

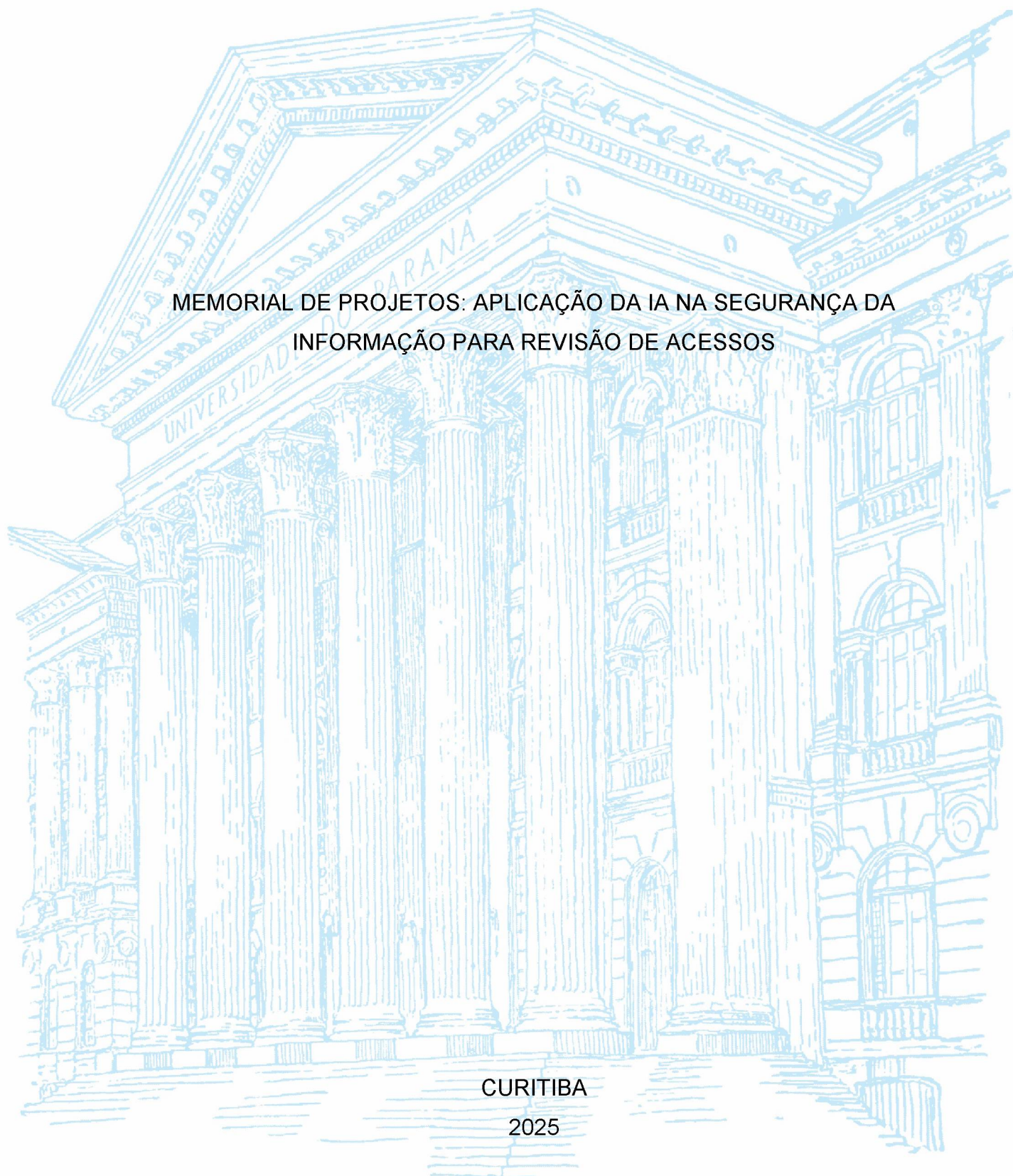
UNIVERSIDADE FEDERAL DO PARANÁ

PRISCILA LIE KAWAMURA

MEMORIAL DE PROJETOS: APLICAÇÃO DA IA NA SEGURANÇA DA
INFORMAÇÃO PARA REVISÃO DE ACESSOS

CURITIBA

2025



PRISCILA LIE KAWAMURA

MEMORIAL DE PROJETOS: APLICAÇÃO DA IA NA SEGURANÇA DA
INFORMAÇÃO PARA REVISÃO DE ACESSOS

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

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2025

TERMO DE APROVAÇÃO

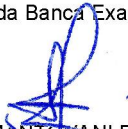
Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **PRISCILA LIE KAWAMURA**, intitulada: **MEMORIAL DE PROJETOS: APLICAÇÃO DA IA NA SEGURANÇA DA INFORMAÇÃO PARA REVISÃO DE ACESSOS**, que após terem inquirido a aluna 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, 29 de Outubro de 2025.



JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora



RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este Memorial de Projetos aborda a aplicação de conceitos de Inteligência Artificial (IA) para a otimização de processos em segurança da informação, com foco em um estudo de caso sobre a revisão de acessos. O projeto detalha a utilização de algoritmos de resolução de problemas por meio de busca visando mitigar riscos do processo manual de revisão de acessos. O problema é modelado por meio de uma estrutura de busca, cujo espaço de estados considera o usuário, seu time, a estrutura hierárquica e seus pares. A solução implementada consiste em um algoritmo de *score* que avalia as permissões com base em múltiplos fatores contextuais, recomendando a revogação de acessos quando o modelo atinge um limiar de confiança de 80%. Os resultados demonstram a eficácia da abordagem na mitigação de riscos de segurança e na redução da intervenção manual, validando como os fundamentos teóricos da IA, explorados ao longo do curso, são essenciais para o desenvolvimento de soluções corporativas robustas e responsáveis.

Palavras-chaves: inteligência artificial; resolução de problemas; algoritmos de busca; gestão de acessos; segurança da informação.

ABSTRACT

This Project Memorial addresses the application of Artificial Intelligence (AI) concepts for process optimization in information security, focusing on a case study on access review. The project details the use of problem-solving by search algorithms aiming to mitigate risks of the manual access review process. The problem is modeled through a search structure whose state space considers the user, their squad, the hierarchical structure, and their peers. The implemented solution consists of a score algorithm that evaluates permissions based on multiple contextual factors, recommending the revocation of access when the model reaches an 80% confidence threshold. The results demonstrate the approach's effectiveness in mitigating security risks and reducing manual intervention, validating how the theoretical foundations of AI, explored throughout the course, are essential for developing robust and responsible corporate solutions.

Keywords: artificial intelligence; problem solving; search algorithms; identity access management; information security.

SUMÁRIO

1 PARECER TÉCNICO.....	7
REFERÊNCIAS.....	11
APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL.....	12
APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA.....	17
APÊNDICE 3 – LINGUAGEM R.....	32
APÊNDICE 4 – ESTATÍSTICA APLICADA I.....	39
APÊNDICE 5 – ESTATÍSTICA APLICADA II.....	49
APÊNDICE 6 – ARQUITETURA DE DADOS.....	64
APÊNDICE 7 – APRENDIZADO DE MÁQUINA.....	73
APÊNDICE 8 – DEEP LEARNING.....	113
APÊNDICE 9 – BIG DATA.....	141
APÊNDICE 10 – VISÃO COMPUTACIONAL.....	144
APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	206
APÊNDICE 12 – GESTÃO DE PROJETOS DE IA.....	214
APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	217
APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING.....	243
APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	250

1 PARECER TÉCNICO

A gestão de acessos em segurança da informação é um desafio constante para as organizações, especialmente no que tange ao processo de revisão de privilégios. A abordagem manual, tradicionalmente utilizada, apresenta riscos significativos, como a manutenção de acessos desnecessários que violam o princípio do menor privilégio. Este parecer técnico apresenta uma solução baseada em Inteligência Artificial para otimizar e automatizar a revisão de acessos, mitigando riscos e aumentando a eficiência operacional

Este projeto teve como objetivo a otimização do processo de revisão de acessos em uma instituição financeira. A Gestão de Identidade e Acesso (IAM - *Identity and Access Management*), consiste em uma estrutura de políticas e tecnologias que garante que as pessoas tenham o acesso apropriado aos recursos necessários (Singh et al., 2023). A IAM se sustenta em quatro pilares fundamentais:

- Administração (Gerenciamento de Identidade): Foca no ciclo de vida das dentro da empresa, desde sua criação e provisionamento inicial até a sua desativação segura. Garante a automação de processos de entrada, movimentação e saída de colaboradores, conhecido como JML (*Joiner, Mover, Leaver*).
- Autenticação: É o processo que verifica se um usuário é quem ele diz ser, através de credenciais como senhas, tokens ou biometria. Tecnologias como Autenticação Multifator (MFA) são essenciais para fortalecer essa verificação.
- Autorização: Após a autenticação, este pilar determina quais recursos um usuário pode acessar e quais ações pode executar. Aqui, são aplicadas políticas de controle de acesso, como o RBAC (*Role-Based Access Control*), que concede permissões com base na função do usuário
- Auditoria: Garante a conformidade e a segurança contínua, monitorando e registrando as atividades dos usuários para identificar riscos e atividades suspeitas.

Dentro desses processos citados, temos o processo de revisão de acesso manual que está sujeito a falhas, como decisões tomadas sem contexto ou sem a devida atenção do revisor na manutenção de privilégios desnecessários, o que compromete o princípio do menor privilégio. O princípio do menor privilégio faz com que o colaborador tenha apenas os acessos necessários para conseguir realizar

suas atividades diárias, prevenindo acessos indevidos ou ações indevidas que não cabe ao colaborador realizar (Farman, 2024).

A manutenção de privilégios desnecessários, mencionada anteriormente, compromete diretamente o Princípio do Menor Privilégio (PoLP), um dos pilares da segurança da informação. A implementação rigorosa deste princípio, aliada à segregação de funções, é fundamental para minimizar a superfície de ataque de uma organização. O processo manual de revisão de acessos pode falhar exatamente neste ponto por diversas razões, especialmente pela ausência de mecanismos automatizados e baseados em dados para avaliação de riscos e concessões de permissões (Nobi et al., 2022):

- **Fadiga de Decisão:** Gestores com dezenas de subordinados diretos precisam revisar centenas, senão milhares, de permissões em ciclos curtos. Isso leva à "aprovação carimbo" (*rubber-stamping*), onde as permissões são mantidas sem uma análise criteriosa.
- **Falta de Contexto:** Um gestor pode não ter o conhecimento técnico detalhado para avaliar se uma permissão específica, como acesso a um determinado banco de dados ou a uma função de administrador em um sistema legado, ainda é necessária para as atividades diárias de um colaborador.
- **Escalabilidade:** Em corporações com alta rotatividade e projetos dinâmicos, o volume de solicitações e revogações de acesso torna a revisão manual um gargalo operacional, aumentando a probabilidade de erros.

Durante a pandemia do covid-19, com o aumento do trabalho remoto o que agravou a situação, destacando a necessidade de soluções de IAM mais eficazes para autenticação remota. Consequentemente, 77% das empresas aumentaram seu orçamento para IAM a fim de mitigar os riscos de cibersegurança. O processo manual está na intersecção dos pilares de Autorização e Auditoria e é onde se mostra mais vulnerável. Para enfrentar esse desafio, o problema foi modelado usando uma estrutura de busca: o espaço de estados representou todas as combinações possíveis de permissões de um usuário, de sua *squad*, de sua posição hierárquica e de seus pares; as ações consistiram em manter ou revogar acessos, avaliadas por meio de uma função de *score* ou árvore de busca.

A necessidade de soluções de IAM mais eficazes, agravada pelo trabalho remoto, está alinhada à evolução do paradigma de segurança para a Arquitetura *Zero Trust* (ZTA). Este modelo abandona a ideia de um perímetro de rede seguro e opera sob o princípio de "nunca confiar, sempre verificar", onde cada solicitação de acesso é rigorosamente autenticada e autorizada. Conforme aponta a pesquisa de Hussain et al. (2022), a ZTA exige uma verificação dinâmica e contextual, tornando a revisão manual de privilégios estáticos um método inseguro e inadequado para os desafios de segurança modernos, mostrando a necessidade da modernização dos processos usados atualmente.

A solução implementada utilizou um algoritmo de *score* baseado em múltiplos fatores como usuários da mesma *squad*, área ou departamento. O algoritmo implementado é, na prática, uma heurística que guia a busca pela melhor configuração de acessos. Ele atribui uma pontuação de confiança para cada permissão de um usuário, baseando-se em múltiplos fatores ponderados, conforme sugerido por análises comparativas de sistemas de IAM baseados em IA (Wairagade; Ranjan, 2025):

- **Análise de Pares (*Peer Analysis*):** Compara as permissões do usuário com as de outros membros da mesma *squad* ou com a mesma função (RBAC). Uma permissão compartilhada pela maioria dos pares recebe uma pontuação de confiança maior. Essa abordagem é uma simplificação de técnicas mais avançadas de detecção de anomalias baseadas em grafos, que modelam as relações entre usuários e recursos para identificar padrões de acesso incomuns que seriam invisíveis em uma análise individual, especialmente com o uso de Deep Learning (Ma et al., 2021).
- **Análise Hierárquica:** Verifica se as permissões são consistentes com as do gestor ou com as de usuários de mesma área e gestão.
- **Análise de Frequência:** Avalia a raridade de uma permissão dentro da organização. Permissões de alto risco são removidas automaticamente sem a necessidade de revisão.

