy = accuracy_score(val_labels, rf_predictions)
print(f"Acurácia do Random Forest com LBP: {rf_accuracy}")


# SVM
svm_model = SVC()
svm_model.fit(X_train_lbp, train_labels)
svm_predictions = svm_model.predict(X_val_lbp)
svm_accuracy = accuracy_score(val_labels, svm_predictions)
print(f"Acurácia do SVM com LBP: {svm_accuracy}")

# RNA (MLP)
mlp_model = MLPClassifier()
mlp_model.fit(X_train_lbp, train_labels)
mlp_predictions = mlp_model.predict(X_val_lbp)
mlp_accuracy = accuracy_score(val_labels, mlp_predictions)
print(f"Acurácia da RNA com LBP: {mlp_accuracy}")


# Random Forest VGG16
rf_model_vgg = RandomForestClassifier()
rf_model_vgg.fit(X_train_vgg16, train_labels)
rf_predictions_vgg = rf_model_vgg.predict(X_val_vgg16)
rf_accuracy_vgg = accuracy_score(val_labels, rf_predictions_vgg)
print(f"Acurácia do Random Forest com VGG16: {rf_accuracy_vgg}")


# SVM VGG16
svm_model_vgg = SVC()
svm_model_vgg.fit(X_train_vgg16, train_labels)
svm_predictions_vgg = svm_model_vgg.predict(X_val_vgg16)
svm_accuracy_vgg = accuracy_score(val_labels, svm_predictions_vgg)
print(f"Acurácia do SVM com VGG16: {svm_accuracy_vgg}")

# RNA (MLP) VGG16
mlp_model_vgg = MLPClassifier()
mlp_model_vgg.fit(X_train_vgg16, train_labels)
mlp_predictions_vgg = mlp_model_vgg.predict(X_val_vgg16)
mlp_accuracy_vgg = accuracy_score(val_labels, mlp_predictions_vgg)
print(f"Acurácia da RNA com VGG16: {mlp_accuracy_vgg}")
```

Resultado

```
Acurácia do Random Forest com LBP: 0.7833333333333333
Acurácia do SVM com LBP: 0.6666666666666666
Acurácia da RNA com LBP: 0.7666666666666667
Acurácia do Random Forest com VGG16: 0.8083333333333333
Acurácia do SVM com VGG16: 0.8083333333333333
Acurácia da RNA com VGG16: 0.7416666666666667
```

Código

```
# 5. Carregue a base de Teste e execute a tarefa 3 nesta base.
!unzip Test_Warwick.zip

test_data_dir = "Test_4cl_amostra"
test_csv_path = "test_lbp_features.csv"
generate_lbp_csv(test_data_dir, test_csv_path)

test_data_dir = "Test_4cl_amostra"
test_csv_path = "test_vgg16_features.csv"
generate_vgg16_csv(test_data_dir, test_csv_path)

# 6. Aplique os modelos treinados nos dados de teste.
test_lbp_df = pd.read_csv("test_lbp_features.csv")
test_vgg16_df = pd.read_csv("test_vgg16_features.csv")

X_test_lbp = test_lbp_df.drop("filename", axis=1)
X_test_vgg16 = test_vgg16_df.drop("filename", axis=1)


# Previsões com os modelos treinados

# Random Forest com LBP
rf_test_predictions = rf_model.predict(X_test_lbp)

# SVM com LBP
svm_test_predictions = svm_model.predict(X_test_lbp)

# RNA com LBP
mlp_test_predictions = mlp_model.predict(X_test_lbp)

# Random Forest com VGG16
rf_test_predictions_vgg = rf_model_vgg.predict(X_test_vgg16)

# SVM com VGG16
svm_test_predictions_vgg = svm_model_vgg.predict(X_test_vgg16)

# RNA com VGG16
mlp_test_predictions_vgg = mlp_model_vgg.predict(X_test_vgg16)


# 7. Calcule as métricas de Sensibilidade, Especificidade e F1-Score
com base em suas matrizes de confusão.
from sklearn.metrics import confusion_matrix, classification_report

# Função para calcular as métricas
def calculate_metrics2(y_true, y_pred):
  cm = confusion_matrix(y_true, y_pred)
  report = classification_report(y_true, y_pred, digits=4)
  print("Matriz de Confusão:")
  print(cm)
  print("\nRelatório de Classificação:")
  print(report)

def calculate_metrics(y_true, y_pred):
  """
  Calcula a sensibilidade, especificidade e F1-score com base na
matriz de confusão.
```

```
  Args:
    y_true: Rótulos verdadeiros.
    y_pred: Rótulos previstos.

  Returns:
    Sensibilidade, especificidade e F1-score.
  """

  cm = confusion_matrix(y_true, y_pred)
  report = classification_report(y_true, y_pred, output_dict=True)

  sensitivity = report['weighted avg']['recall']
  f1_score = report['weighted avg']['f1-score']

  especificidades = []

  # Calcula a especificidade para cada classe
  for i in range(cm.shape[0]):
    VN = np.sum(cm) - np.sum(cm[i, :]) - np.sum(cm[:, i]) + cm[i, i]
# Verdadeiros Negativos
    FP = np.sum(cm[:, i]) - cm[i, i]  # Falsos Positivos
    especificidade = VN / (VN + FP) if (VN + FP) > 0 else 0
    especificidades.append(especificidade)

  print("Matriz de Confusão:")
  print(cm)
  print(f"Sensibilidade={sensitivity}, F1-Score={f1_score}")
  print(f"Especificidade classes 0 a 4={especificidades}, Média das
especificidades={np.mean(especificidades)}")


test_labels = [filename.split('/')[1] for filename in
test_lbp_df['filename']]

# Calcular métricas para cada modelo
print("Métricas para Random Forest com LBP:")
calculate_metrics(test_labels, rf_test_predictions)

print("\nMétricas para SVM com LBP:")
calculate_metrics(test_labels, svm_test_predictions)

print("\nMétricas para RNA com LBP:")
calculate_metrics(test_labels, mlp_test_predictions)

print("\nMétricas para Random Forest com VGG16:")
calculate_metrics(test_labels, rf_test_predictions_vgg)

print("\nMétricas para SVM com VGG16:")
calculate_metrics(test_labels, svm_test_predictions_vgg)

print("\nMétricas para RNA com VGG16:")
calculate_metrics(test_labels, mlp_test_predictions_vgg)
```

Resultado

```
Métricas para Random Forest com LBP:
Matriz de Confusão:
[[74 16 11  0]
 [ 5 60 24  1]
 [ 7 40 43  0]
 [ 0  0  0 90]]
```

```
Sensibilidade=0.7196765498652291, F1-Score=0.7222027071304382
Especificidade classes 0 a 4=[0.9555555555555556, 0.800711743772242,
0.8754448398576512, 0.99644128113879], Média das
especificidades=0.9070383550810597

Métricas para SVM com LBP:
Matriz de Confusão:
[[36 44 21  0]
 [ 0 59 29  2]
 [ 1 40 44  5]
 [ 0  0  1 89]]
Sensibilidade=0.6145552560646901, F1-Score=0.6124390684216153
Especificidade classes 0 a 4=[0.9962962962962963, 0.701067615658363,
0.8185053380782918, 0.9750889679715302], Média das
especificidades=0.8727395545011203

Métricas para RNA com LBP:
Matriz de Confusão:
[[45 50  6  0]
 [ 1 72 17  0]
 [ 1 51 35  3]
 [ 0  5  4 81]]
Sensibilidade=0.628032345013477, F1-Score=0.6334706192546397
Especificidade classes 0 a 4=[0.9925925925925926, 0.6227758007117438,
0.9039145907473309, 0.9893238434163701], Média das
especificidades=0.8771517068670094

Métricas para Random Forest com VGG16:
Matriz de Confusão:
[[101   0   0   0]
 [ 18  41  31   0]
 [  0  15  75   0]
 [  0   0   3  87]]
Sensibilidade=0.8194070080862533, F1-Score=0.8075419975550647
Especificidade classes 0 a 4=[0.9333333333333333, 0.9466192170818505,
0.8790035587188612, 1.0], Média das
especificidades=0.9397390272835113

Métricas para SVM com VGG16:
Matriz de Confusão:
[[100   1   0   0]
 [ 13  41  36   0]
 [  0   6  83   1]
 [  0   0   0  90]]
Sensibilidade=0.8463611859838275, F1-Score=0.8324981138908549
Especificidade classes 0 a 4=[0.9518518518518518, 0.9750889679715302,
0.8718861209964412, 0.99644128113879], Média das
especificidades=0.9488170554896533

Métricas para RNA com VGG16:
Matriz de Confusão:
[[96  5  0  0]
 [10 38 42  0]
 [ 0  3 87  0]
 [ 0  0  0 90]]
Sensibilidade=0.8382749326145552, F1-Score=0.8234019579040894
Especificidade classes 0 a 4=[0.9629629629629629, 0.9715302491103203,
0.8505338078291815, 1.0], Média das
especificidades=0.9462567549756162
```

O melhor modelo foi o SVM utilizando as características extraídas com VGG16. A métrica usada para essa conclusão foi a Sensibilidade, que é importante em contextos onde não identificar um positivo pode ter consequências graves (como em diagnósticos médicos, por exemplo).

2 Redes Neurais

Código

```
import os
import shutil
import random
import cv2
import numpy as np
import pandas as pd
from skimage.feature import local_binary_pattern
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16,
preprocess_input
from tensorflow.keras.applications.resnet50 import ResNet50,
preprocess_input as resnet_preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix


# Diretórios dos dados
train_dir = "train"
val_dir = "val"
test_dir = "Test_4cl_amostra"

# Parâmetros
img_width, img_height = 224, 224
batch_size = 32
epochs = 10
num_classes = 4


# Função para criar o modelo VGG16
def create_vgg16_model():
    base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(img_width, img_height, 3))
    for layer in base_model.layers:
        layer.trainable = False

    x = Flatten()(base_model.output)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)
    predictions = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs=base_model.input, outputs=predictions)
    return model
```

```
# Função para criar o modelo ResNet50
def create_resnet50_model():
    base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(img_width, img_height, 3))
    for layer in base_model.layers:
        layer.trainable = False

    x = Flatten()(base_model.output)
    x = Dense(256, activation='relu')(x)
    x = Dropout(0.5)(x)
    predictions = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs=base_model.input, outputs=predictions)
    return model


# Função para treinar o modelo
def train_model(model, train_data, val_data, epochs,
augmentation=False):
    model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])

    if augmentation:
        train_datagen = ImageDataGenerator(
            rescale=1. / 255,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True
        )
    else:
        train_datagen = ImageDataGenerator(rescale=1. / 255)

    train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical'
    )

    val_datagen = ImageDataGenerator(rescale=1. / 255)
    val_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical'
    )

    history = model.fit(
        train_generator,
        steps_per_epoch=train_generator.samples // batch_size,
        epochs=epochs,
        validation_data=val_generator,
        validation_steps=val_generator.samples // batch_size
    )

    return model, history


# Função para avaliar o modelo
def evaluate_model(model, test_dir):
```

```
    test_datagen = ImageDataGenerator(rescale=1. / 255)
    test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False
    )

    predictions = model.predict(test_generator)
    y_pred = np.argmax(predictions, axis=1)
    y_true = test_generator.classes

    accuracy = accuracy_score(y_true, y_pred)
    report = classification_report(y_true, y_pred, digits=4)
    cm = confusion_matrix(y_true, y_pred)

    print(f"Acurácia: {accuracy}")
    print("\nRelatório de Classificação:")
    print(report)
    print("\nMatriz de Confusão:")
    print(cm)


# Treinar VGG16 sem Data Augmentation
vgg16_model = create_vgg16_model()
vgg16_model_no_aug, vgg16_history_no_aug = train_model(vgg16_model,
train_dir, val_dir, epochs, augmentation=False)
print("Avaliação do VGG16 sem Data Augmentation:")
evaluate_model(vgg16_model_no_aug, test_dir)


# Treinar VGG16 com Data Augmentation
vgg16_model = create_vgg16_model()
vgg16_model_aug, vgg16_history_aug = train_model(vgg16_model,
train_dir, val_dir, epochs, augmentation=True)
print("\nAvaliação do VGG16 com Data Augmentation:")
evaluate_model(vgg16_model_aug, test_dir)


# Treinar ResNet50 sem Data Augmentation
resnet50_model = create_resnet50_model()
resnet50_model_no_aug, resnet50_history_no_aug =
train_model(resnet50_model, train_dir, val_dir, epochs,
augmentation=False)
print("\nAvaliação do ResNet50 sem Data Augmentation:")
evaluate_model(resnet50_model_no_aug, test_dir)


# Treinar ResNet50 com Data Augmentation
resnet50_model = create_resnet50_model()
resnet50_model_aug, resnet50_history_aug =
train_model(resnet50_model, train_dir, val_dir, epochs,
augmentation=True)
print("\nAvaliação do ResNet50 com Data Augmentation:")
evaluate_model(resnet50_model_aug, test_dir)
```

Resultado

```
Avaliação do VGG16 sem Data Augmentation:
Found 371 images belonging to 4 classes.
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15s 1s/step
Acurácia: 0.9056603773584906

Relatório de Classificação:
              precision    recall  f1-score   support

           0     0.9792    0.9307    0.9543       101
           1     0.8247    0.8889    0.8556        90
           2     0.8750    0.8556    0.8652        90
           3     0.9444    0.9444    0.9444        90

    accuracy                         0.9057       371
   macro avg     0.9058    0.9049    0.9049       371
weighted avg     0.9080    0.9057    0.9064       371


Matriz de Confusão:
[[94  7  0  0]
 [ 2 80  8  0]
 [ 0  8 77  5]
 [ 0  2  3 85]]

Avaliação do VGG16 com Data Augmentation:
Found 371 images belonging to 4 classes.
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 201ms/step
Acurácia: 0.9083557951482479

Relatório de Classificação:
              precision    recall  f1-score   support

           0     0.9886    0.8614    0.9206       101
           1     0.8302    0.9778    0.8980        90
           2     0.9277    0.8556    0.8902        90
           3     0.9043    0.9444    0.9239        90

    accuracy                         0.9084       371
   macro avg     0.9127    0.9098    0.9082       371
weighted avg     0.9149    0.9084    0.9085       371


Matriz de Confusão:
[[87 14  0  0]
 [ 1 88  1  0]
 [ 0  4 77  9]
 [ 0  0  5 85]]

Avaliação do ResNet50 sem Data Augmentation:
Found 371 images belonging to 4 classes.
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 691ms/step
Acurácia: 0.40431266846361186

Relatório de Classificação:
              precision    recall  f1-score   support

           0     0.0000    0.0000    0.0000       101
           1     0.3136    1.0000    0.4775        90
           2     0.5600    0.1556    0.2435        90
           3     0.7797    0.5111    0.6174        90

    accuracy                         0.4043       371
   macro avg     0.4133    0.4167    0.3346       371
```

```
weighted avg     0.4011    0.4043    0.3247       371


Matriz de Confusão:
[[  0 101   0   0]
 [  0  90   0   0]
 [  0  63  14  13]
 [  0  33  11  46]]

Avaliação do ResNet50 com Data Augmentation:
Found 371 images belonging to 4 classes.
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8s 426ms/step
Acurácia: 0.40161725067385445

Relatório de Classificação:
              precision    recall  f1-score   support

           0     0.0000    0.0000    0.0000       101
           1     0.3136    1.0000    0.4775        90
           2     0.5455    0.1333    0.2143        90
           3     0.7581    0.5222    0.6184        90

    accuracy                         0.4016       371
   macro avg     0.4043    0.4139    0.3275       371
weighted avg     0.3923    0.4016    0.3178       371


Matriz de Confusão:
[[  0 101   0   0]
 [  0  90   0   0]
 [  0  63  12  15]
 [  0  33  10  47]]
```

Código

```
# Redefinindo o método de avaliação para não precisar executar todo o
bloco anterior novamente.
def evaluate_model2(model, test_dir):
    test_datagen = ImageDataGenerator(rescale=1. / 255)
    test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False
    )

    predictions = model.predict(test_generator)
    y_pred = np.argmax(predictions, axis=1)
    y_true = test_generator.classes

    calculate_metrics(y_true, y_pred)


# Executando os modelos na base de teste e gerando as métricas
print("Avaliação do VGG16 sem Data Augmentation:")
evaluate_model2(vgg16_model_no_aug, test_dir)

print("\nAvaliação do VGG16 com Data Augmentation:")
```

```
evaluate_model2(vgg16_model_aug, test_dir)

print("\nAvaliação do ResNet50 sem Data Augmentation:")
evaluate_model2(resnet50_model_no_aug, test_dir)

print("\nAvaliação do ResNet50 com Data Augmentation:")
evaluate_model2(resnet50_model_aug, test_dir)
```

Resultado

```
Avaliação do VGG16 sem Data Augmentation:
Found 371 images belonging to 4 classes.
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 153ms/step
Matriz de Confusão:
[[94  7  0  0]
 [ 2 80  8  0]
 [ 0  8 77  5]
 [ 0  2  3 85]]
Sensibilidade=0.9056603773584906, F1-Score=0.9063512208488926
Especificidade classes 0 a 4=[0.9925925925925926, 0.9395017793594306,
0.9608540925266904, 0.9822064056939501], Média das
especificidades=0.968788717543166

Avaliação do VGG16 com Data Augmentation:
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 156ms/step
Matriz de Confusão:
[[87 14  0  0]
 [ 1 88  1  0]
 [ 0  4 77  9]
 [ 0  0  5 85]]
Sensibilidade=0.9083557951482479, F1-Score=0.9085397152673824
Especificidade classes 0 a 4=[0.9962962962962963, 0.9359430604982206,
0.9786476868327402, 0.9679715302491103], Média das
especificidades=0.9697146434690919

Avaliação do ResNet50 sem Data Augmentation:
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 184ms/step
Matriz de Confusão:
[[  0 101   0   0]
 [  0  90   0   0]
 [  0  63  14  13]
 [  0  33  11  46]]
Sensibilidade=0.40431266846361186, F1-Score=0.324674758916673
Especificidade classes 0 a 4=[1.0, 0.298932384341637,
0.9608540925266904, 0.9537366548042705], Média das
especificidades=0.8033807829181494