A recomendação de revogação só era validada quando a confiança do modelo era igual ou superior a 80%. Este limiar foi definido para otimizar o equilíbrio entre precisão (evitar a revogação de acessos legítimos, ou falsos positivos) e recall

(identificar o máximo de acessos desnecessários). Conforme estudos como o de Glöckler et al. (2024), a gestão de requisitos em IAM é um desafio complexo, e abordagens sistemáticas são essenciais para o sucesso (Glöckler et al., 2024). A solução automatizada atua como um sistema de recomendação para o gestor, que ainda realiza a validação final, mas agora munido de uma análise de dados robusta. Essa abordagem está alinhada aos princípios da IA Explicável (XAI), que defende que os modelos de decisão automatizada devem ser transparentes e compreensíveis, um fator crucial para a adoção de tecnologias de IA em áreas críticas como a cibersegurança (Rjoub et al., 2023)

REFERÊNCIAS

FARMAN, Mahr. **Implementing the Policy of Least Privilege: Enhancing Security through Segregation of Duties**. 2024. Preprint. DOI: 10.13140/RG.2.2.32556.88964. Disponível em: https://www.researchgate.net/publication/384356835_Implementing_the_Policy_of_Least_Privilege_Enhancing_Security_through_Segregation_of_Duties. Acesso em: 4 set. 2025.

GLÖCKLER, J.; SEDLMEIR, J.; FRANK, M.; FRIDGEN, G. **A Systematic Review of Identity and Access Management Requirements in Enterprises and Potential Contributions of Self-Sovereign Identity. & Information Systems Engineering**. Disponível em: <https://link.springer.com/article/10.1007/s12599-023-00830-x>. Acesso em: 4 set. 2025.

HUSSAIN, T.; AL-DHURAIB, B.; ZAINAL, A.; IDRIS, M. Y. I.; RASID, S. Z. A.; HASHIM, F. **A survey on Zero Trust Architecture: challenges and future trends**. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1155/2022/6476274>. Acesso em: 16 out. 2025.

MA, X.; WU, J.; XUE, S.; YANG, J.; ZHOU, C.; SHENG, Q. Z.; XIONG, H.; AKOGLU, L. **A Comprehensive Survey on Graph Anomaly Detection with Deep Learning. ACM Computing Surveys**. Disponível em: https://www.researchgate.net/publication/355309912_A_Comprehensive_Survey_on_Graph_Anomaly_Detection_with_Deep_Learning. Acesso em: 17 out. 2025.

NOBI, M.; GUPTA, M.; PRAHARAJ, L.; ABDELSALAM, M.; KRISHNAN, R.; SANDHU, R. **Machine Learning in Access Control: A Taxonomy and Survey**. Disponível em: <https://arxiv.org/abs/2207.01739>. Acesso em: 15 set. 2025.

RJOUB, G.; BENTAHAR, J.; WAHAB, O.; MIZOUNI, R.; SONG, A.; COHEN, R.; OTROK, H. MOURAD, A. **A Survey on Explainable Artificial Intelligence for Cybersecurity**. Disponível em: <https://arxiv.org/html/2303.12942>. Acesso em: 18 out. 2025.

SINGH, C.; WARRAICH, J.; THAKKAR, R. **IAM Identity Access Management: importance in maintaining security systems within organizations**. Disponível em: https://www.researchgate.net/publication/374034268_IAM_Identity_Access_Management-Importance_in_Maintaining_Security_Systems_within_Organizations. Acesso em: 5 set. 2025.

WAIRAGADE, A.; RANJAN, S. **AI in Identity and Access Management (IAM) for Enterprise Systems: A Comparative Analysis**. Disponível em: <https://www.sciencedirect.com/science/article/pii/S187705092502071X>. Acesso em: 30 set. 2025.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

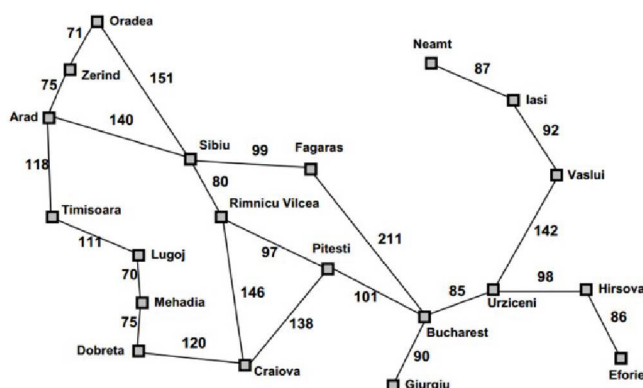
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



Arad	366	Mehadia	241
Bucarest	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 Bucarest.

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

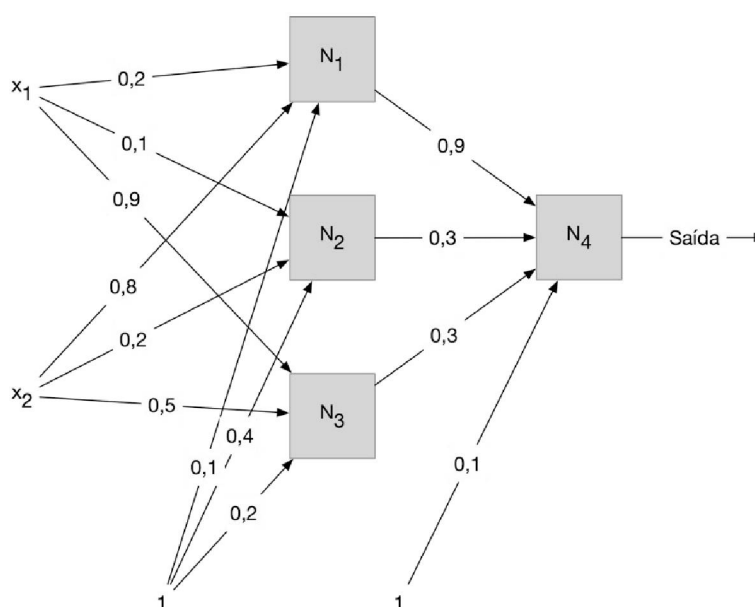
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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:

- (6,25 pontos)** Valor de saída do neurônio N_1
- (6,25 pontos)** Valor de saída do neurônio N_2
- (6,25 pontos)** Valor de saída do neurônio N_3
- (6,25 pontos)** Valor de saída da rede como um todo

B – RESOLUÇÃO

- 1 a) Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no

desenvolvimento de sistemas e algoritmos capazes de realizar tarefas que normalmente exigiriam inteligência humana. O objetivo da inteligência artificial é criar sistemas que possam aprender, raciocinar, tomar decisões e resolver problemas de forma autônoma, imitando ou mesmo superando as capacidades humanas em determinadas áreas.

b) A abordagem que melhor descreve a situação apresentada é a "Agir Racionalmente" dado que a descrição fornecida aborda vários aspectos das 4 abordagens aprendidas em aula tais como: agir de forma autônoma, imitar ou permitir que o agente ultrapasse a capacidade humana permitindo que o mesmo passe pelo teste de turing, representação de raciocínio humano permitindo tomar boas decisões. Observando todos os aspectos, percebemos que apesar de se encaixar em outras abordagens só podemos concluir que a abordagem que melhor se encaixa é a "Agir Racionalmente" justamente por suas vastas possibilidades que agregam as principais características das 4 abordagens.

c) O ChatGPT (sigla para "Generative Pre-Trained Transformer") é um modelo de linguagem baseado em deep learning (aprendizagem profunda), um braço da inteligência artificial.

O chatGPT é um software que utiliza o modelo GPT para gerar texto e que são treinados para compreender linguagem natural. Eles emitem uma saída em resposta a uma entrada do usuário, que é chamada de "Prompt", além de utilizarem processos como Tokenização e embedding.

Na prática, a plataforma utiliza um algoritmo baseado em redes neurais que permitem estabelecer uma conversa com o usuário a partir do processamento de um imenso volume de dados.

Funciona a partir de uma base de conhecimento atualizada que permite decodificar palavras para oferecer respostas textuais às pessoas.

Trata-se de um modelo treinado a partir de dados de textos disponíveis na internet.

Referências:

<https://platform.openai.com/docs>

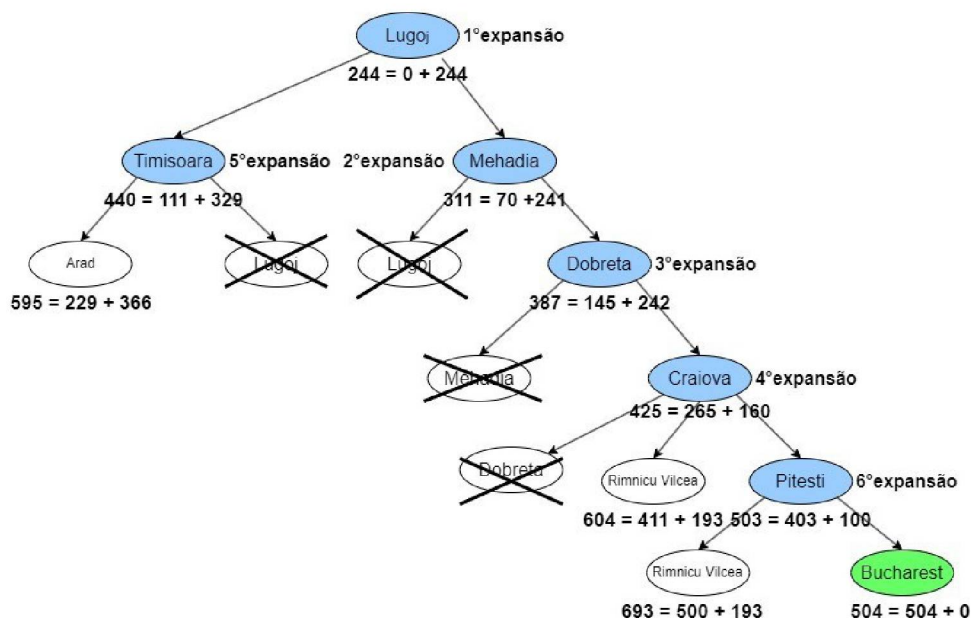
<https://platform.openai.com/docs/introduction>

<https://platform.openai.com/docs/guides/fine-tuning>

<https://fia.com.br/blog/chat-gpt/>

d) O ChatGPT se aproxima mais da abordagem "Agir como Humano" pois seu funcionamento tenta simular o comportamento humano das seguintes formas, ele responde suas perguntas com um determinado "conhecimento", ele processa a linguagem natural podendo até mesmo se adaptar a sua forma de falar e até mesmo responder em outros idiomas, possui um raciocínio que apesar de ser diferente do humano é um tipo de raciocínio e por final ele ainda consegue aprender com cada Prompt e avaliação do usuário em relação as suas respostas.

2 a)



APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

- Carregue a base de dados media_precos_carros_brasil.csv
- Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes
- Verifique se há dados duplicados nos dados
- Crie duas categorias, para separar colunas numéricas e categóricas. Imprima o resumo de informações das variáveis numéricas e categóricas (estatística descritiva dos dados)
- Imprima a contagem de valores por modelo (model) e marca do carro (brand)
- Dê um breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados

2 Visualização dos dados

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

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

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

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

- Escolha as variáveis **numéricas** (modelos de Regressão) para serem as variáveis independentes do modelo. A variável target é **avg_price**. **Observação:** caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no modelo. Indique **quais variáveis** foram transformadas e **como** foram transformadas
- Crie partições contendo 75% dos dados para treino e 25% para teste
- Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros. **Observação:** caso julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo
- Grave os valores preditos em variáveis criadas

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

B - RESOLUÇÃO

1 Análise Exploratória dos dados

```
#1 a)

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

# b)

dados.isna().any()

#resultado:
#year_of_reference      True
#month_of_reference     True
#fipecode               True
#authentication         True
#brand                  True
#model                  True
#fuel                   True
#gear                   True
#engine_size            True
#year_model             True
#avg_price_brl          True
#dtype: bool
```



```
# Exclusão das colunas com valor faltante
```

```
dados = dados.dropna()
```

```
# Verificando a alteração
```

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

```
#resultado
```

```
#year_of_reference      False
```

```
#month_of_reference     False
```

```
#fipecode               False
```

```
#authentication        False
```

```
#brand                  False
```

```
#model                  False
```

```
#fuel                   False
```

```
#gear                   False
```

```
#engine_size            False
```

```
#year_model             False
```

```
#avg_price_brl          False
```

```
#dtype: bool
```

```
#c) Valores Duplicados - Verificando
```

```
#resultado
```

```
#2
```

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

```
# Removendo valores duplicados
```

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

```
#d) Categorias - categorias numéricas e categóricas
```

```
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']
# Variáveis numéricas
dados[numericas_cols].describe()
#Resultado

```

Tabela 1 – Resultado das Variáveis Numéricas

	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

Tabela 2 – Resultado das variáveis categóricas pt1

	month_of_reference	fipe_code	authentication	brand
count	202295	202295	202295	202295
unique	12	2091	202295	6
top	January	003281-6	cfzlcztzfwrcp	Fiat
freq	24260	425	1	44962

Tabela 3 – Resultado das variáveis categóricas pt2

	model	fuel	gear	engine_size
count	202295	202295	202295	202295
unique	2112	3	2	29
top	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6
freq	425	168684	161883	47420

```

#e)

```

```

# Contagem de valores - categoria de 'Model'

```

```

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

```

```

# Contagem de valores = categoria 'Brand'

```

```

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

```

```

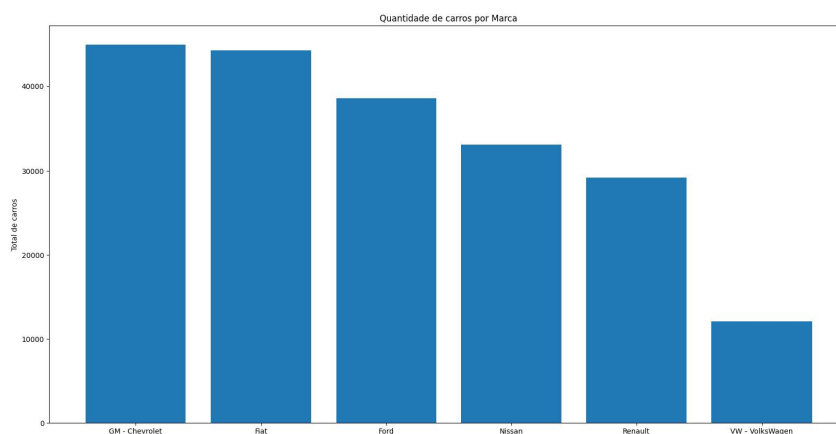
#RESPOSTA: Na análise de modelo, o Focus 1.6 e o Palio week são os

```

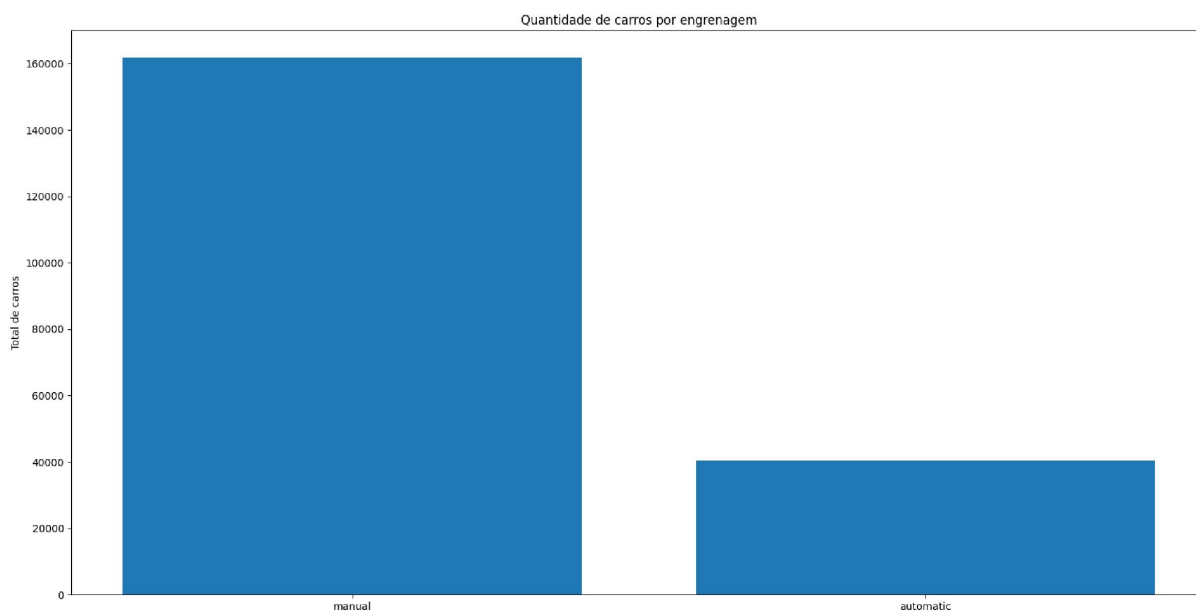
*#carros que foram mais comprados de acordo com a base de dados. Apesar
→ de juntos o Focus 1.6 e Focus 2.0 serem mais vendidos que o Palio a
→ marca mais vendida é a Fiat, seguida da Volkswagen da GM sendo a
→ terceira mais vendida. Os carros que mais aparecem são os carros a
→ Gasolina e o Manual. A média de preço de todos os carros
→ apresentados na base é de R\$52.756,91.*

2 Visualização dos dados

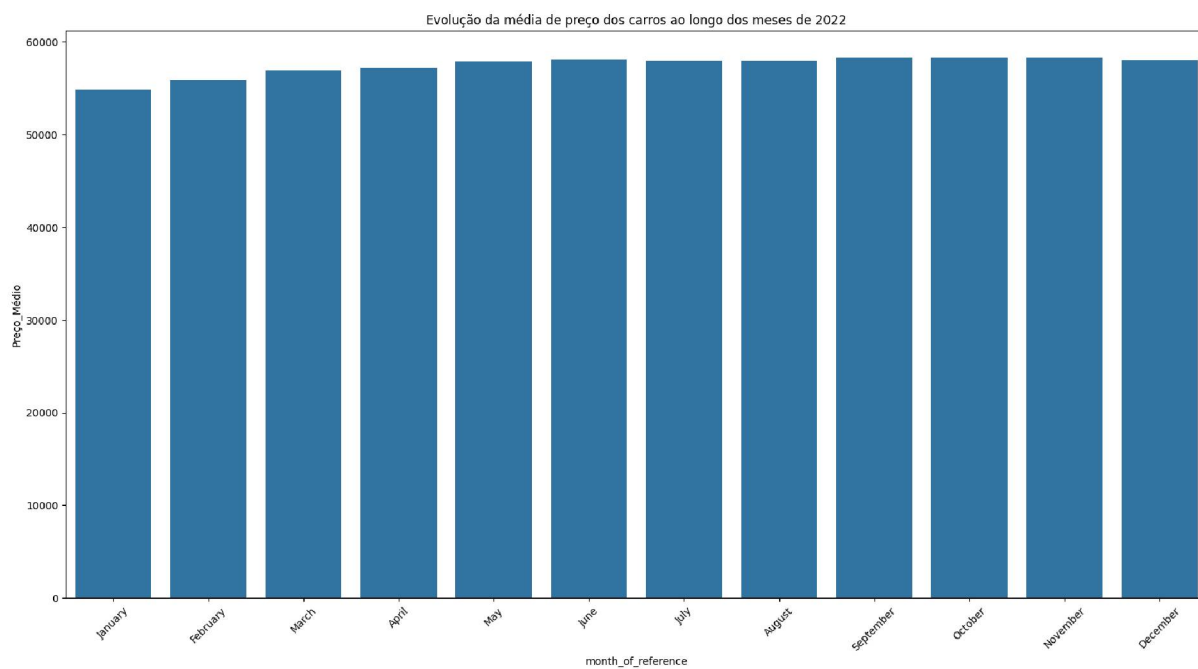
a)



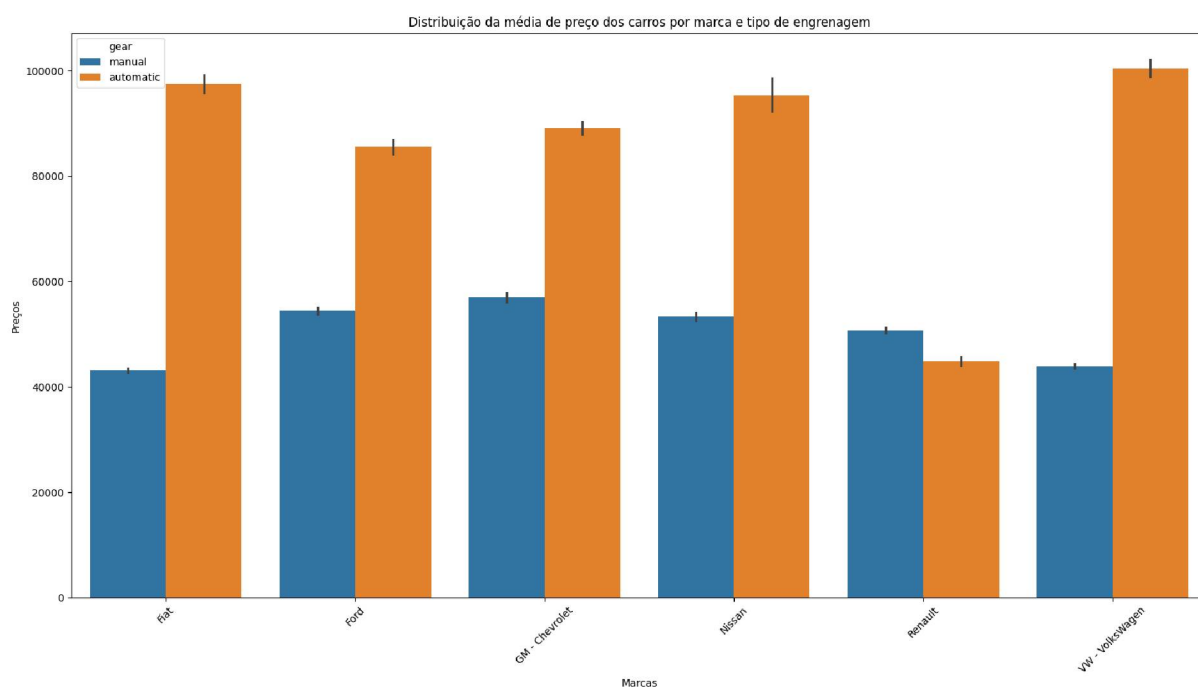
b)



c)



d)



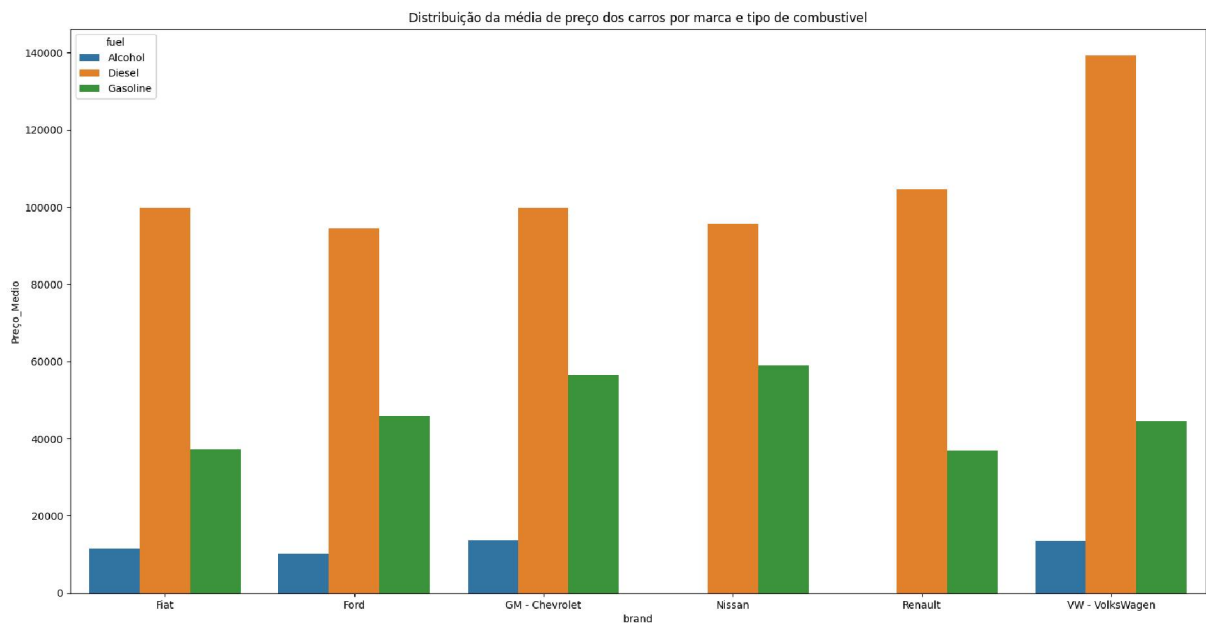
e)

- Os preços dos carros com engrenagem automática são bem superiores aos de engrenagem manual.
- A marca Renault é a única que os carros com engrenagens automáticos tem valores

inferiores ao de engrenagem manual, sendo também a marca com valores menores em comparação com as outras marcas.

- Os carros da marca Volkswagen tem os valores médios mais caros.

f)



g)

- Os carros a álcool tem o preço médio mais baixo, em todas as marcas, exceto a marca Nissan e Renault que não tem carros a álcool.

- O preço médio de todas as marcas tem valores superiores nos carros com combustível a Diesel.