Avaliação do ResNet50 com Data Augmentation:
12/12 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 143ms/step
Matriz de Confusão:
[[  0 101   0   0]
 [  0  90   0   0]
 [  0  63  12  15]
 [  0  33  10  47]]
Sensibilidade=0.40161725067385445, F1-Score=0.31782865580518466
Especificidade classes 0 a 4=[1.0, 0.298932384341637,
0.9644128113879004, 0.9466192170818505], Média das
especificidades=0.8024911032028469
```

O melhor resultado encontrado foi VGG16 com Data Augmentation. Para essa conclusão foi considerada a métrica de Sensibilidade, que é importante em contextos onde não identificar um positivo pode ter consequências graves. Além disso, nas outras duas métricas verificadas, este mesmo modelo também apresentou o melhor resultado.

APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

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

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

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

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

B – RESOLUÇÃO

Casos de uso de ChatGPT e suas dimensões éticas: uma análise crítica

Nome do(a) autor(a): Alexandre Baptista Neiva de Lima, Bruno Pieper Bunhak, Marcos Antenor de Souza Morais, Renan de Paula Rosa, Silvio Zimmermann Junior

RESUMO

Inteligência Artificial (IA) é definida como o campo que emprega recursos científicos e de engenharia para atribuir inteligência a máquinas e programas de computador, capacitando-os a realizar tarefas anteriormente exclusivas aos humanos (BOLANDER, 2019). Uma aplicação recente da IA é o Processamento de Linguagem Natural (NLP), exemplificado pelos Large Language Models (LLMs), que são treinados com extensos volumes de texto para prever palavras em sentenças parciais (DEVLIN et al., 2018). LLMs, como ChatGPT da OpenAI e Gemini do Google, são usados em tradução, classificação, escrita criativa e criação de código (ELOUNDOU et al., 2023). Desde seu lançamento, o ChatGPT tem sido utilizado por milhões para tarefas como agendamento, lembretes e gerenciamento de listas, além de melhorar campanhas de marketing, reduzir custos e gerar insights em empresas e organizações (AYINDE et al., 2023). Há usos possíveis também na educação, como na escrita de artigos e ensaios acadêmicos, por exemplo (MOLLICK; MOLLICK, 2022). No entanto, o crescimento do uso do ChatGPT destaca seus impactos sociais, econômicos, éticos e culturais, especialmente em relação aos vieses incorporados nos modelos (VIDHYA; DEVI; A.; MANJU, 2023). Este trabalho apresenta estudos de caso sobre o uso do ChatGPT, analisando criticamente seus dilemas éticos e propondo soluções com base na literatura especializada.

Palavras-chave: Estudo de Caso. Ética. Inteligência Artificial. ChatGPT. Interação Humano-Máquina.

ABSTRACT

Artificial Intelligence (AI) is defined as the field that employs scientific and engineering resources to attribute intelligence to machines and computer programs, enabling them to perform tasks previously exclusive to humans (BOLANDER, 2019). A recent application of AI is Natural Language Processing (NLP), exemplified by Large Language Models (LLMs), which are trained with extensive volumes of text to predict words in partial sentences (DEVLIN et al., 2018). LLMs, such as OpenAI's ChatGPT and Google's Gemini, are used in translation, classification, creative writing, and code generation (ELOUNDOU et al., 2023). Since its launch, ChatGPT has been utilized by millions for tasks like scheduling, reminders, and list management, as well as improving marketing campaigns, reducing costs, and generating insights in businesses and organizations (AYINDE et al., 2023). There are also possible uses in education, such as writing articles and academic essays, for example (MOLLICK; MOLLICK, 2022). However, the growing use of ChatGPT highlights its social, economic, ethical, and cultural impacts, especially concerning the biases incorporated into the models (VIDHYA; DEVI; A.; MANJU, 2023). This work presents case studies on the use of ChatGPT, critically analyzing its ethical dilemmas and proposing solutions based on specialized literature.

Keywords: Case Study. Ethics. Artificial Intelligence. ChatGPT. Human-Machine Interaction.

1. INTRODUÇÃO

Há mais de 60 anos, John McCarthy definiu Inteligência Artificial (AI) como sendo o campo capaz de empregar recursos científicos e de engenharia para atribuir inteligência (ou comportamentos inteligentes) a máquinas e programas de computador. Embora essa definição tenha problemas crônicos e inerentes, o objetivo deste campo quase sempre se traduz no esforço de construir computadores ou robôs capazes de realizar tarefas que antes apenas humanos teriam capacidade de executar (BOLANDER, 2019). Preenchendo algumas das lacunas nessa definição, Nilsson (2009) afirma que AI é "[...] é a atividade dedicada a tornar as máquinas inteligentes, e inteligência é a qualidade que permite a uma entidade funcionar apropriadamente e com perspicácia em seu ambiente".

Uma das aplicações recentes derivadas dos estudos de Inteligência Artificial e de Aprendizado de Máquina, pertencente a subcategoria de aplicações de Processamento de Linguagem Natural (NLP) são os chamados Large Language Models (LLM), ou no português Grandes Modelos de Linguagem. Esses modelos são treinados através de extensos volumes de dados de texto e são capazes de prever, de forma auto supervisionada, uma palavra em uma sentença parcial (DEVLIN et al., 2018). Implementados em diferentes soluções (como o ChatGPT, da OpenAI, ou o Gemini, do Google), estes modelos passaram a ser utilizados para diferentes tarefas, como tradução, classificação, escrita criativa e criação de código. LLMs podem

processar e produzir várias outras formas de dados sequenciais, não se limitando ao processamento de linguagem natural (ELOUNDOU et al., 2023).

Lançado ao público há pouco tempo, o ChatGPT já é utilizado por milhões de pessoas em todo o mundo nas mais diversas tarefas como agendamento de compromissos, definindo lembretes e gerenciando listas de afazeres (BISWAS, 2023b). Além de tarefas de uso geral, o ChatGPT também pode ser utilizado de uma ampla gama de atividades e empreendimentos humanos tais como: em empresas e organizações governamentais ou do terceiro setor, melhorando campanhas de marketing, reduzindo custos, gerando insights de comportamento de consumidores (AYINDE et al., 2023); na educação, por meio da escrita de artigos e ensaios acadêmicos, elaboração de perguntas e questionários, além de ampliar o escopo de possibilidades de interação entre alunos e professores em sala de aula (MOLLICK; MOLLICK, 2022).

No entanto, conforme o uso do ChatGPT cresce e se populariza, tornam-se cada vez mais evidentes seus impactos sociais, econômicos, éticos e culturais. Na medida em que inputs e dados dos usuários são incorporados aos modelos e algoritmos da ferramenta, vieses dos mais variados tipos também podem ser reforçados, a depender dos objetivos corporativos e da conduta dos grupos empresariais que administram essas ferramentas, o que lança um grande desafio na fronteira dos estudos da interação humano-máquina (VIDHYA; DEVI; A.; MANJU, 2023).

No presente trabalho, são apresentados estudos de caso que envolvam o uso e aplicações de ChatGPT em diferentes contextos e atividades, com ênfase especial na análise crítica das dimensões e dilemas éticos contidos em cada um dos casos analisados. Em seguida, serão listadas, com base na literatura especializada sobre o tema, possíveis soluções para os problemas apontados nos estudos de caso. Por fim, são apresentadas as considerações finais.

2. REVISÃO DE LITERATURA

O ChatGPT é uma ferramenta de Inteligência Artificial Generativa, que são tecnologias capazes de gerar novos conteúdos (textos, códigos, imagens e vídeos) a partir de entradas de dados do usuário, também conhecidas como “prompts”. Essas ferramentas são treinadas com base em uma grande variedade de dados obtidos pela Internet, por meio de uma arquitetura de Redes Neurais Artificiais de tipo Transformer. A ideia de sistemas de IA generativos, por meio do qual seja possível interagir com máquinas por meio de texto e linguagem natural não é nova, e remonta ao início das pesquisas no campo da IA e computação (STAHL; EKE, 2024).

O notório avanço nas pesquisas e, consequentemente nas aplicações que envolvam algum grau de Processamento de Linguagem Natural (NLP), presenciados nos últimos anos tiveram por consequência importantes marcos para acadêmicos e profissionais da indústria da área Interação Humano-Máquina, como por exemplo o comportamento inadequado do bot Tay da Microsoft no Twitter; as infrações de privacidade da Alexa, da Amazon, além das limitações e vieses presentes no próprio ChatGPT (HIRSCHBERG; MANNING, 2015).

No que se refere especificamente ao ChatGPT, diversas pesquisas acadêmicas já foram conduzidas demonstrando algumas das falhas apresentadas

pelo modelo, entre as quais destacam-se: criação de conteúdo falso (alucinações), reiteração dos pontos de vista apresentados por um usuário; reforço de vieses de caráter discriminatório presentes nos dados de treinamento do modelo; e perpetuação de comportamentos que humanos possam tentar evitar em razão de potenciais riscos (EIGNER; HÄNDLER, 2024).