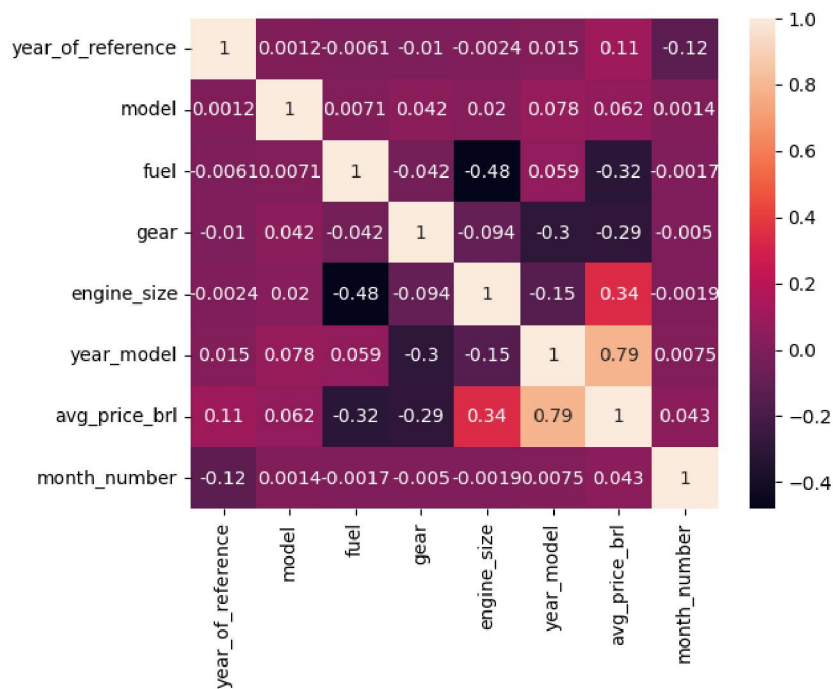
- Nos carros à diesel o maior preço médio foi localizado na marca Volkswagen.

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

a)

R: Foram escolhidas year of reference, model, fuel, gear, engine size, year model e month number para avaliação, as variáveis model, fuel, engine size, gear e year model foram transformadas utilizando LabelEncoder

Mapa de Correlação das Variáveis Numéricas



*# Variável X contém apenas variáveis numéricas de interesse para a
 ↳ análise, excluindo a variável target*

```
X = data_nums.drop(['avg_price_brl'], axis = 1)
```

Variável Y contém apenas a variável target - Faixa Salarial

```
Y = data_nums['avg_price_brl']
```

```
Y.head()
```

```
#0      9162.0
```

```
#1      8832.0
```

```
#2      8388.0
```

```
#3      8453.0
```

```
#4     12525.0
```

```
#Name: avg_price_brl, dtype: float64
```

#b)

Divisão: 25% dos dados são de teste e 75% de treinamento

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
```

```
↳ 0.25, random_state = 42)
```

Observando os dados de treinamento

```
print(X_train.shape)
X_train.head(1)
#(151721, 7)
```

	year_of_reference	model	fuel	gear
156364	2022	364	1	1

engine_size	year_model	month_number
10	20	8

	year_of_reference	model	fuel	gear
180633	2022	1235	2	1

engine_size	year_model	month_number
5	15	11

```
#c)
model_random_forest = RandomForestRegressor()
model_random_forest_with_params = RandomForestRegressor(max_depth=29,
↳ min_samples_leaf=32, min_samples_split=28, n_estimators=208,
↳ random_state=43)
model_xgboost = XGBRegressor()
model_random_forest.fit(X_train, Y_train)
model_random_forest_with_params.fit(X_train, Y_train)
#Resultado
#RandomForestRegressor
#RandomForestRegressor(max_depth=29, min_samples_leaf=32,
↳ min_samples_split=28,
n_estimators=208, random_state=43)
```



```
model_xgboost.fit(X_train, Y_train)
```

```
#Resultado
```

```
#XGBRegressor
```

```
#XGBRegressor(base_score=None, booster=None,
```

```
→ callbacks=None, colsample_bylevel=None, colsample_bynode=None,
```

```
→ colsample_bytree=None, device=None, early_stopping_rounds=None,
```

```
→ enable_categorical=False, eval_metric=None, feature_types=None,
```

```
→ gamma=None, grow_policy=None, importance_type=None,
```

```
→ interaction_constraints=None, learning_rate=None, max_bin=None,
```

```
→ max_cat_threshold=None, max_cat_to_onehot=None,
```

```
→ max_delta_step=None, max_depth=None, max_leaves=None,
```

```
→ min_child_weight=None, missing=nan, monotone_constraints=None,
```

```
→ multi_strategy=None, n_estimators=None, n_jobs=None,
```

```
→ num_parallel_tree=None, random_state=None, ...)
```

```
#R: Foram treinados três modelos, RandomFlorest (sem parametros), Random
```

```
→ Florest com utilização de parametros (max depth, min_samples_leaf,
```

```
→ min_samples_split, n_estimators, random_state) e XGBOOST
```

d)

```
predicted_values_random_florest = model_random_florest.predict(X_test)
```

```
predicted_values_random_florest
```

```
#Resultado
```

```
#array([ 42417.29,  11120.21,   9043.58, ..., 105895.85,   9401.27,  
        24422.68])
```

```
predicted_values_random_florest_with_params =
```

```
→ model_random_florest_with_params.predict(X_test)
```

```
predicted_values_random_florest_with_params
```

```
#Resultado
```

```

#array([ 45209.77079159,  11414.42517017,  14984.46717642, ...,
        103988.1374703 ,  14943.40484225,  24046.74051621])

predicted_values_xgboost = model_xgboost.predict(X_test)
predicted_values_xgboost
#Resultado
#array([ 45716.93 ,  11505.324,   9243.854, ..., 108816.414,  10348.15
↪ ,
        23879.635], dtype=float32)

#e)
model_random_florest.feature_importances_
feature_importances_random_florest =
↪ pd.DataFrame(model_random_florest.feature_importances_, index =
↪ X_train.columns, columns=['importance']).sort_values('importance',
↪ ascending = False)
feature_importances_random_florest

```

Variável	Importância
engine_size	0.453158
year_model	0.391178
model	0.070232
gear	0.034472
fuel	0.032931
year_of_reference	0.012318
month_number	0.005711

```

model_random_florest_with_params.feature_importances_feature_
importances_random_florest_with_params = pd.DataFrame(
    model_random_florest_with_params.feature_importances_,
    index=X_train.columns,

```

```
columns=['importance']
).sort_values('importance', ascending = False)
```

```
feature_importances_random_forest_with_params
```

Variável	Importância
engine_size	0.469577
year_model	0.411965
model	0.047686
gear	0.022826
fuel	0.035146
year_of_reference	0.010479
month_number	0.002321

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

```
feature_importances_xgboost
```

Variável	Importância
engine_size	0.428532
year_model	0.221430
model	0.030906
gear	0.129561
fuel	0.165807
year_of_reference	0.017530
month_number	0.006233

f)

R: Em todos os modelos engine_size e year_model tiveram maiores valores de importância, tendo seus maiores valores no modelo random forest with params. As demais variáveis não obtiveram um valor alto de importância.

g)

R: Melhor modelo de avaliação escolhido foi o Random Florest, sem utilização de nenhum parametro.

h)

R: Apesar de a utilização de parametros no modelo RandomForest com aumento ligeiramente a importancia das variaveis. O modelo Random Florest (sem parametros), gerou melhores métricas de mse e mae contendo assim menos erros. E teve a maior acurácia perante os outros modelos.

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

$$\text{Volume} = b_0 + b_1 * \text{dap}^2 * H_t$$

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

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

Tarefas

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

- O modelo alométrico é dado por: $\text{Volume} = b_0 + b_1 * \text{dap}^2 * H_t$

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

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

- Coeficiente de determinação: R^2

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- $S_{yx}\%$

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

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

7. Escolha o melhor modelo.

B - RESOLUÇÃO

1 Pesquisa com Dados de Satélite (Satellite)

1.1. Carregue a base de dados Satellite

```
data(Satellite)
summary(Satellite)
```

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

```
set.seed(7)
indices <- createDataPartition(Satellite$classes, p=0.80, list=FALSE)
treino <- Satellite[indices, ]
teste <- Satellite[-indices, ]
```

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

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

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

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

```
#Random Forest
confusionMatrix(predict.rf, teste$classes)
```

Overall Statistics

Accuracy : 0.9213

Kappa : 0.9025

```
# SVM
```

```
confusionMatrix(predict.svm, teste$classes)
```

Overall Statistics

Accuracy : 0.9112

Kappa : 0.8901

```
#RNA
```

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

Overall Statistics

Accuracy : 0.5802

Kappa : 0.4692

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

```
# O melhor foi o RandomForest que obteve uma acurácia de 0.9213.
```

2 Estimativa de Volumes de Árvores

2.1 Carregar o arquivo Volumes.csv

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

2.2 Eliminar a coluna NR

```
df$NR <- NULL
```

2.3 Criar partição de dados

```
set.seed(7)
```

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

2.4 Treinar os modelos

```
rf2 <- caret::train(VOL~., data=treino, method="rf")
svm2 <- caret::train(VOL~., data=treino, method="svmRadial")
rna2 <- caret::train(VOL~., data=treino, method="nnet", trace=FALSE)
```

2.5 Efetuar as predições

```
predict.rf2 <- predict(rf2, teste)
predict.svm2 <- predict(svm2, teste)
predict.rna2 <- predict(rna2, teste)
predict.alom <- predict(alom, teste)
```

2.6 Calcular as métricas

```
# Coeficiente de determinação
coeficienteDeDeterminacao <- function(y_real, y_pred) {
  x <- sum((y_real-y_pred)^2)
  z <- sum((y_real-mean(y_real))^2)
  return (1-(x/z))
}

coeficienteDeDeterminacao(teste$VOL, predict.rf2) # 0.8535647
coeficienteDeDeterminacao(teste$VOL, predict.svm2) # 0.8484652
coeficienteDeDeterminacao(teste$VOL, predict.rna2) # - 0.7244946
coeficienteDeDeterminacao(teste$VOL, predict.alom) # 0.8263134

# Erro padrão da estimativa:
erroPadraoDaEstimativa <- function(y_real, y_pred) {
```

```

x <- sum((y_real-y_pred)^2)
z <- length(y_real)-2
return (sqrt(x/z))
}

```

```

percentErroPadraoDaEstimativa <- function(y_real, y_pred) {
  return (((erroPadraoDaEstimativa(y_real, y_pred)/mean(y_real))*100))
}

```