Em razão do extenso uso e atenção que ferramentas como o ChatGPT vem recebendo atualmente, vários estudos têm sido conduzidos para explorar os potenciais benefícios no uso dessa tecnologia, e dilemas éticos que surgem por este motivo. As pesquisas que se dedicam sobre a temática da ética em aplicações de LLM abrangem desde a detecção de condutas imorais até a presença de vieses de cunho discriminatório (VIDHYA, 2023). No entanto, de acordo com Stahl e Eke (2024), muitas dessas pesquisas têm focado em questões específicas, e deixam a desejar no que diz respeito à capacidade de considerar, simultaneamente, benefícios advindos da ferramenta e preocupações éticas resultantes. O autor também propõe que é de interesse coletivo que esse reequilíbrio aconteça, de modo a beneficiar toda a comunidade interessada em tópicos ligados a IA e LLMs.

Considerando o contexto apresentado, e as considerações levantadas por autores e especialistas da área, são apresentados à seguir alguns estudos de caso que demonstram os dilemas éticos que surgem no uso da ferramenta.

1. CHATGPT E O USO NA EDUCAÇÃO

Diversos estudos e publicações destacam os potenciais benefícios do ChatGPT na educação, sugerindo até mesmo diretrizes para sua aplicação em sala de aula. No entanto, as preocupações relacionadas aos chatbots, como a falta de recursos, inadequação tecnológica para tarefas específicas, regulações insuficientes e questões de segurança de dados, ainda não foram suficientemente exploradas. Especificamente, não está claro se o ChatGPT, um chatbot avançado, conseguirá superar essas questões ou se agravará os problemas já presentes em outras soluções de chatbot, o que poderia resultar em reações protetivas rápidas e severas, como a proibição do ChatGPT em redes educacionais de grandes cidades devido ao risco de facilitar trapaças em tarefas escolares (Shen-Berro, 2023).

O estudo conduzido por Tlili et. al (2023) realizou entrevistas com diferentes stakeholders de uma escola (estudantes, professores, diretores) a fim de entender melhor sua relação com a ferramenta. Em geral, todos os entrevistados tinham alguma familiaridade e experiência prévia com o uso de chatbots no contexto educacional. O foco da pesquisa foi compreender a experiência de usuário destes stakeholders com a ferramenta. A pesquisa então aponta 10 cenários que levantam preocupações de cunho ético e educacional.

Um dos cenários apresentados é a possibilidade real de alunos trapacearem em atividades como escrita de artigos ou mesmo ao responder questionários de múltipla escolha. É possível submeter todas essas atividades a uma ferramenta como o ChatGPT e obter respostas relativamente acuradas. Isso se torna mais crítico ainda porque é possível fugir de ferramentas de detecção de plágio de conteúdo gerado por IA simplesmente adicionando algumas palavras ao conteúdo gerado, por exemplo.

Outros cenários apontados na pesquisa demonstram a opacidade nos dados de treinamento desses modelos. Em muitos casos, a resposta gerada à partir de uma pergunta feita pelo usuário era inadequada ou mesmo errada. Em outras situações, a resposta gerada pelo modelo era completamente diferente, ainda que diferentes usuários fizessem exatamente a mesma pergunta.Nesse sentido, outra preocupação que aparece é quanto a clareza sobre como os dados das conversas com os usuários são coletados, armazenados e utilizados pela OpenAI para retreinar seus modelos. Em sua política de privacidade, há menção a essa coleta de dados, e no entanto, a empresa nega que faça qualquer coleta.

Outra questão importante é a falta de feedback emocional das ferramentas de IA, algo que é relativamente comum em uma relação entre professores e alunos, quando há a oportunidade de falar sobre os métodos de aprendizado utilizados e como a experiência de aprendizado em si tem se dado no curso do processo pedagógico.

De maneira geral, as preocupações apontadas por professores e alunos estão relacionadas a questões éticas fundamentais como o encorajamento ao plágio e práticas de trapaça; a falta de acurácia das respostas geradas pelo modelo, e a notável presença de vieses e informações falsas. Além disso, os respondentes da pesquisa se mostraram bastante preocupados com o fato do ChatGPT criar referências falsas quando solicitado a fazer tarefas como essa.

2. CHATGPT E O USO MILITAR

O uso da IA (e do ChatGPT) em contextos militares levanta questões éticas, como a responsabilidade por decisões autônomas, a minimização de danos colaterais e a conformidade com leis internacionais. Essas aplicações requerem uma abordagem cuidadosa para equilibrar benefícios operacionais com preocupações éticas e legais (MACEY-DARE, 2023).

Há uma série de riscos ao usar a inteligência artificial e muitos países e organizações internacionais estão buscando estabelecer normas regulamentações para garantir o uso responsável e ético da inteligência artificial no contexto militar. Os modelos de dados que alimentam os algoritmos podem ter desvios, como erros de identificação facial biométrica ou inexistência de uma avaliação profunda do seu ambiente. No nível civil, especialistas em Inteligência Artificial pediram uma suspensão de seis meses do desenvolvimento desta tecnologia para estudar as suas possíveis consequências. (EURONEWS, 2023)

Quanto à violação dos Direitos Humanos (DH), há um risco significativo de que a IA seja usada para vigiar e controlar populações, infringindo DH fundamentais. Sistemas de IA podem ser usados para coletar e analisar grandes quantidades de dados sobre indivíduos, potencialmente levando a abusos de privacidade e liberdade. Em regimes autoritários, essa tecnologia pode ser empregada para suprimir dissidências e manter o controle social (BISWAS, 2023a).

A legislação internacional atual pode não estar preparada para lidar com os desafios trazidos pelo uso militar de IA. A ausência de normas claras sobre a utilização de tecnologias autônomas em combate cria um vácuo legal, onde ações questionáveis podem ser tomadas sem consequências claras. Há uma necessidade urgente de

desenvolver um quadro legal robusto que regule o uso de IA em contextos militares (ESMAILZADEH, 2023).

Em suma, enquanto o uso de IA, como o Chat GPT, em operações militares pode oferecer vantagens estratégicas significativas, os dilemas éticos são profundos e complexos. É imperativo que essas tecnologias sejam desenvolvidas e implementadas com um forte enfoque ético, garantindo que seu uso esteja alinhado com os princípios fundamentais dos direitos humanos e da dignidade humana. O desenvolvimento de políticas e regulamentações claras será crucial para mitigar os riscos e assegurar que a IA seja usada de maneira responsável e ética no campo militar (MONTEIRO, 2022), (SILVA, 2022).

3. SOLUÇÕES RESPONSÁVEIS

Uma das primeiras e mais importantes considerações a fazer quando se trata de soluções responsáveis é compreender o princípio de que incorporar a tecnologia é preferível a rejeitá-la, ainda que ela apresente desafios éticos significativos. De acordo com o autor, é necessário mais análise e discussão sobre como adotar o ChatGPT de forma responsável em ambientes educacionais, em vez de simplesmente proibi-lo. Um importante fator nessa equação é sempre considerar que os conteúdos gerados pelas IAs generativas podem ser conceitualmente errados e também radicalmente diferentes de usuário para usuário, ainda que estes usem o mesmo prompt como input de dados (KUNG et. al, 2023). Em suma, o potencial revolucionário das ferramentas de IA generativa não podem ser descartados e precisam de atenção especial (TLILI et. al, 2023).

Tlili et. al (2023) ainda ressalta que o componente emocional de IAs generativas ainda aparece como uma limitação da tecnologia. A grande maioria dos chatbots são desenhados para atender a tarefas específicas, e não possuem qualidades socioemocionais, o que pode se apresentar como uma barreira para ampliar a efetividade da interação entre humanos e agentes virtuais, especialmente em um contexto de aprendizado. Por outro lado, há considerações importantes quanto a atribuir características humanas a inteligências artificiais como atribuir a chatbots a autoria ou coautoria sobre um conteúdo, por exemplo.

De acordo com Tlili et. al (2023), o design de soluções de ChatBots como o ChatGPT devem considerar aspectos como a inclusão e usabilidade da ferramenta, além dos aspectos técnicos, como a arquitetura da rede neural de treinamento, o tamanho da base e os dados utilizados no processo. De acordo com o autor, é necessário ir além da privacidade e segurança dos dados pessoais. Deve-se desenvolver diretrizes e estratégias que estejam alinhadas com valores humanos fundamentais e com o sistema legal.

4. CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo demonstrar, por meio de estudos de caso disponíveis na literatura, de que forma dilemas éticos e suas nuances aparecem a partir do uso de ferramentas de IA Generativa, em especial o ChatGPT. Uma análise crítica foi realizada com base nos artigos selecionados, e ao final, buscou-se levantar também com base na literatura especializada, soluções responsáveis, possíveis e viáveis para mitigar alguns dos problemas que surgem em cenários de caso de uso de IAs generativas.

No contexto da educação, o estudo de caso apresentado demonstrou que ainda que os benefícios dessas ferramentas seja relevante, alguns desafios éticos gerados pelo uso da ferramenta, como o plágio ou a criação de conteúdo sem autenticidade, além de não term sido devidamente explorados pela literatura no momento, também podem ser facilmente contornadas pelos alunos, gerando ainda mais problemas eticos. Enquanto no estudo de caso do setor militar, questões como a coleta massiva de dados por sistemas de inteligência artificial, e a dificuldade em responsabilizar e coibir agentes autônomos por decisões de algoritmos de IA levanta profundos questionamentos sobre a segurança e confiabilidade do uso dessas ferramentas em situações reais, como em um campo de batalha. Todos esses desafios evidenciam os limites da legislação internacional para lidar com os problemas advindos desse tipo de tecnologia.

Apesar de todos os desafios apresentados, Tlili et. al (2023) demonstra que o potencial revolucionário das ferramentas de Inteligência Artificial e do ChatGPT não pode ser descartado, e que, por isso, o princípio a ser adotado é de que é melhor adotar a tecnologia de modo responsável, e não ignorá-la completamente, ou desprezar seu uso. Entre as recomendações para utilizar essas tecnologias de modo responsável, estão a consideração do componente socioemocional na interação entre humano-máquina, análise crítica das respostas e conteúdos gerados por IA, além de considerar questões legais, regulatórias e estratégias que se estendem para além do escopo da privacidade e segurança dos usuários.

REFERÊNCIAS

AYINDE, Lateef et al. ChatGPT as an important tool in organizational management: A review of the literature. Business Information Review, v. 40, n. 3, p. 137-149, 2023.

BISWAS, Som. Prospective role of chat gpt in the military: According to chatgpt. Qeios, 2023a.

BISWAS, Som S. Role of chat gpt in public health. Annals of biomedical engineering, v. 51, n. 5, p. 868-869, 2023b.

BOLANDER, Thomas. What do we lose when machines take the decisions?. Journal of Management and Governance, v. 23, p. 849-867, 2019.

DEVLIN, Jacob et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

EIGNER, Eva; HÄNDLER, Thorsten. Determinants of llm-assisted decision-making. arXiv preprint arXiv:2402.17385, 2024.

ELOUNDOU, Tyna et al. Gpts are gpts: An early look at the labor market impact potential of large language models. arXiv preprint arXiv:2303.10130, 2023.

EURONEWS.Que questões levanta o uso da inteligência artificial na guerra. Disponível em: https://pt.euronews.com/2023/05/08/que-questoes-levanta-o-uso-da-inteligencia-artificial-na-guerra.

ESMAILZADEH, Yaser. Potential Risks of ChatGPT: Implications for Counterterrorism and International Security. International Journal of Multicultural and Multireligious Understanding (IJMMU) Vol, v. 10, 2023.

KUNG, Tiffany H. et al. Performance of ChatGPT on USMLE: potential for AI-assisted medical education using large language models. PLoS digital health, v. 2, n. 2, p. e0000198, 2023.

HIRSCHBERG, Julia; MANNING, Christopher D. Advances in natural language processing. Science, v. 349, n. 6245, p. 261-266, 2015.

MACEY-DARE, Rupert. ChatGPT and Generative AI Systems as Military Ethics Advisors. Available at SSRN 4413206, 2023.

MOLLICK, Ethan R.; MOLLICK, Lilach. New modes of learning enabled by ai chatbots: Three methods and assignments. Available at SSRN 4300783, 2022.
MONTEIRO, António Pedro Lopes. A inteligência artificial nos processos de modernização e edificação das capacidades militares do exército: vetor de desenvolvimento, liderança e formação. 2022. 56 f. Monografia (Especialização) - Curso de Departamento de Estudos Pós-Graduados, Instituto Universitário Militar, Pedrouços, 2022

NILSSON, Nils J. The quest for artificial intelligence. Cambridge University Press, 2009.

SHEN-BERRO, J. New York City Schools blocked ChatGPT. Here's what other large districts are doing. Chalkbeat, 2023. Disponível em: https://www.chalkbeat.org/2023/1/6/23543039/chatgpt-school-districts-ban-block-artificial-intelligence-open-ai.

SILVA, Luciano Santos da. A inteligência artificial e sua aplicação no mundo militar. A Lucerna, [s. l], v. 11, n. 1, p. 85-89, 25 mar. 2022.

STAHL, Bernd Carsten; EKE, Damian. The ethics of ChatGPT–Exploring the ethical issues of an emerging technology. International Journal of Information Management, v. 74, p. 102700, 2024.

TLILI, Ahmed et al. What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education. Smart learning environments, v. 10, n. 1, p. 15, 2023.

VIDHYA, N. Gowri et al. Prognosis of exploration on Chat GPT with artificial intelligence ethics. Brazilian Journal of Science, v. 2, n. 9, p. 60-69, 2023.

APÊNDICE L - GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

- Qual o objetivo do estudo descrito pelo artigo?
- Qual o problema/oportunidade/situação que levou a necessidade de realização deste estudo?

- Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?
- Quais os principais resultados obtidos pelo estudo?

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Nome do artigo escolhido:

Requirements practices and gaps when engineering human-centered Artificial Intelligence systems

Qual o objetivo do estudo descrito pelo artigo?

O objetivo do estudo descrito no artigo é investigar as práticas atuais e identificar lacunas na engenharia de requisitos para sistemas de inteligência artificial, centrados no ser humano, especialmente nas fases iniciais de desenvolvimento. Os autores buscam compreender como profissionais estão abordando os aspectos centrados no ser humano durante o processo de engenharia de requisitos e quais diretrizes essenciais devem ser seguidas. Além disso, o estudo visa mapear as diretrizes industriais existentes e comparar as práticas industriais com as lacunas observadas na literatura para propor melhorias que possam orientar o desenvolvimento de sistemas de IA mais éticos, transparentes e centrados no usuário.

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

A necessidade deste estudo surgiu de uma situação em que o desenvolvimento de sistemas de inteligência artificial está se expandindo rapidamente, mas enfrenta desafios significativos relacionados à falta de práticas padronizadas na engenharia de requisitos para IA, especialmente no que diz respeito a aspectos humanos e éticos.

A oportunidade identificada pelos autores é melhorar a forma como os requisitos são capturados, especificados e gerenciados para garantir que os sistemas de IA atendam não apenas aos requisitos técnicos, mas também às necessidades humanas, como transparência, confiança, justiça e controle do usuário.

A situação problemática que impulsionou o estudo é o uso crescente de IA em setores sensíveis, onde erros ou vieses nos sistemas de IA podem levar a consequências negativas para os usuários, como discriminação ou falhas na interpretação de dados importantes. O estudo visa, então, identificar práticas inadequadas, lacunas e oportunidades de melhoria no processo de engenharia de requisitos para o desenvolvimento de uma IA mais ética e centrada nas necessidades dos usuários.

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

Para conduzir o estudo, os autores utilizaram uma metodologia baseada em pesquisa bibliográfica e em um levantamento com profissionais da indústria. As etapas seguidas foram fazer uma revisão sistemática da literatura, fazer um mapeamento de diretrizes industriais, fazer uma pesquisa com profissionais da indústria, e a análise dos dados.

Quais os principais resultados obtidos pelo estudo?

O artigo revelou que, embora aspectos centrados no ser humano, como transparência, controle do usuário e mitigação de vieses, sejam considerados relevantes na engenharia de requisitos para IA, sua implementação prática ainda é limitada. A maioria dos profissionais utiliza ferramentas genéricas que não atendem adequadamente às necessidades específicas de sistemas de IA, evidenciando a oportunidade de desenvolver ferramentas de engenharia de requisitos mais apropriadas. Os principais desafios identificados foram a especificação de requisitos de dados, a necessidade de fornecer explicações claras das previsões dos sistemas de IA, e a carência de padrões éticos sólidos. Além disso, aspectos como feedback e controle do usuário, embora essenciais, são pouco considerados. Os resultados destacam uma lacuna entre teoria e prática, sugerindo que as práticas da indústria frequentemente não contemplam temas abordados na literatura, como a necessidade de planejar para erros do sistema e avaliar a viabilidade da IA antes de sua adoção.

Esses resultados sugerem a necessidade de novas ferramentas e frameworks que integrem diretrizes centrados no ser humano de maneira prática na engenharia de requisitos para IA.

APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

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

2 Regressão (RNA)

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

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

Informações:

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

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/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

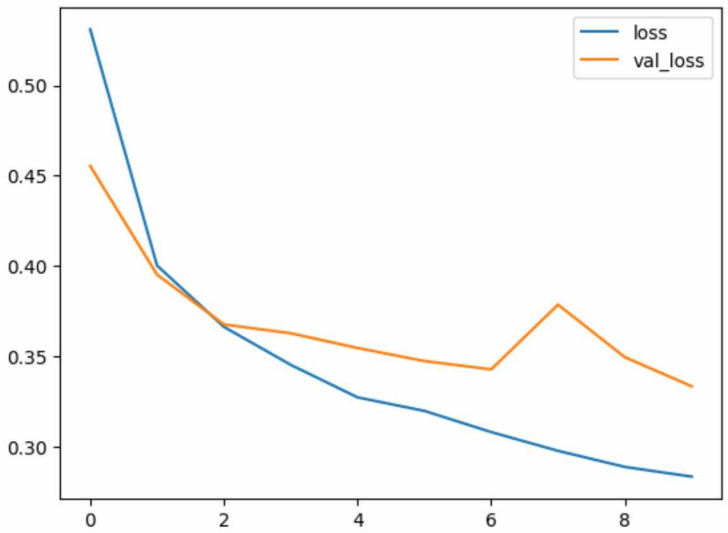
1 Classificação (RNA)

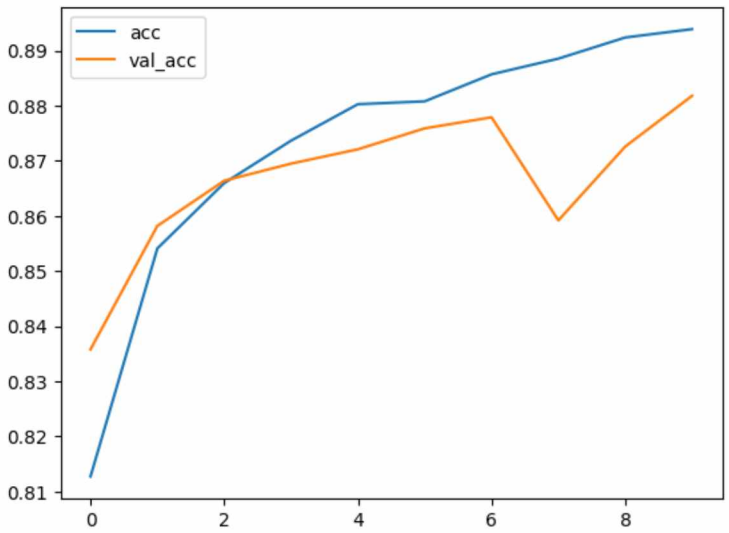
Código

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

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

x_train, x_test = x_train/255.0, x_test/255.0

i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)

model = tf.keras.models.Model(i, x)

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

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

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

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

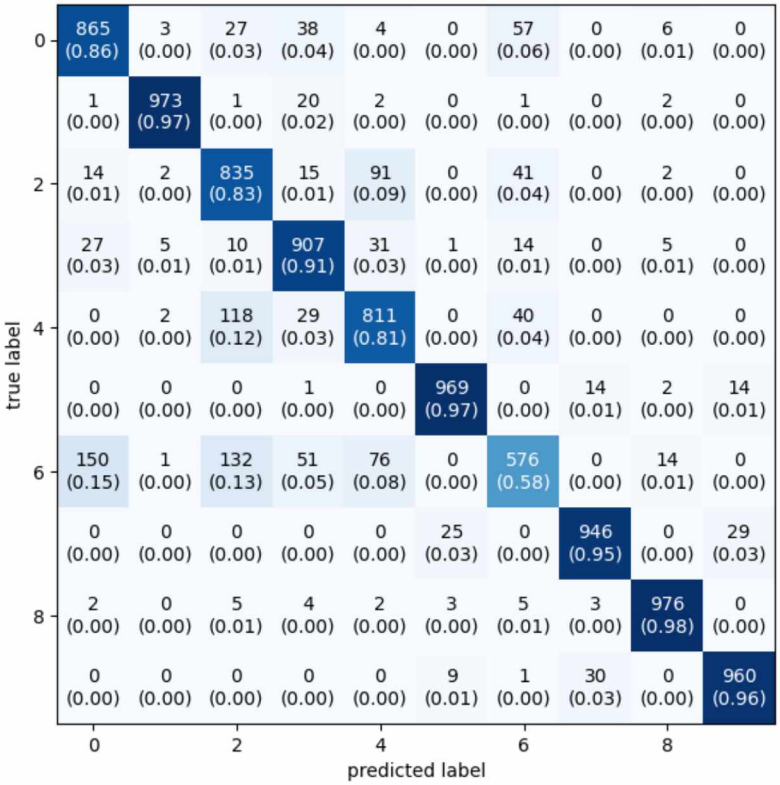
```
# Avaliar o modelo com a base de teste
print( model.evaluate(x_test, y_test) )