```

erroPadraoDaEstimativa(teste$VOL, predict.rf2) # 0.1445527
percentErroPadraoDaEstimativa(teste$VOL, predict.rf2) # 11.07658
erroPadraoDaEstimativa(teste$VOL, predict.svm2) # 0.1470481
percentErroPadraoDaEstimativa(teste$VOL, predict.svm2) # 11.2678
erroPadraoDaEstimativa(teste$VOL, predict.rna2) # 0.49606
percentErroPadraoDaEstimativa(teste$VOL, predict.rna2) # 38.01139
erroPadraoDaEstimativa(teste$VOL, predict.alom) # 0.1574296
percentErroPadraoDaEstimativa(teste$VOL, predict.alom) # 12.0633

```

2.7 Escolha o melhor modelo

O melhor modelo é o Random Forest em todas as funções, de acordo com o critério de melhor modelo.

Tabela 4 – Comparação dos resultados dos modelos de regressão.

Métrica	RF	SVM	RNA	ALOM	(Melhor Modelo)
Coeficiente de determinação	0.8535	0.8484	-0.7244	0.8263	Próximo de 1
Erro padrão da estimativa	0.1445	0.1470	0.4960	0.1574	Próximo de 0
Porcentagem de erro	11.07	11.26	38.01	12.06	Próximo de 0

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

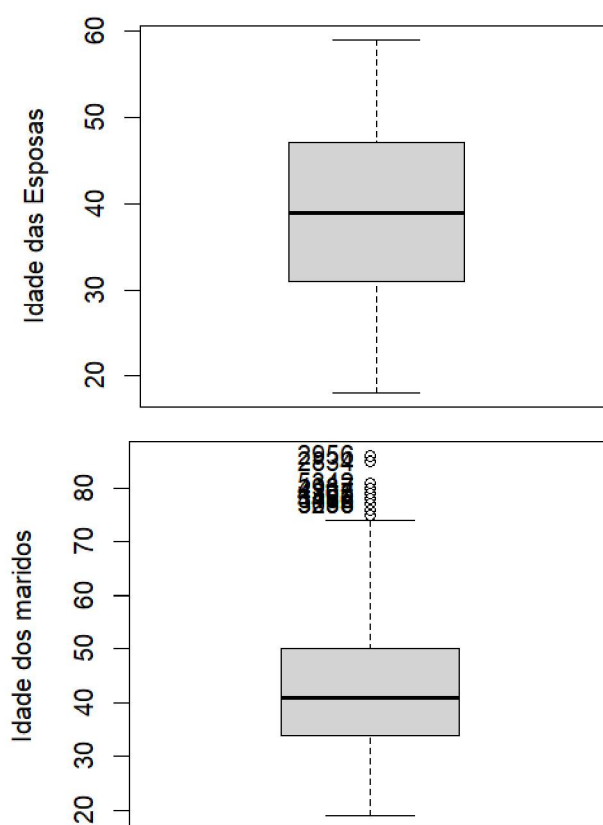
Obs:

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

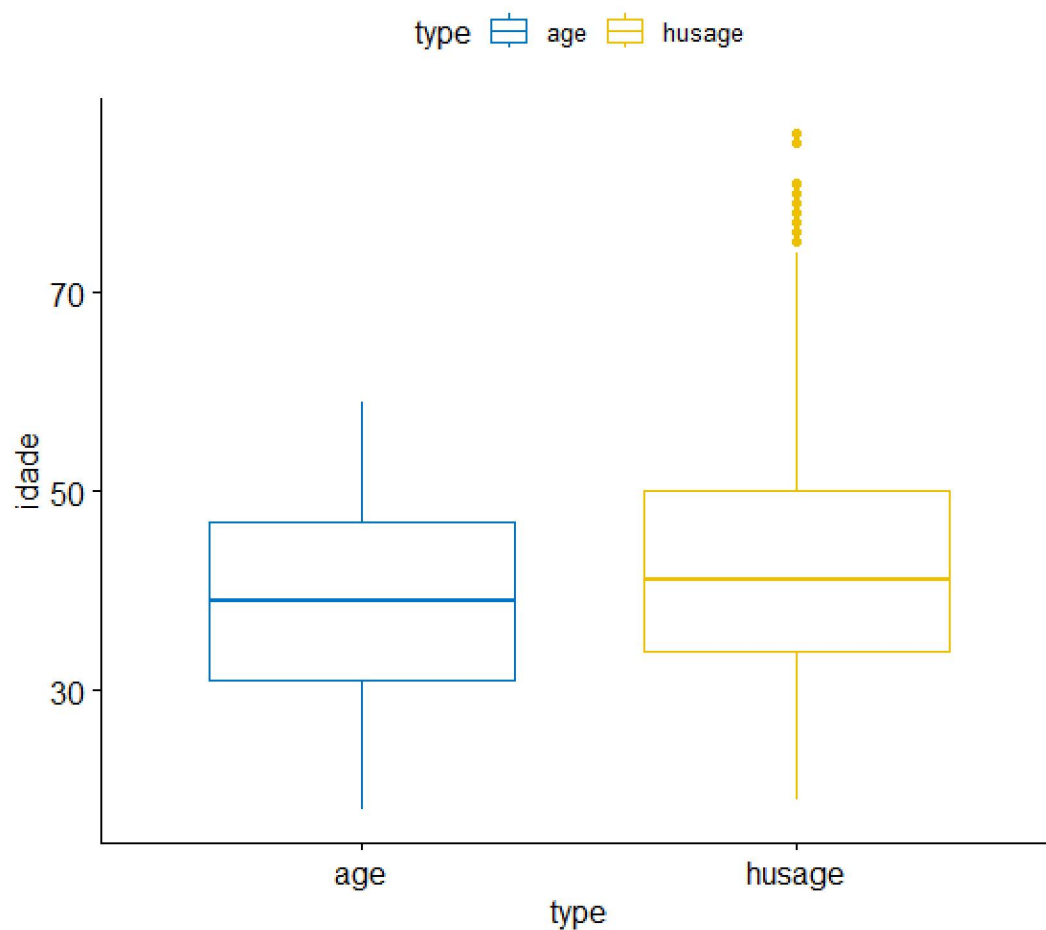
Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

B - RESOLUÇÃO**1. Gráficos e tabelas**

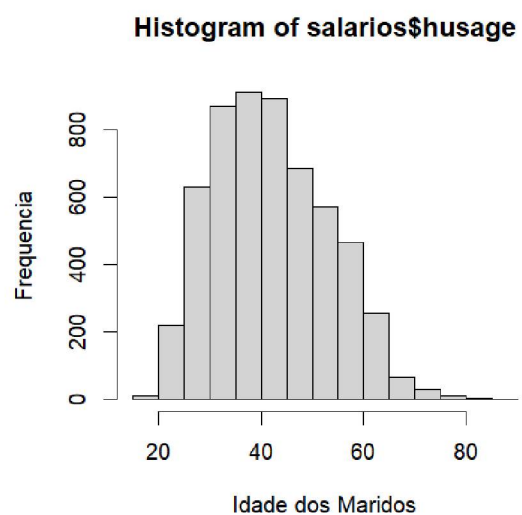
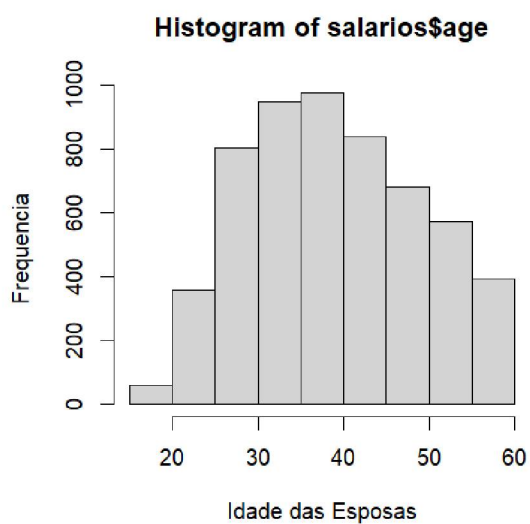
a) (15 pontos) Elaborar os gráficos box-plot e histograma das variáveis “age” (idade da esposa) e “husage” (idade do marido) e comparar os resultados.

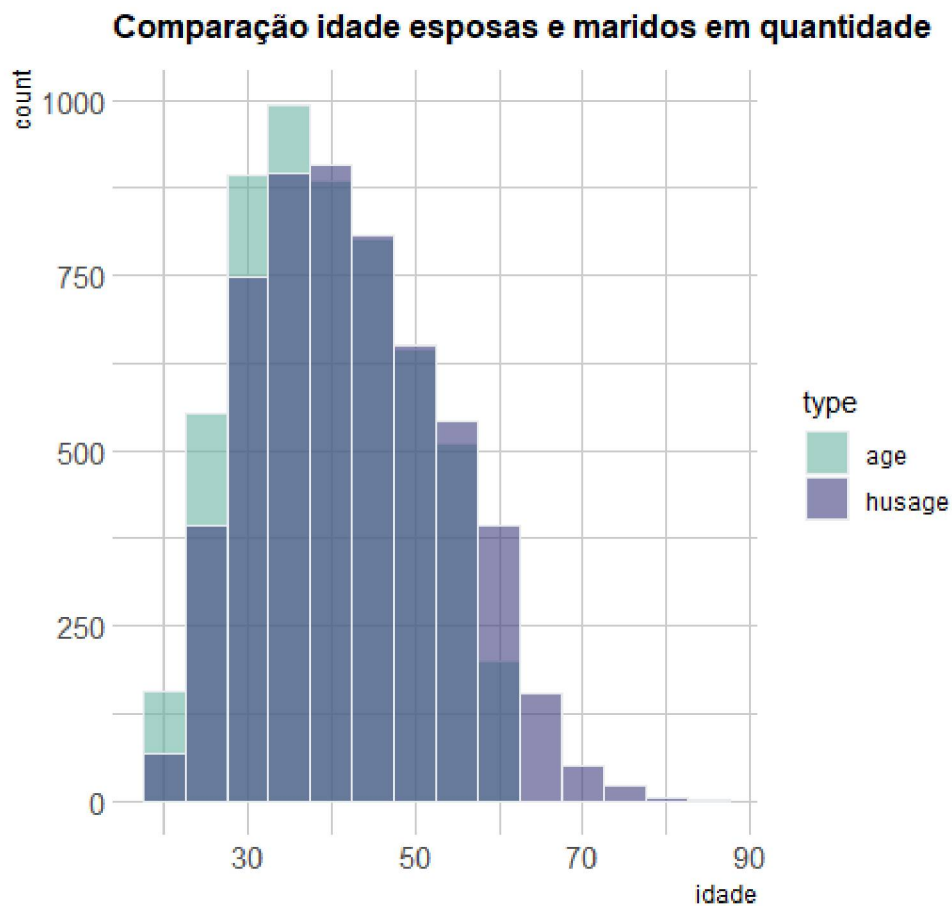


Comparação idade esposas e maridos



R: Podemos observar que a mediana da idade dos maridos é levemente superior à das esposas, os maridos possuem alguns outliers enquanto as esposas não possuem e a idade dos maridos ultrapassa os 70 enquanto a das esposas não.





R: Podemos observar pelo histograma que em idades até 40 anos as esposas superam as quantidades de maridos dos 40 anos e idades superiores a quantidade de maridos supera as esposas.

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

Idade das esposas

Class limits	f	rf	rf(%)	cf	cf(%)
[17.82,20.804)	61	0.01	1.08	61	1.08
[20.804,23.787)	161	0.03	2.86	222	3.94
[23.787,26.771)	312	0.06	5.54	534	9.48
[26.771,29.754)	505	0.09	8.96	1039	18.44
[29.754,32.738)	562	0.10	9.98	1601	28.42
[32.738,35.721)	571	0.10	10.13	2172	38.55
[35.721,38.705)	624	0.11	11.08	2796	49.63

[38.705,41.689)	510	0.09	9.05	3306	58.68
[41.689,44.672)	542	0.10	9.62	3848	68.30
[44.672,47.656)	432	0.08	7.67	4280	75.97
[47.656,50.639)	389	0.07	6.90	4669	82.87
[50.639,53.623)	358	0.06	6.35	5027	89.23
[53.623,56.606)	304	0.05	5.40	5331	94.62
[56.606,59.59)	303	0.05	5.38	5634	100.00

Idade dos maridos

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

Podemos observar que a maior frequência das esposas está entre 35.721 e 38.705, enquanto a maior frequência dos maridos está entre 38.253 e 43.114.

2 Medidas de posição e dispersão

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

R:

Idade maridos:

Média de idade dos maridos: 42.45296

Mediana da idade dos maridos: 41

Moda da idade dos maridos: 44 (com 201 repetições)

Idade esposas:

Média de idade das esposas: 39.42758

Mediana da idade das esposas: 39

Moda de idade das esposas: 37 (com 217 repetições)

A média de idade dos maridos é de 42 anos enquanto a das esposas é de 39 anos, sendo a média de idade dos maridos 7% superior a da idade das esposas.

A mediana da idade dos maridos é de 41 anos e das esposas é de 39 anos sendo a mediana de idade dos maridos 5% superior a idade das esposas.

A moda de idade dos maridos é de 44 anos enquanto a moda de idade das esposas é de 37 anos sendo a moda dos maridos 18% superior.

b) (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

R:

Idades maridos:

Variância: 126.0717

Desvio Padrão: 11.22817

Coef. Variação: 26.44849

Idade esposas:

Variância: 99.75234

Desvio Padrão: 9.98761

Coef. Variação: 25.33153

A variância da idade dos maridos é 26.3847% superior do que a das esposas.

O desvio padrão de idade dos maridos é 12.42099% superior ao desvio padrão da idade das esposas.

O coeficiente de variação dos maridos é 4,40% superior ao coeficiente de variação das esposas. Tanto o coeficiente de variação dos maridos (26.44849) quanto o das esposas (25.33153) possuem uma média dispersão ($15\% \leq CV \leq 30\%$)

3 Testes paramétricos ou não paramétricos

a) (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:

1) 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.

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

R: O teste escolhido foi o de U de Mann-Whitney (não paramétrico), pois ao realizar os testes de normalidade nos dados, foi observado que os dados não possuem distribuição normal.

Os testes realizados foram Shapiro-Wilk utilizando as primeiras 5000 amostras nos dados de idades de maridos e de esposas, JarqueBeraTest utilizando parametro robust=TRUE e Anderson-Darling Test (ad.tes)

Shapiro-wilk normality test

data: salarios\$husage[0: 5000]

W = 0.98151, p-value < 0. 000000000000000022

```
> with(salarios, shapiro.test(salarios$age[0:5000])) # p-value <
↪ 0.000000000000000022
```

Shapiro-wilk normality test

data: salarios\$age[0:5000]

W = 0.97695, p-value < 0. 000000000000000022

```
> JarqueBeraTest(salarios$husage, robust = TRUE) # p-value <
↪ 0.000000000000000022
```

Robust Jarque Bera Test

data: salarios\$husage

X-squared = 153.12, df = 2, p-value < 0.000000000000000022

```
> JarqueBeraTest(salarios$age, robust = TRUE) # p-value <
↪ 0.000000000000000022
```

Robust Jarque Bera Test

```
data: salarios$age
X-squared = 158.49, df = 2, p-value < 0.00000000000000022

> ad.test(salarios$husage) # p-value < 0.00000000000000022
```

Anderson-Darling normality test

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

> ad.test(salarios$age) # p-value < 0.00000000000000022
```

Anderson-Darling normality test

```
data: salarios$age
A = 31.828, p-value < 0.00000000000000022
```

Intervalo de confiança resultados:

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

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

wilcoxon rank sum test with continuity correction

```
data: idade by type
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

O p-value do teste é 0.00000000000000022, que é menor que o nível de significância 0,05. Podemos concluir que a idade mediana das esposas é estatisticamente diferente da idade mediana dos maridos (rejeitamos H_0).

O intervalo de confiança da diferença entre as medianas está entre -3.000024 e -2.000033, com uma mediana de -2.999966.

H_0 : A idade mediana das esposas não é estatisticamente maior que a idade mediana dos maridos

H_a : A idade mediana das esposas é estatisticamente maior do que a idade mediana dos maridos.

wilcoxon rank sum test with continuity correction

data: idade by type

W = 13619912, p-value = 1

alternative hypothesis: true location shift is greater than 0

95 percent confidence interval:

-3.000034

Inf

sample estimates:

difference **in** location

-2.999966

Como p-value > 0.05, aceitamos H_0 , a idade mediana das esposas não é estatisticamente maior do que a idade mediana dos maridos.

H_0 : A idade mediana das esposas não é estatisticamente menor do que a idade mediana dos maridos

H_a : A idade mediana das esposas é estatisticamente menor do que a idade mediana dos maridos

```
wilcox.test(idade ~ type, data = salarios2,
exact = FALSE, alternative = "less", conf.int = TRUE)
```

wilcoxon rank sum test with continuity correction

data: idade by type

W = 13619912, p-value < 0.00000000000000022

alternative hypothesis: true location shift is less than 0

95 percent confidence interval:

-Inf -2.000046

sample estimates :

difference in location

-2.999966

Como $p\text{-value} < 0.05$, rejeitamos H_0 , sendo, portanto, H_a válida: A idade mediana das esposas é estatisticamente menor do que a idade mediana dos maridos.

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B - RESOLUÇÃO

Métricas resultados:

	RMSE	Rsquare
Ridge base de treinos	0.425665	0.2921375
Ridge base de testes	0.5006593	0.259081
Lasso base de treinos	0.426272	0.2901174
Lasso base de testes	0.5009585	0.2581953
Elastic base de treinos	0.4259722	0.2911154
Elastic base de testes	0.500484	0.2595998

Predição e intervalos de confiança:

	Valor da predição	Intervalo de Confiança negativo	Intervalo de Confiança Positivo
Regressão Ridge	3.32512	3.30257	3.34767
Regressão Lasso	3.286964	3.264414	3.309514
Regressão Elastic	3.278598	3.256048	3.301147

Conclusão: Os três modelos de regressão possuem métricas bem similares, não existindo assim um modelo que seja superior aos demais.