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

Resultado





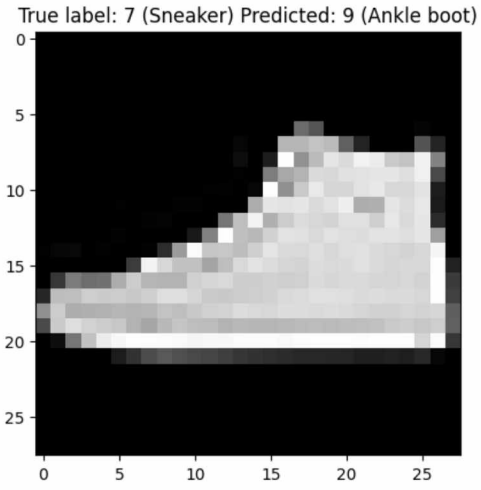

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy:
0.8825 - loss: 0.3272
[0.3336341381072998, 0.8817999958992004]
```



Código

```
misclassified = np.where(y_pred != y_test)[0]
descricao_classes_en = ['T-
shirt/top','Trouser','Pullover','Dress','Coat','Sandal','Shirt','Snea
ker','Bag','Ankle boot']
def getDescricaoClasse(i):
    return descricao_classes_en[i]
i = np.random.choice(misclassified)

plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s (%s) Predicted: %s (%s)" % (y_test[i],
getDescricaoClasse(y_test[i]), y_pred[i],
getDescricaoClasse(y_pred[i])))
```

Resultado



No gráfico de perda, vemos que a perda com os dados de treino diminuiu rapidamente, enquanto com os dados de validação chegou até a aumentar em uma das épocas. Como os valores de perda ainda estavam diminuindo, talvez o treinamento com mais épocas gere um melhor resultado.

No gráfico de acurácia também verificamos que com os dados de treino o resultado melhora rapidamente, e com os dados de validação ocorre uma oscilação em determinada época. Do mesmo modo, como o gráfico mostra a acurácia melhorando a cada época, sem uma estabilização, é provável que o aumento das épocas auxilie a melhorar o resultado.

Na exibição de uma classificação errada, vemos um tênis (sneaker) que foi incorretamente classificado como um tipo de bota (ankle boot). As imagens serem de baixa definição e o modelo do tênis (com a canela mais alta) deve ter contribuído para este caso de classificação incorreta.

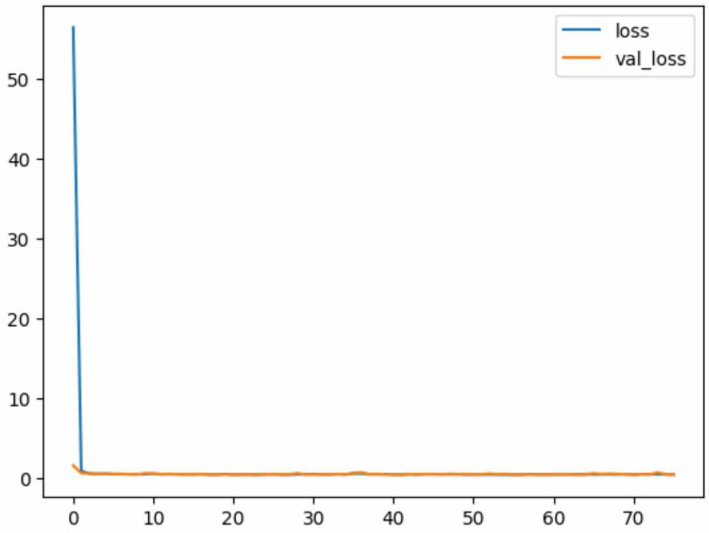
2 Regressão (RNA)

Código

```
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

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

data.columns = [
  'acidez_fixa', # 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
]

X = data[ data.columns[:-1] ].astype(float)
Y = data[ data.columns[-1] ].astype(float)
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y,
                                        test_size=0.25)

# 3 camadas
i = tf.keras.layers.Input(shape=(11,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)

model = tf.keras.models.Model(i, x)

# Criação de funções para as métricas R2 e RMSE serem inseridas no
modelo
def rmse(y_true, y_pred):
  return backend.sqrt(backend.mean( backend.square(y_pred - y_true),
axis=-1) )

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

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

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

# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
                            monitor='val_loss',
                            patience=20,
                            restore_best_weights=True)

r = model.fit(x_train, y_train,
              epochs=1500,
              validation_data=(x_test, y_test),
              callbacks=[early_stop])


plt.plot( r.history["loss"], label="loss" )
plt.plot( r.history["val_loss"], label="val_loss" )
plt.legend()

# Predição
y_pred = model.predict(x_test).flatten()

# Cálculo das métricas de acurácia: mse, r2 e rmse
mse  = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2   = r2_score(y_test, y_pred)

# Resultados das métricas de acurácia
print("mse     = ", mse)
print("rmse    = ", rmse)
print("r2      = ", r2)
```

Resultado




```
mse      =  0.4233919871052984
rmse     =  0.6506857821600979
r2       =  0.3350669133300508
```

Verificando o gráfico de perda, percebemos que apenas na primeira época a perda com os dados de treino foi alta, diminuindo bastante logo em seguida. Com os dados de validação, as diferenças foram sutis de uma época para outra, não sendo obtida melhora muito significativa ao longo do treinamento. A configuração de "Early Stop" também fez com que o treinamento fosse interrompido bem antes das 1500 épocas definidas como limite.

Pelos valores verificados na avaliação, o modelo precisa de melhorias. O R2 baixo (0.33) mostra que o modelo tem dificuldade em explicar os dados. Alterar as camadas da rede, normalizar os dados e ajustar hiper parâmetros são técnicas que podem auxiliar a obter melhores resultados.

3 Sistemas de Recomendação

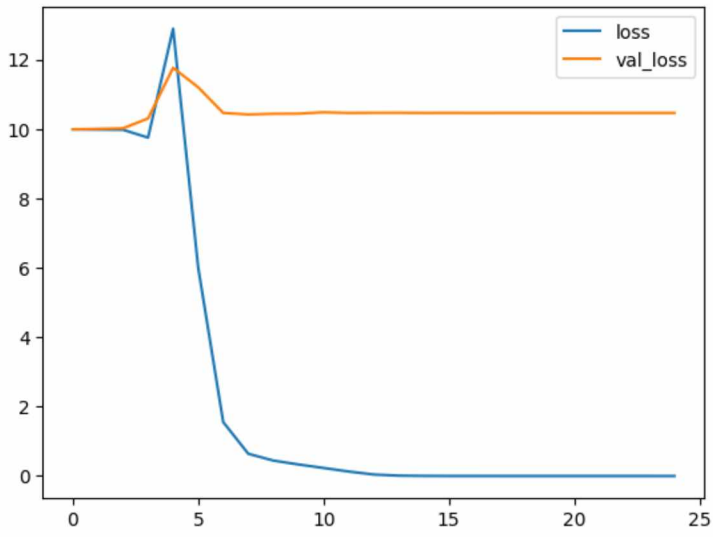
Código

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

from sklearn.utils import shuffle

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

```
!wget --no-check-certificate
'https://drive.google.com/uc?export=download&id=1BIE2y-
eC_4tb_x5AWmwwFf93aBf4e8CX' -O /tmp/Base_livros.csv

df = pd.read_csv("/tmp/Base_livros.csv", encoding='utf-8')

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

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

# Dimensões
N = len(set(df.new_ID_usuario))
M = len(set(df.new_ISBN))

# dimensão do embedding
K = 10

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

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

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

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

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

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

usuario_ids, livros_isbn, ratings = shuffle(df.new_ID_usuario,
df.new_ISBN, df.Notas)

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

train_user = usuario_ids[:Ntrain]
train_livros = livros_isbn[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = usuario_ids[Ntrain:]
test_livros = livros_isbn[Ntrain:]
test_ratings = ratings[Ntrain:]

# centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating

epochs = 25
r = model.fit(
    x=[train_user, train_livros],
```

```
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso
    validation_data=([test_user, test_livros], test_ratings)
)

plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

# Gerar o array com o usuário único
# repete a quantidade de filmes
input_usuario = np.repeat(a=6574, repeats=M)
film = np.array(list(set(livros_isbn)))

preds = model.predict( [input_usuario, film] )

# descentraliza as predições
rat = preds.flatten() + avg_rating
rat = np.clip(rat, 0, 10)

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

titulo = df[df['new_ISBN'] == film[idx]]['Titulo'].iloc[0]
print("Recomendação: Livro - ", film[idx], "-", titulo, " / ",
rat[idx] , "*")
```

Resultado




```
Recomendação: Filme -  174 - Legal Tender  /  10.0 *
```

No gráfico de perda, verificamos que, com os dados de treinamento, a perda ficou bem baixa, e com os dados de validação ficou muito maior. Isso é um indicativo

de overfitting, ou seja, o modelo não está muito ajustado aos dados de treinamento, conseguindo generalizar para outro conjunto de dados

No exemplo de recomendação, foi selecionado um usuário, e o modelo recomendou o livro "Legal Tender" para ele, prevendo uma possível nota "10".

4 Deepdream

Código

```
import tensorflow as tf
import numpy as np

import matplotlib as mpl

import IPython.display as display
import PIL.Image

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>'))

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

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)
```

Resultado

O main loop é o ciclo principal do DeepDream. Nele, são feitas várias iterações modificando a imagem original, gerando outra formada por repetições de padrões, formas e cores detectados pela rede neural. O modelo "sonha" com as características que ele reconhece nas imagens e tenta realçá-las de maneira exagerada, gerando uma imagem com visual mais surreal.

Levar até uma oitava faz com que a nova imagem tenha mais variações, sendo mais complexa que a imagem gerada anteriormente no main loop. As texturas e padrões gerados são mais diversos ao levar o modelo até uma oitava.

APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

- Explicação do **contexto e o 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

Conjunto de Dados Brutos

Os dados apresentados foram extraídos do Balanço Energético Nacional 2023, fornecido pela Empresa de Pesquisa Energética (EPE). Eles mostram o consumo absoluto de eletricidade por setor econômico no Brasil entre 2020 e 2022.
https://anuario.ibge.gov.br/images/aeb/2023/s4/2_pdf/s4t3105.pdf

Tabela 4.3.1.5 - Distribuição percentual do consumo de eletricidade, segundo os setores - 2020-2022

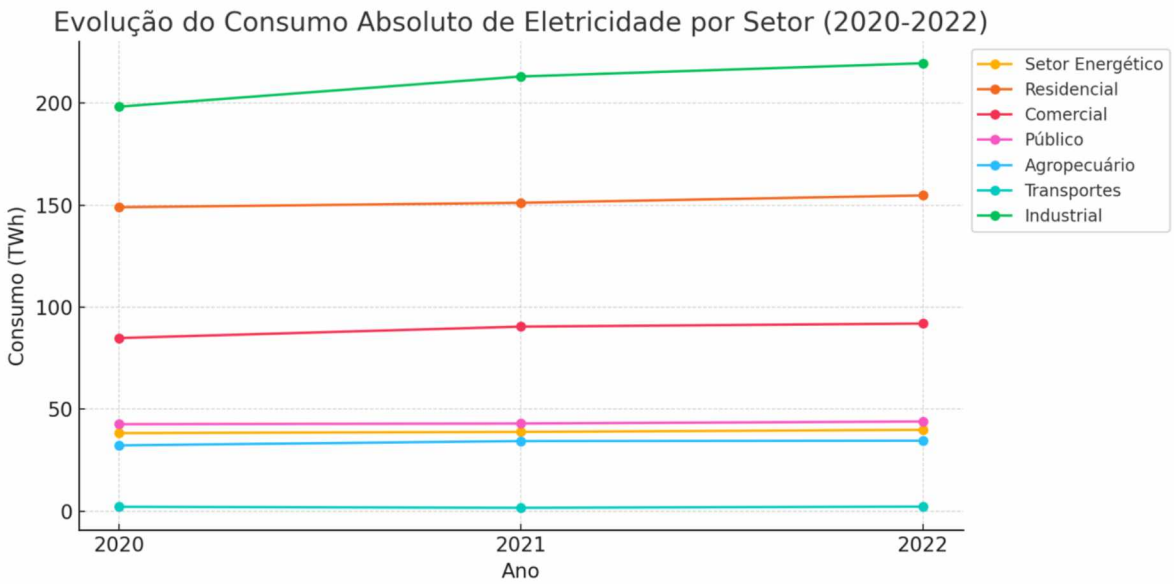
Setores	Distribuição percentual do consumo de eletricidade (%)		
	2020	2021	2022
Total (TWh)	**547,7**	**572,8**	**586,1**
Total (%)	**100,0**	**100,0**	**100,0**
Setor energético	7,0	6,8	6,8
Residencial	27,2	26,4	26,4
Comercial	15,5	15,8	15,7
Público	7,8	7,5	7,5
Agropecuário	5,9	6,0	5,9
Transportes	0,4	0,3	0,4
Industrial	36,2	37,3	37,4

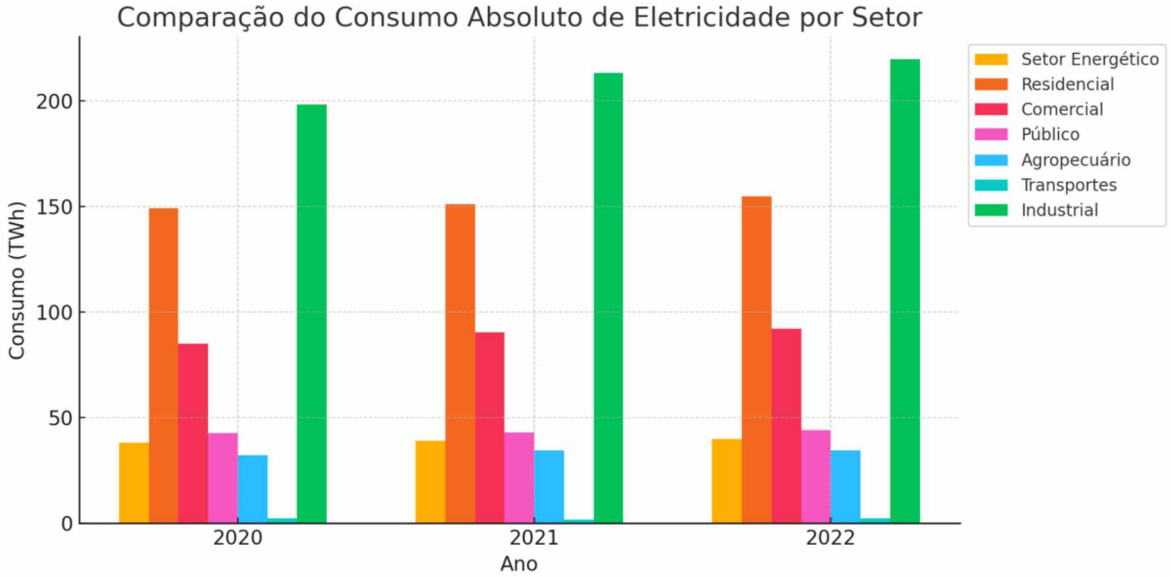
Fonte: Balanço energético nacional 2023: ano base 2022. Brasília, DF: Empresa de Pesquisa Energética - EPE, [2024]. Disponível em: https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/balanco-energetico-nacional-2023. Acesso em: jan. 2024.

Contexto e Público-Alvo

Este estudo busca entender o comportamento do consumo de eletricidade no Brasil e seus impactos em setores-chave. O público-alvo inclui formuladores de políticas energéticas, empresas do setor elétrico, economistas e consumidores interessados na eficiência energética.

Visualização dos Dados





A escolha das visualizações e ferramentas foi feita para melhor representar a evolução do consumo de eletricidade no Brasil e facilitar a análise dos dados.

Gráfico de Linhas (Valores Absolutos - TWh)
Justificativa: O gráfico de linhas destaca a evolução do consumo absoluto de eletricidade ao longo dos anos, permitindo identificar aumentos ou reduções no consumo de cada setor.
Benefício: Ideal para comparar setores distintos e analisar variações específicas em cada um deles.

Gráfico de Barras (Comparação por Ano - TWh)
Justificativa: Útil para visualizar a distribuição do consumo absoluto entre os setores em um determinado ano, facilitando a comparação direta.
Benefício: Permite uma análise rápida de qual setor consome mais eletricidade em números absolutos.

A ferramenta utilizada para a visualização dos dados foi Python, com as bibliotecas Matplotlib e Seaborn, permitindo personalizar completamente os gráficos, ajustando cores, tamanhos e estilos. Essas escolhas garantem que os dados sejam apresentados de forma clara e impactante, facilitando a interpretação para formuladores de políticas, analistas e o público interessado no consumo de eletricidade.

Narrativa/Storytelling

Nos últimos três anos, observa-se um crescimento contínuo no consumo de eletricidade no setor industrial, que representa a maioria da demanda. O consumo residencial também tem aumentado, sugerindo mudanças nos padrões de uso de energia. O setor de transportes apresenta oscilações, indicando possível variação na eletrificação do segmento.

Possíveis Ações:

- Implementação de programas de eficiência energética na indústria para otimizar o alto consumo.
- Incentivo ao uso de fontes renováveis para suprir o aumento da demanda no setor residencial.
- Expansão da eletrificação no setor de transportes, garantindo maior estabilidade no consumo.

APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

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

2) Compare a representação de dois modelos vetoriais

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

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

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

B – RESOLUÇÃO

1) Algoritmo Genético

Código

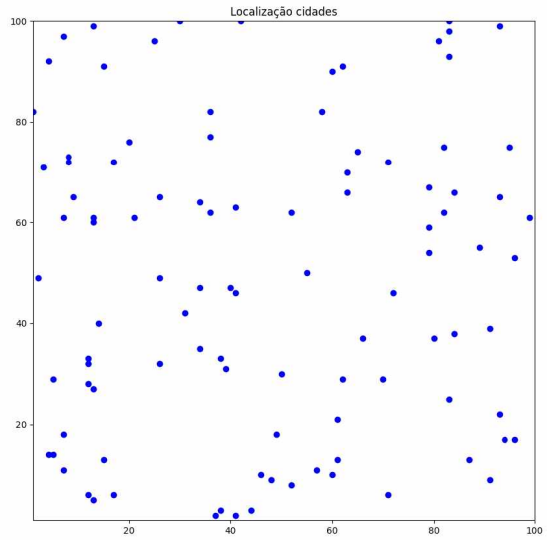
```
import random
import matplotlib.pyplot as plt
import math
import copy

X_LIM_MIN = 1
X_LIM_MAX = 100
Y_LIM_MIN = 1
Y_LIM_MAX = 100

QUANTIDADE_CIDADES = 100
TAMANHO_POPULACAO = 100

def gera_localizacao_cidades(num_cidades=100):
  array_cidades = []
  for i in range(num_cidades):
    coord_x = random.randint(X_LIM_MIN,X_LIM_MAX)
    coord_y = random.randint(Y_LIM_MIN,Y_LIM_MAX)
    ponto_cidade = [coord_x,coord_y]
    array_cidades.append(ponto_cidade)

  return array_cidades

def imprime_pontos(array_cidades, titulo='Localização cidades',
imprimir_linhas=False):
  x = [p[0] for p in array_cidades]
  y = [p[1] for p in array_cidades]

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

  plt.scatter(x, y, color='blue', marker='o')
```

```
#  if imprimir_linhas:
#    plt.plot(x, y, color='red', linestyle='-', linewidth=1)
  if imprimir_linhas:
    for i in range(len(x) - 1):
      plt.plot([x[i], x[i+1]], [y[i], y[i+1]], color='red',
linestyle='-', linewidth=1)
      plt.text(x[i], y[i], str(i + 1), color='black', fontsize=8,
ha='right', va='bottom')

  plt.xlim(X_LIM_MIN, X_LIM_MAX)
  plt.ylim(Y_LIM_MIN, Y_LIM_MAX)

  plt.title(titulo)

  plt.show()

localizacao_cidades = gera_localizacao_cidades(QUANTIDADE_CIDADES)

imprime_pontos(localizacao_cidades)

def populacao_inicial(tamanho):
  resultado = [];
  for i in range(tamanho):
    ordem_visitacao_cidades = random.sample(range(1,
QUANTIDADE_CIDADES), QUANTIDADE_CIDADES-1)
    resultado.append([0] + ordem_visitacao_cidades + [0])
  return resultado

def calcula_distancia_total(array_cidades_ordenado):
  distancia_total = 0
  for i in range(len(array_cidades_ordenado)-1):
    cidade1 = localizacao_cidades[array_cidades_ordenado[i]]
    cidade2 = localizacao_cidades[array_cidades_ordenado[i+1]]
    #print(cidade1,cidade2)
    distancia = math.sqrt((cidade2[0] - cidade1[0])**2 + (cidade2[1]
- cidade1[1])**2)
    distancia_total += distancia
  return distancia_total

def avaliacao(populacao):
  fit = []
  for array_ordem_cidades in populacao:
    distancia_total = calcula_distancia_total(array_ordem_cidades)

    fit.append(distancia_total)
  return fit

#Gera a população inicial e exibe a avaliação das soluções candidatas
p0 = populacao_inicial(TAMANHO_POPULACAO)
a0 = avaliacao(p0)

def buscaMelhor(geracao_avaliar):
  ava = avaliacao(geracao_avaliar)
  melhor = 0
  for i in range(0,len(ava)):
    if ava[i] < ava[melhor]:
      melhor = i
  return copy.deepcopy(geracao_avaliar[melhor])