```
> library(dplyr)
> library(caret)
> library(glmnet)
>
> load("trabalhosalarios.RData")
>
> dat <- trabalhosalarios
>
> View(dat)
> glimpse(dat)
Rows: 2,574
Columns: 17
...
> gc()

              used   (Mb) gc trigger   (Mb) max used   (Mb)
Ncells 14078688 751.9   24930525 1331.5 24930525 1331.5
Vcells 52489476 400.5   94570028  721.6 94570028  721.6
>
>
> set.seed(302)
>
> index = sample(1:nrow(dat),0.8*nrow(dat))
>
> train = dat[index,]
> test = dat[-index,]
>
> dim(train)
[1] 2059  17
> dim(test)
[1] 515  17
>
> # Objeto com as variaveis para padronizar
```



```

> cols = c('husage', 'husearns', 'huseduc', 'hushrs',
+          'age', 'educ', 'exper')
>
> pre_proc_val <- preProcess(train[,cols],
+                             method = c("center", "scale"))
>
> train[,cols] = predict(pre_proc_val, train[,cols])
> test[,cols] = predict(pre_proc_val, test[,cols])
>
> summary(train)

```

husage		husunion		husearns		huseduc	
↪ husblk		hushisp		hushrs		kidge6	
Min.	:-2.1039	Min.	:0.0000	Min.	:-1.7145	Min.	:-5.0191
↪ Min.	:0.00000	Min.	:0.00000	Min.	:-3.3291	Min.	:0.0000
1st Qu.:	-0.8087	1st Qu.:	0.0000	1st Qu.:	-0.6615	1st Qu.:	-0.5568
↪ 1st Qu.:	0.00000	1st Qu.:	0.00000	1st Qu.:	-0.1961	1st Qu.:	0.0000
Median :	-0.1113	Median :	0.0000	Median :	-0.2166	Median :	-0.1849
↪ Median :	0.00000	Median :	0.00000	Median :	-0.1961	Median :	0.0000
Mean :	0.0000	Mean :	0.2205	Mean :	0.0000	Mean :	0.0000
↪ Mean :	0.06217	Mean :	0.04905	Mean :	0.0000	Mean :	0.3439
3rd Qu.:	0.6857	3rd Qu.:	0.0000	3rd Qu.:	0.4212	3rd Qu.:	0.9306
↪ 3rd Qu.:	0.00000	3rd Qu.:	0.00000	3rd Qu.:	0.5871	3rd Qu.:	1.0000
Max. :	2.8775	Max. :	1.0000	Max. :	3.9006	Max. :	1.6744
↪ Max. :	1.00000	Max. :	1.00000	Max. :	4.4250	Max. :	1.0000

earns		age		black		educ	
↪ hispanic		union		exper		kidlt6	
Min. :	5.0	Min. :	-2.12418	Min. :	0.00000	Min. :	-5.6264
↪ Min. :	0.00000	Min. :	0.0000	Min. :	-1.89545	Min. :	0.0000
1st Qu.:	200.0	1st Qu.:	-0.83944	1st Qu.:	0.00000	1st Qu.:	-0.6265
↪ 1st Qu.:	0.00000	1st Qu.:	0.0000	1st Qu.:	-0.86181	1st Qu.:	0.0000
Median :	320.0	Median :	-0.09001	Median :	0.00000	Median :	-0.2098
↪ Median :	0.00000	Median :	0.0000	Median :	-0.03489	Median :	0.0000

```

Mean    : 370.7    Mean    : 0.00000    Mean    :0.06022    Mean    : 0.0000
↪ Mean    :0.05148    Mean    :0.1481    Mean    : 0.00000    Mean    :0.2545
3rd Qu.: 490.0    3rd Qu.: 0.65943    3rd Qu.:0.00000    3rd Qu.: 1.0401
↪ 3rd Qu.:0.00000    3rd Qu.:0.0000    3rd Qu.: 0.68866    3rd Qu.:1.0000
Max.     :2884.5    Max.     : 2.26536    Max.     :1.00000    Max.     : 1.8734
↪ Max.     :1.00000    Max.     :1.0000    Max.     : 2.65259    Max.     :1.0000

lwage
Min.      : -0.2231
1st Qu.: 1.8458
Median : 2.1656
Mean     : 2.1958
3rd Qu.: 2.5257
Max.     : 4.2782

> summary(test)

      husage      husunion      husearns      huseduc
↪ husblck      hushisp      hushrs      kidge6
Min.      : -1.80501  Min.      : 0.0000  Min.      : -1.72343  Min.      : -5.0191
↪ Min.      : 0.00000  Min.      : 0.00000  Min.      : -3.32913  Min.      : 0.000
1st Qu.: -0.70910  1st Qu.: 0.0000  1st Qu.: -0.72231  1st Qu.: -0.5568
↪ 1st Qu.: 0.00000  1st Qu.: 0.00000  1st Qu.: -0.19614  1st Qu.: 0.000
Median : -0.01171  Median : 0.0000  Median : -0.20173  Median : -0.1849
↪ Median : 0.00000  Median : 0.00000  Median : -0.19614  Median : 0.000
Mean     : 0.05078  Mean     : 0.2272  Mean     : -0.03584  Mean     : -0.0167
↪ Mean     : 0.07961  Mean     : 0.06214  Mean     : -0.02625  Mean     : 0.365
3rd Qu.: 0.68568  3rd Qu.: 0.0000  3rd Qu.: 0.42118  3rd Qu.: 0.9306
↪ 3rd Qu.: 0.00000  3rd Qu.: 0.00000  3rd Qu.: 0.46963  3rd Qu.: 1.000
Max.     : 2.47898  Max.     : 1.0000  Max.     : 3.90062  Max.     : 1.6744
↪ Max.     : 1.00000  Max.     : 1.00000  Max.     : 4.42503  Max.     : 1.000

      earns      age      black      educ
↪ hispanic      union      exper      kidlt6
Min.      : 1.0    Min.      : -1.91006  Min.      : 0.00000  Min.      : -5.62636
↪ Min.      : 0.00000  Min.      : 0.0000  Min.      : -1.79209  Min.      : 0.0000

```

```

1st Qu.: 210.0    1st Qu.: -0.73238    1st Qu.: 0.00000    1st Qu.: -0.62650
↪ 1st Qu.: 0.00000    1st Qu.: 0.00000    1st Qu.: -0.65508    1st Qu.: 0.00000
Median : 335.0    Median : 0.01706    Median : 0.00000    Median : -0.62650
↪ Median : 0.00000    Median : 0.00000    Median : -0.03489    Median : 0.00000
Mean   : 372.0    Mean   : 0.06404    Mean   : 0.07767    Mean   : -0.09496
↪ Mean   : 0.07184    Mean   : 0.1379    Mean   : 0.08533    Mean   : 0.2544
3rd Qu.: 479.5    3rd Qu.: 0.76649    3rd Qu.: 0.00000    3rd Qu.: 0.62347
↪ 3rd Qu.: 0.00000    3rd Qu.: 0.00000    3rd Qu.: 0.74034    3rd Qu.: 1.0000
Max.    :1800.0    Max.    : 2.26536    Max.    :1.00000    Max.    : 1.87343
↪ Max.    :1.00000    Max.    :1.0000    Max.    : 2.44586    Max.    :1.0000

lwage
Min.     : -3.401
1st Qu.: 1.874
Median : 2.175
Mean    : 2.200
3rd Qu.: 2.526
Max.     : 4.234
>
> # variaveis do modelo
> cols_reg = c('husage', 'husunion', 'husearns', 'huseduc', 'husblack',
↪ 'hushisp', 'hushrs', 'kidge6', 'age', 'black', 'educ',
+ 'hispanic', 'union', 'exper', 'kidlt6', 'lwage')
>
> # geração de variaveis dummies para organizar os datasets
> #em objetos tipo matriz
>
> fml <- formula(lwage~husage+husunion+husearns+huseduc+husblack+hushisp+
+ hushrs+kidge6+age+black+educ+hispanic+union+
+ exper+kidlt6)
>
> dummies <- dummyVars(fml, data = dat[,cols_reg])
>

```

```

> train_dummies = predict(dummies, newdata = train[,cols_reg])
> test_dummies = predict(dummies, newdata = test[,cols_reg])
> print(dim(train_dummies)); print(dim(test_dummies))
[1] 2059  15
[1] 515  15
>
> x = as.matrix(train_dummies)
> y_train = train$wage
> x_test = as.matrix(test_dummies)
> y_test = test$wage
>
> # 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
+   )
+ }
>
> standardize <- function(value, pre_proc_values, column) {
+   (value-pre_proc_values[["mean"]][[column]])/
+   pre_proc_values[["std"]][[column]]
+ }
>
> husage = standardize(40, pre_proc_val, "husage")

```

```

> husunion = 0
> husearns = standardize(600, pre_proc_val, "husearns")
> huseduc = standardize(13, pre_proc_val, "huseduc")
> husblack = 1
> hushisp = 0
> hushrs = standardize(40, pre_proc_val, "hushrs")
> kidge6 = 1
> age = standardize(38, pre_proc_val, "age")
> black = 0
> educ = standardize(13, pre_proc_val, "educ")
> hispanic = 1
> union = 0
> exper = standardize(18, pre_proc_val, "exper")
> kidlt6 = 1
>
> # Vamos construir uma matriz de dados para a predicao
>
> dflist <- list(husage=husage,
+               husunion=husunion,
+               husearns=husearns,
+               huseduc=huseduc,
+               husblack=husblack,
+               hushisp=hushisp,
+               hushrs=hushrs,
+               kidge6=kidge6,
+               age=age,
+               black=black,
+               educ=educ,
+               hispanic=hispanic,
+               union=union,
+               exper=exper,
+               kidlt6=kidlt6)

```

```

>
> our_pred = as.matrix(data.frame(args=dflist))
>
> confidence_interval <- function(n, m, s) {
+   dam <- s/sqrt(n)
+   cilwr <- m + (qnorm(0.025))*dam
+   ciupr <- m - (qnorm(0.025))*dam
+   ret <- list("lower" = cilwr, "upper" = ciupr)
+   return(ret)
+ }
>
> #####
> #                               REGRESSAO RIDGE                               #
> #####
> # Vamos calcular o valor otimo de lambda;
> lambdas <- 10^seq(2, -3, by = -.1)
> # Calculando o lambda:
> ridge_lamb <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas)
>
> # Vamos ver qual o lambda otimo
> best_lambda_ride <- ridge_lamb$lambda.min
> best_lambda_ride
[1] 0.01584893
>
> # Estimando o modelo Ridge
> ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0, family =
  ↪ 'gaussian', lambda = best_lambda_ride)
>
> # Vamos ver o resultado (valores) da estimativa
> # (coeficientes)
> ridge_reg[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"

```

```

                                s0
husage      -0.0024455313
husunion    0.0006554252
husearns    0.1165514550
huseduc     0.0259588560
husblack    0.1102064740
hushisp     0.0436424524
hushrs      -0.0351909179
kidge6      -0.0920898819
age         0.0350734272
black       -0.1405975710
educ        0.1655944730
hispanic    -0.0057816320
union       0.1850796037
exper       -0.0082137149
kidlt6      -0.0215058607
>
> ridge_predictions_train <- predict(ridge_reg, s = best_lambda_ridge,
  ↪ newx = x)
>
> # As metricas da base de treinamento sao:
> eval_results(y_train, ridge_predictions_train, train)
      RMSE   Rsquare
1 0.425665 0.2921375
>
> # Predicao e avaliacao nos dados de teste:
> ridge_predictions_test <- predict(ridge_reg, s = best_lambda_ridge,
  ↪ newx = x_test)
>
> # As metricas da base de teste sao:
> eval_results(y_test, ridge_predictions_test, test)
      RMSE   Rsquare

```