melhor_solucao = buscaMelhor(p0)
```

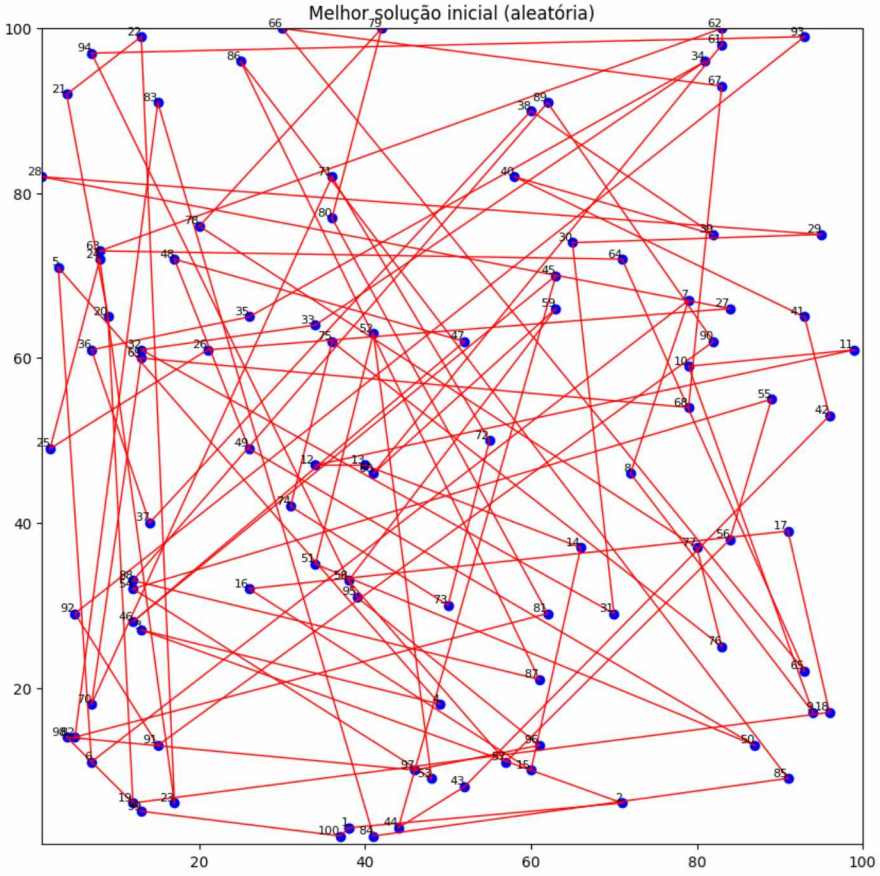
```
print(f'Distância total: {calcula_distancia_total(melhor_solucao)}')
cidades_imprimir = []
for i in melhor_solucao:
  cidades_imprimir.append(localizacao_cidades[i])
imprime_pontos(cidades_imprimir, 'Melhor solução inicial
(aleatória)', True)

def selecao(pop, p=0.70):
  sorteados = []
  distancia = []
  n = 3
  for i in range(n):
    ind = random.randrange(0,len(pop));
    while ind in sorteados:
      ind = random.randrange(0,len(pop));

    sorteados.append(ind)
    distancia.append(calcula_distancia_total(pop[ind]))

  candidatos_ordenados = sorted(zip(distancia, sorteados))

  if random.random() < p:  # Probabilidade "p" de selecionar a melhor
opção
    return candidatos_ordenados[0][1]  # Retorna o melhor candidato
  else:
    # Senão, retorna um dos outros candidatos, para maior diversidade
de soluções
    return random.choice([candidate[1] for candidate in
candidatos_ordenados[1:]])

def cruzamento(pop, qtd, nova):
  for _ in range(qtd):
    indA = selecao(pop)
    indB = indA;
    while indA == indB:
      indB = selecao(pop)

    # Define o pedaço do elemento pai que vai ser usado no cruzamento
    ponto_corte = random.randrange(1, 100)
    qtd_elementos_corte = random.randrange(1,6)
    if ponto_corte + qtd_elementos_corte > QUANTIDADE_CIDADES:
      qtd_elementos_corte = QUANTIDADE_CIDADES - ponto_corte

    individuoA = copy.deepcopy(pop[indA])
    individuoB = copy.deepcopy(pop[indB])

    trecho_A =
individuoA[ponto_corte:ponto_corte+qtd_elementos_corte]
    nova_solucao = individuoB
    for i in trecho_A:
      nova_solucao.remove(i)
    nova_solucao = nova_solucao[0:ponto_corte] + trecho_A +
nova_solucao[ponto_corte:]

    nova.append(nova_solucao)

def mutacao(pop, qtd, nova):
  qtdSaida = len(nova) + qtd
  while len(nova) < qtdSaida:
    indA = selecao(pop)
    solucao = copy.deepcopy(pop[indA])
```

```
    pos1 = random.randrange(1,len(solucao)-1);
    pos2 = pos1;
    while pos1 == pos2:
      pos2 = random.randrange(1,len(solucao)-1);

    aux = solucao[pos1]
    solucao[pos1] = solucao[pos2]
    solucao[pos2] = aux

    nova.append(solucao)

geracao = copy.deepcopy(p0)

NUM_GERACOES = 1000
numGeracoes = NUM_GERACOES;
while numGeracoes > 0:
  nova = []
  melhor_solucao_atual = buscaMelhor(geracao)
  nova.append(melhor_solucao_atual)

  cruzamento(geracao, 95, nova)
  mutacao(geracao, 4, nova)
  numGeracoes-=1

  av = avaliacao(nova);
  nova2 = sorted(zip(av,nova),reverse=False)
  geracao = [x for _,x in nova2]

print(f'A melhor solucao encontrada: {geracao[0]}')
print(f'Distância total: {calcula_distancia_total(geracao[0])}')

cidades_imprimir = []
for i in geracao[0]:
  cidades_imprimir.append(localizacao_cidades[i])
imprime_pontos(cidades_imprimir, 'Solução final', True)
```

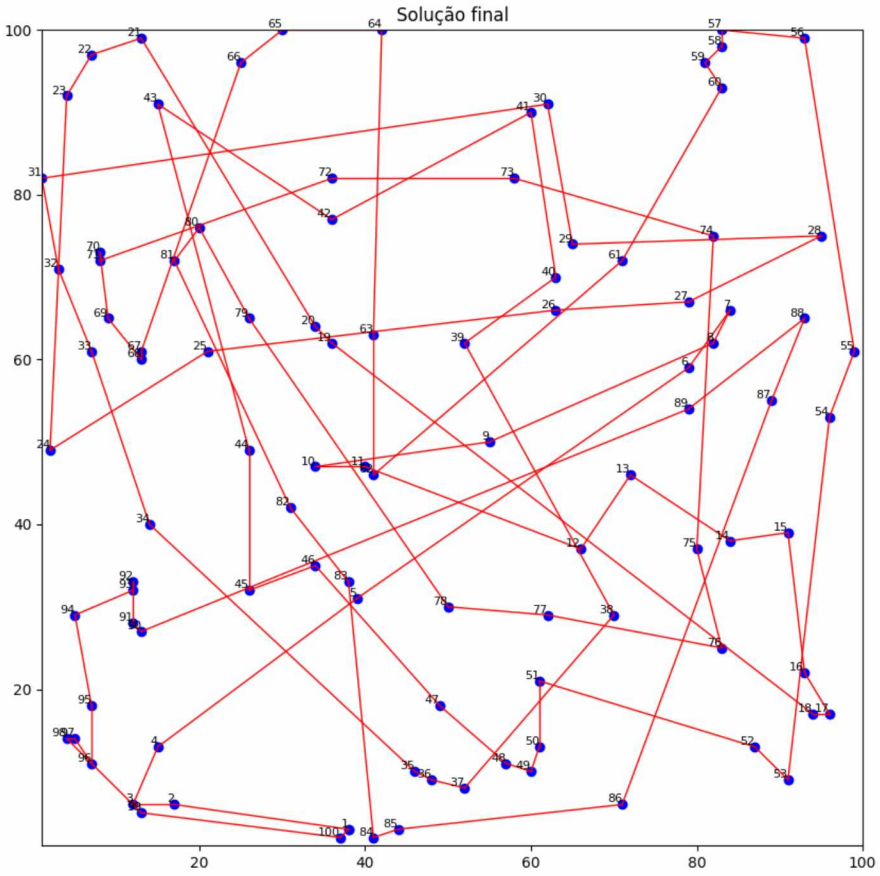
Resultado



```
Distância total: 4796.407848131166
```




```
Distância total: 1897.1546734620565
```



2) Compare a representação de dois modelos vetoriais

Código

```
import spacy
import nltk
import numpy as np
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Carregar o modelo spaCy para o português
nlp = spacy.load("pt_core_news_lg")

nltk.download('stopwords')
nltk.download('punkt_tab')

texts = [
    "O cachorro corre no parque.",
    "O gato brinca com a bola.",
    "A criança corre atrás do cachorro.",
    "O pássaro voa alto no céu.",
    "Os carros estão estacionados na rua.",
    "O parque está cheio de flores coloridas."
]

# Função para pré-processar o texto: tokenização e remoção de
stopwords
def preprocess(text):
    stop_words = set(stopwords.words('portuguese'))
    tokens = word_tokenize(text.lower())
    filtered_tokens = [word for word in tokens if word.isalpha() and
word not in stop_words]
    return filtered_tokens

# Pré-processando os textos
processed_texts = [preprocess(text) for text in texts]

texts_str = [" ".join(text) for text in processed_texts]

# Gerando os vetores TF-IDF
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(texts_str)

tfidf_vectors = tfidf_matrix.toarray()

# Função para obter os vetores de palavras com o spaCy
def get_spacy_vector(word):
    # Usamos o modelo spaCy para obter o vetor da palavra
    token = nlp(word)
    return token.vector

# Gerando os vetores spaCy para as palavras do TF-IDF
spacy_vectors = [get_spacy_vector(word) for word in
vectorizer.get_feature_names_out()]

# Função para aplicar PCA e plotar
def plot_pca(vectors, labels, title):
    pca = PCA(n_components=2)
```

```
    pca_result = pca.fit_transform(vectors)

    plt.figure(figsize=(8, 6))
    plt.scatter(pca_result[:, 0], pca_result[:, 1])

    for i in range(pca_result.shape[0]):
        plt.text(pca_result[i, 0], pca_result[i, 1], labels[i],
fontsize=12)

    plt.title(title)
    plt.xlabel('Componente Principal 1')
    plt.ylabel('Componente Principal 2')
    plt.show()

# Comparando as representações - Visualizando as palavras
plot_pca(tfidf_vectors, vectorizer.get_feature_names_out(), "PCA -
TF-IDF")
plot_pca(spacy_vectors, [word for word in
vectorizer.get_feature_names_out()], "PCA - spaCy")
```

Resultado