```

1 0.5006593 0.259081
>
> predict_our_ridge <- predict(ridge_reg, s = best_lambda_ridge,
+                               newx = our_pred)
> # O resultado da predicao eh:
> predict_our_ridge
      s1
[1,] 2.16152
>
> # O resultado eh um valor padronizado, vamos converte-lo
> # para o valor nominal, consistente com o dataset original
> wage_pred_ridge = (predict_our_ridge * sd(dat$lwage)) + mean(dat$lwage)
>
> # O resultado eh:
> wage_pred_ridge
      s1
[1,] 3.32512
>
> CI_ridge <- confidence_interval( nrow(train), wage_pred_ridge,
+   sd(dat$lwage))
>
> CI_ridge$lower
      s1
[1,] 3.30257
> CI_ridge$upper
      s1
[1,] 3.34767

> #####
> #                               REGRESSAO LASSO                               #
> #####
>

```



```

> # Vamos atribuir alpha = 1 para implementar a regressao
> # lasso
> lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
+                         lambda = lambdas, standardize = TRUE, nfolds = 5)
>
> best_lambda_lasso <- lasso_lamb$lambda.min
> best_lambda_lasso
[1] 0.006309573
>
> # Vamos estimar o modelo Lasso
> lasso_model <- glmnet(x, y_train, alpha = 1,
+                      lambda = best_lambda_lasso,
+                      standardize = TRUE)
>
> # Vamos visualizar os coeficientes estimados
> lasso_model[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"

          s0
husage      .
husunion    .
husearns    0.11644518
huseduc     0.01526450
husblck     .
hushisp     .
hushrs      -0.03055660
kidge6      -0.07226852
age         0.02377897
black       -0.01629025
educ        0.17165821
hispanic    .
union       0.17389132
exper       .

```

```

kidlt6      .
>
> lasso_predictions_train <- predict(lasso_model, s = best_lambda_lasso,
  ↪ newx = x)
>
> # As metricas da base de treinamento sao:
> eval_results(y_train, lasso_predictions_train, train)
      RMSE   Rsquare
1 0.426272 0.2901174
>
> # Vamos fazer as predicoes na base de teste
> lasso_predictions_test <- predict(lasso_model, s = best_lambda_lasso,
  ↪ newx = x_test)
>
> # As metricas da base de teste sao:
> eval_results(y_test, lasso_predictions_test, test)
      RMSE   Rsquare
1 0.5009585 0.2581953
>
> # Vamos para a predicao
> predict_our_lasso <- predict(lasso_model, s = best_lambda_lasso, newx =
  ↪ our_pred)
> predict_our_lasso
      s1
[1,] 2.088435
>
> wage_pred_lasso = (predict_our_lasso * sd(dat$lwage)) + mean(dat$lwage)
>
> wage_pred_lasso
      s1
[1,] 3.286964
>

```

```

> CI_lasso <- confidence_interval(
+   nrow(train),
+   wage_pred_lasso,
+   sd(dat$wage))
>
> # O intervalo de confianca eh:
> CI_lasso$lower
      s1
[1,] 3.264414
> CI_lasso$upper
      s1
[1,] 3.309514
>
> #####
> #                               REGRESSAO ELASTICNET                               #
> #####
>
> elastic_predictions_train <- predict(elastic_reg, x)
>
> # As metricas de performance na base de treinamento
> # sao:
> eval_results(y_train, elastic_predictions_train, train)
      RMSE   Rsquare
1 0.4259722 0.2911154
>
> # Vamos fazer as predicoes na base de teste
> elastic_predictions_test <- predict(elastic_reg, x_test)
>
> # As metricas de performance na base de teste sao:
> eval_results(y_test, elastic_predictions_test, test)
      RMSE   Rsquare
1 0.500484 0.2595998

```

```

>
> predict_our_elastic <- predict(elastic_reg, our_pred)
> predict_our_elastic
[1] 2.072409
>
> wage_pred_elastic=(predict_our_elastic * sd(dat$wage)) +
  → mean(dat$wage)
> wage_pred_elastic
[1] 3.278598
>
> CI_elastic <- confidence_interval(
+   nrow(train),
+   wage_pred_elastic,
+   sd(dat$wage))
>
> # O intervalo de confianca eh:
> CI_elastic$lower
[1] 3.256048
> CI_elastic$upper
[1] 3.301147

> # O melhor parametro alpha escolhido eh:
> elastic_reg$bestTune
      alpha      lambda
4 0.378853 0.008377773
>
> # E os parametros sao:
> elastic_reg[["finalModel"]][["beta"]]
15 x 77 sparse Matrix of class "dgCMatrix"

```

APÊNDICE 6 – ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B - RESOLUÇÃO

1 Construção de Características: Identificador automático de idioma

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

espanhol = [
"Hola, ¿cómo estás?",
"Me encanta leer libros.",
"El clima está agradable hoy.",
"¿Dónde está el restaurante más cercano?",
"¿Qué hora es?",
"Voy al parque todos los días.",
```

```

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

```

```

portugues = [
"Estou indo para o trabalho agora.",
"Adoro passar tempo com minha família.",
"Preciso comprar leite e pão.",
"Vamos ao cinema no sábado.",
"Gosto de praticar esportes ao ar livre.",
"O trânsito está terrível hoje.",
"A comida estava deliciosa!",
"Você já visitou o Rio de Janeiro?",
"Tenho uma reunião importante amanhã.",
"A festa começa às 20h.",
"Estou cansado depois de um longo dia de trabalho.",
"Vamos fazer um churrasco no final de semana.",
"O livro que estou lendo é muito interessante.",
"Estou aprendendo a cozinhar pratos novos.",
"Preciso fazer exercícios físicos regularmente.",
"Vou viajar para o exterior nas férias.",
"Você gosta de dançar?",
"Hoje é meu aniversário!",
"Gosto de ouvir música clássica.",
"Estou estudando para o vestibular.",
]

```



```

"Meu time de futebol favorito ganhou o jogo.",
"Quero aprender a tocar violão.",
"Vamos fazer uma viagem de carro.",
"O parque fica cheio aos finais de semana.",
"O filme que assisti ontem foi ótimo.",
"Preciso resolver esse problema o mais rápido possível.",
"Adoro explorar novos lugares.",
"Vou visitar meus avós no domingo.",
"Estou ansioso para as férias de verão.",
"Gosto de fazer caminhadas na natureza.",
"O restaurante tem uma vista incrível.",
"Vamos sair para jantar no sábado.",
]

```

A “amostras” de texto precisa ser “transformada” em padrões

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

```

[ ]: import random

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

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

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

#O DataFrame do pandas facilita a visualização.

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

```

```

[ ]:

```

	0	1
0	Estou aprendendo a cozinhar pratos novos.	português
1	Me gusta bailar salsa.	espanhol
2	I like listening to music.	inglês
3	Meu time de futebol favorito ganhou o jogo.	português
4	Adoro explorar novos lugares.	português
..	...	...
87	I'm sorry for the inconvenience.	inglês
88	Me encanta leer libros.	espanhol
89	O parque fica cheio aos finais de semana.	português


```

90          Where is the nearest restaurant?      inglês
91          Tengo una reunión a las 2 PM.          espanhol

```

```
[92 rows x 2 columns]
```

Construção dos atributos - medição das frases e sentença para extrair características e melhorar o resultado. A matriz de confusão obtida:

```

[ ]:
      0  1  2  3  4
0  5.833333  3  0  0  português
1  4.500000  0  0  0  espanhol
2  4.200000  0  1  0  inglês
3  4.375000  4  0  0  português
4  6.250000  0  0  0  português
..  ...  ..  ..  ..  ...
87  4.333333  0  1  0  inglês
88  4.750000  0  0  1  espanhol
89  4.125000  2  0  0  português
90  5.400000  0  3  0  inglês
91  3.142857  0  0  2  espanhol

```

```
[92 rows x 5 columns]
```

Treinando o modelo com SVM e Separando o conjunto de treinamento do conjunto de testes obtemos:

Acurácia nos dados de treinamento: 95.65%

	precision	recall	f1-score	support
espanhol	0.88	1.00	0.94	23
inglês	1.00	1.00	1.00	22
português	1.00	0.88	0.93	24
accuracy			0.96	69
macro avg	0.96	0.96	0.96	69
weighted avg	0.96	0.96	0.96	69

	precision	recall	f1-score	support
espanhol	0.78	1.00	0.88	7
inglês	1.00	1.00	1.00	8
português	1.00	0.75	0.86	8
accuracy			0.91	23
macro avg	0.93	0.92	0.91	23
weighted avg	0.93	0.91	0.91	23

Atividade 2 - Melhorar o desempenho de RP em conjunto de dados existentes

Base de dados

```
[ ]: Inflation rate      GDP
count      4424.000000  4424.000000
mean        1.228029    0.001969
std         1.382711    2.269935
min        -0.800000   -4.060000
25%         0.300000   -1.700000
50%         1.400000    0.320000
75%         2.600000    1.790000
max         3.700000    3.510000

[8 rows x 36 columns]
```

Tratamento dos dados, Target e Gender removidos

```
[ ]: X_data = students_dropout.iloc[:, :36]
X_data = X_data.drop(['Gender'], axis=1)

#print(X_data.to_string())

Y = students_dropout['Gender']
Y_orig = students_dropout['Gender']

print(Y.unique())
```

[1 0]

Gerado uma variável com dados originais e tratados. Após é realizado a normalização.

```
[ ]: from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd

X= X_data.copy(deep=True)
X_orig = X_data.copy(deep=True)

# normalização min-max
X = pd.DataFrame( minmax_scale(X) )
X.drop_duplicates()

print(X.head())
```

	0	1	2	3	4	5	6	7	8	\
0	0.0	0.285714	0.555556	0.013858	1.0	0.0	0.284211	0.0	0.418605	
1	0.0	0.250000	0.111111	0.925989	1.0	0.0	0.684211	0.0	0.000000	
2	0.0	0.000000	0.555556	0.907512	1.0	0.0	0.284211	0.0	0.837209	
3	0.0	0.285714	0.222222	0.978108	1.0	0.0	0.284211	0.0	0.860465	

	9	...	25	26	27	28	29	30	31	32	\
0	0.255814	...	0.0	0.0	0.000000	0.000000	0.00	0.000000	0.0	0.372093	
1	0.046512	...	0.0	0.0	0.26087	0.181818	0.30	0.735897	0.0	0.732558	
2	0.837209	...	0.0	0.0	0.26087	0.000000	0.00	0.000000	0.0	0.372093	
3	0.837209	...	0.0	0.0	0.26087	0.303030	0.25	0.667692	0.0	0.209302	
4	0.860465	...	0.0	0.0	0.26087	0.181818	0.30	0.700000	0.0	0.732558	

	33	34
0	0.488889	0.766182
1	0.111111	0.640687
2	0.488889	0.766182
3	0.000000	0.124174
4	0.111111	0.640687

[5 rows x 35 columns]

```
[ ]: from sklearn.model_selection import train_test_split
import numpy as np

Y_orig.reset_index(drop=True)

# com os dados originais
X_oring_train, X_oring_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)
```

Treinamento (SVM).

```
[ ]: 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_oring_train, y_orig_train)

# predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(X_oring_train)
cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_orig_train, y_orig_pred))
```

```
# 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))
```

Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO

```
[[2074  77]
 [1076  91]]
```

	precision	recall	f1-score	support
0	0.66	0.96	0.78	2151
1	0.54	0.08	0.14	1167
accuracy			0.65	3318
macro avg	0.60	0.52	0.46	3318
weighted avg	0.62	0.65	0.56	3318

Matriz de confusão - com os dados ORIGINAIS usados para TESTES

```
[[693  24]
 [354  35]]
```

	precision	recall	f1-score	support
0	0.66	0.97	0.79	717
1	0.59	0.09	0.16	389
accuracy			0.66	1106
macro avg	0.63	0.53	0.47	1106
weighted avg	0.64	0.66	0.56	1106

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

```
[ ]: 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)

y_pred = modelo.predict(X_train)
cm_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO')
print(cm_train)
print(classification_report(y_train, y_pred))
```

```

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))

```

Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO

```

[[2019  132]
 [ 799  368]]

```

	precision	recall	f1-score	support
0	0.72	0.94	0.81	2151
1	0.74	0.32	0.44	1167
accuracy			0.72	3318
macro avg	0.73	0.63	0.63	3318
weighted avg	0.72	0.72	0.68	3318

Matriz de confusão - com os dados TRATADOS usados para TESTES

```

[[653  64]
 [284 105]]

```

	precision	recall	f1-score	support
0	0.70	0.91	0.79	717
1	0.62	0.27	0.38	389
accuracy			0.69	1106
macro avg	0.66	0.59	0.58	1106
weighted avg	0.67	0.69	0.64	1106

APÊNDICE 7 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B - RESOLUÇÃO

Seed utilizado: 202490

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

Especificações:

O trabalho pode ser feito por uma equipe de 1 a 6 integrantes.

Para cada problema, preencher as colunas dos quadros com o que pede. Além disso, fazer as solicitações pedidas antes dos quadros.

CLASSIFICAÇÃO

Para o experimento de Classificação:

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

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – CV	size=31 decay=0,7	0.8263	Prediction bus opel saab van bus 42 1 0 1 opel 0 29 12 0 saab 0 11 30 1 van 1 1 1 37
SVM – Hold-out	C=1 Sigma= 0.06828384	0.7784	Prediction bus opel saab van bus 42 0 2 1 opel 0 27 14 0 saab 0 14 24 1 van 1 1 3 37
SVM – CV	C=1 Sigma= 0.06828384	0.7784	Prediction bus opel saab van bus 42 0 2 1 opel 0 27 14 0 saab 0 14 24 1 van 1 1 3 37

RF – CV	mtry=10	0.7246	Prediction bus opel saab van bus 42 1 1 0 opel 0 21 17 1 saab 1 19 21 1 van 0 1 4 37
RF – Hold-out	mtry=2	0.7186	Prediction bus opel saab van bus 42 0 1 0 opel 0 22 21 0 saab 0 17 18 1 van 1 3 3 38
KNN	k=1	0.6647	Prediction bus opel saab van bus 33 3 4 4 opel 0 16 15 1 saab 6 21 22 1 van 2 0 0 42
RNA – Hold-out	size=5 decay=0.1	0.6228	Prediction bus opel saab van bus 30 1 6 3 opel 0 4 1 0 saab 12 31 34 0 van 1 6 2 36

Predição Veículos

C o m p	cir c	D ci c	R ad R a	Pr A xi s R a	M ax L R a	Sc at R a	El on g	Pr A xi s R ec t	M ax L R ec t	Sc Va r M ax is	Sc Va r m ax is	R a G yr	Sk ew M ax is	Sk ew m ax is	K ur tm ax is	K ur tm ax is	H oll R a	Pred ito tipo
85	37	87	15 8	71	5	15 2	42	15	14 0	16 9	30 0	1 8 5	60	7	15	18 5	18 0	bus
11 5	61	11 6	23 0	70	11	31 0	10	25	15 9	12 5	73 0	3 4 8	70	15	8	17 9	20 1	saab
10 5	51	10 7	21 9	67	11	21 8	33	25	15 7	22 5	63 3	2 2 1	74	15	19	18 3	19 7	saab

```
#install.packages("caret")
#install.packages("e1071")
#install.packages("mlbennch")
#install.packages("mice")
library(mlbench)
library(caret)
library(mice)

dados <- read.csv("./csvs/6 - Veiculos - Dados.csv", header=T)
dados$a <- NULL
View(dados)
novos_dados <- read.csv("./csvs/6 - Veiculos - NOVOS-CASOS - Dados.csv",
  ↪ header=T)
novos_dados$a <- NULL
novos_dados$tipo <- NULL
View(novos_dados)

#####
### KNN ###
#####

set.seed(202490)

ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
knn_train <- dados[ran,]
knn_test <- dados[-ran,]
tuneGrid <- expand.grid(k = c(1,3,5,7,9))

set.seed(202490)

knn <- train(tipo~., data=knn_train, method="knn", tuneGrid=tuneGrid)
knn
```



```

#knn$bestTune$k
predict.knn <- predict(knn, knn_test)
cof <- confusionMatrix(predict.knn, as.factor(knn_test$tipo))
accuracy.knn <- cof$overall[[1]]

#####
### SVM Hold-out ###
#####

set.seed(202490)

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

set.seed(202490)

svm <- train(tipo~., data=treino, method="svmRadial")
predicoes.svm <- predict(svm, teste)
cof <- confusionMatrix(predicoes.svm, as.factor(teste$tipo))
accuracy.svm_holdout <- cof$overall[[1]]

#####
### SVM Cross-validation ###
#####

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

set.seed(202490)

svmcrossvalidation <- train(tipo~., data=treino, method="svmRadial",
  ↪ trControl=ctrl)

```

```

svmcrossvalidation
predicoes.crossvalidation <- predict(svmcrossvalidation, teste)
cof <- confusionMatrix(predicoes.crossvalidation, as.factor(teste$tipo))
accuracy.svm_cross <- cof$overall[[1]]

#####
### RNA-Hold-out ###
#####

imp <- mice(dados)
dados_rna <- complete(imp, 1)

set.seed(202490)

rna_indices <- createDataPartition(dados_rna$tipo, p=0.80, list=FALSE)
rna_train <- dados_rna[rna_indices,]
rna_test <- dados_rna[-rna_indices,]

set.seed(202490)

rna <- train(tipo~., data=rna_train, method="nnet", trace=FALSE)
predict.rna <- predict(rna, rna_test)
cof <- confusionMatrix(predict.rna, as.factor(rna_test$tipo))
accuracy.rna_holdout <- cof$overall[[1]]

#####
### RNA Cross-validation ###
#####

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202490)

```

```

rna_crossvalidation <- train(tipo~., data=rna_train,
  ↪ method="nnet",trace=FALSE, trControl=ctrl)
predict.rnacrossvalidation <- predict(rna_crossvalidation, rna_test)
confusionMatrix(predict.rnacrossvalidation, as.factor(rna_test$tipo))
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
  ↪ seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202490)
rna_crossvalidation <- train(form=tipo~., data=rna_train, method="nnet",
  ↪ tuneGrid=grid, trControl=ctrl, maxit = 2000, trace=FALSE)
rna_crossvalidation
predict.rnacrossvalidation <- predict(rna_crossvalidation, rna_test)
cof <- confusionMatrix(predict.rnacrossvalidation,
  ↪ as.factor(rna_test$tipo))
accuracy.rna_cross <- cof$overall[[1]]
predict.rnacrossvalidation_new <- predict(rna_crossvalidation,
  ↪ novos_dados)
result.rnacrossvalidation_new <- cbind(novos_dados,
  ↪ predict.rnacrossvalidation_new)
View(result.rnacrossvalidation_new)

#####
### RF Hold-out ###
#####

set.seed(202490)
rf <- train(tipo~., data=treino, method="rf")
rf
predicoes.rf <- predict(rf, teste)
cof <- confusionMatrix(predicoes.rf, as.factor(teste$tipo))
accuracy.rf_holdout <- cof$overall[[1]]

#####

```

```

### RF CV ###
#####

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202490)
rf_cross <- train(tipo~., data=treino, method="rf", trControl=ctrl)
rf_cross
predict.rf_cross <- predict(rf_cross, teste)
confusionMatrix(predict.rf_cross, as.factor(teste$tipo))
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202490)
rf_cross <- train(tipo~., data=treino, method="rf", trControl=ctrl,
  ↪ tuneGrid=tuneGrid)
rf_cross
predict.rf_cross <- predict(rf_cross, teste)
cof <- confusionMatrix(predict.rf_cross, as.factor(teste$tipo))
cof
accuracy.rf_cross <- cof$overall[[1]]

### Results ###

df <- data.frame(
  Tecnica = c("KNN",
    "SVM - Hold-Out",
    "SVM - CV",
    "RNA - Hold-out",
    "RNA - CV",
    "RF - Hold-out",
    "RF - CV"),
  Parametro = c(knn$bestTune$k,
    paste("C=", svm$bestTune$C, " ", "Sigma=",
  ↪ svm$bestTune$sigma, sep=""),

```

```

paste("C=", svmcrossvalidation$bestTune$C, " ", "Sigma=",
  ↪ svmcrossvalidation$bestTune$sigma, sep=""),
  ↪ paste("Size=", rna$bestTune$size, " ", "Decay=",
  ↪ rna$bestTune$decay, sep=""),
paste("Size=", rna_crossvalidation$bestTune$size, " ",
  ↪ "Decay=", na_crossvalidation$bestTune$decay, sep=""),
  ↪ rf$bestTune$mtry, cross$bestTune$mtry),
Acuracia = c(accuracy.knn,
  accuracy.svm_holdout,
  accuracy.svm_cross,
  accuracy.rna_holdout,
  accuracy.rna_cross,
  accuracy.rf_holdout,
  accuracy.rf_cross)
)
df

```

Diabetes

```

#install.packages("caret")
#install.packages("e1071")
#install.packages("mlbennch")
#install.packages("mice")
library(mlbench)
library(caret)
library(mice)

dados <- read.csv("./csvs/10 - Diabetes - Dados.csv", header=T)
dados$num <- NULL
View(dados)

novos_dados <- read.csv("./csvs/10 - Diabetes - NOVOS-CASOS - Dados.csv",
  ↪ header=T)

```

```

novos_dados$num <- NULL
novos_dados$diabetes <- NULL

View(novos_dados)

#####
### KNN ###
#####

set.seed(202490)

ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
knn_train <- dados[ran,]
knn_test <- dados[-ran,]
tuneGrid <- expand.grid(k = c(1,3,5,7,9))

set.seed(202490)

knn <- train(diabetes~., data=knn_train, method="knn", tuneGrid=tuneGrid)
knn

#knn$bestTune$k

predict.knn <- predict(knn, knn_test)
cof <- confusionMatrix(predict.knn, as.factor(knn_test$diabetes))
cof

accuracy.knn <- cof$overall[[1]]

#####
### SVM Hold-out ###
#####

```

```

set.seed(202490)

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

set.seed(202490)

svm <- train(diabetes~., data=treino, method="svmRadial")
svm
predicoes.svm <- predict(svm, teste)
cof <- confusionMatrix(predicoes.svm, as.factor(teste$diabetes))
cof
accuracy.svm_holdout <- cof$overall[[1]]
predict.svm_new <- predict(svm, novos_dados)
result.svm_new <- cbind(novos_dados, predict.svm_new)
View(result.svm_new)

#####
### SVM Cross-validation ###
#####

ctrl <- trainControl(method="cv", number=10)
set.seed(202490)

svmcrossvalidation <- train(diabetes~., data=treino, method="svmRadial",
  ↪ trControl=ctrl)
svmcrossvalidation
predicoes.crossvalidation <- predict(svmcrossvalidation, teste)
cof <- confusionMatrix(predicoes.crossvalidation,
  ↪ as.factor(teste$diabetes))
cof

```



```

accuracy.svm_cross <- cof$overall[[1]]

#####
### RNA-Hold-out ###
#####

imp <- mice(dados)
dados_rna <- complete(imp, 1)

set.seed(202490)

rna_indices <- createDataPartition(dados_rna$diabetes, p=0.80,
  ↪ list=FALSE)
rna_train <- dados_rna[rna_indices,]
rna_test <- dados_rna[-rna_indices,]

set.seed(202490)

rna <- train(diabetes~., data=rna_train, method="nnet", trace=FALSE)
rna
predict.rna <- predict(rna, rna_test)
cof <- confusionMatrix(predict.rna, as.factor(rna_test$diabetes))
cof
accuracy.rna_holdout <- cof$overall[[1]]

#####
### RNA Cross-validation ###
#####

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

set.seed(202490)

```



```

rna_crossvalidation <- train(diabetes~., data=rna_train,
  ↪ method="nnet", trace=FALSE, trControl=ctrl)
predict.rnacrossvalidation <- predict(rna_crossvalidation, rna_test)
confusionMatrix(predict.rnacrossvalidation, as.factor(rna_test$diabetes))
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10), decay =
  ↪ seq(from = 0.1, to = 0.9, by = 0.3))

```

```
set.seed(202490)
```

```

rna_crossvalidation <- train(form=diabetes~., data=rna_train,
  ↪ method="nnet", tuneGrid=grid, trControl=ctrl, maxit = 2000,
  ↪ trace=FALSE)
rna_crossvalidation

```

```

predict.rnacrossvalidation <- predict(rna_crossvalidation, rna_test)
cof <- confusionMatrix(predict.rnacrossvalidation,
  ↪ as.factor(rna_test$diabetes))
cof
accuracy.rna_cross <- cof$overall[[1]]

```

```

#####
### RF Hold-out ###
#####

```

```
set.seed(202490)
```

```

rf <- train(diabetes~., data=treino, method="rf")
rf
predicoes.rf <- predict(rf, teste)
cof <- confusionMatrix(predicoes.rf, as.factor(teste$diabetes))
cof

```

```

accuracy.rf_holdout <- cof$overall[[1]]

#####
### RF CV ###
#####

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202490)

rf_cross <- train(diabetes~., data=treino, method="rf", trControl=ctrl)
rf_cross
predict.rf_cross <- predict(rf_cross, teste)
confusionMatrix(predict.rf_cross, as.factor(teste$diabetes))

tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202490)

rf_cross <- train(diabetes~., data=treino, method="rf", trControl=ctrl,
  ↪ tuneGrid=tuneGrid)
rf_cross

predict.rf_cross <- predict(rf_cross, teste)
cof <- confusionMatrix(predict.rf_cross, as.factor(teste$diabetes))
cof

accuracy.rf_cross <- cof$overall[[1]]

### Results ###

df <- data.frame(
  Tecnica = c("KNN",

```

```

    "SVM - Hold-Out",
    "SVM - CV",
    "RNA - Hold-out",
    "RNA - CV",
    "RF - Hold-out",
    "RF - CV"),
Parametro = c(knn$bestTune$k,
              paste("C=", svm$bestTune$C, " ", "Sigma=",
                    ↪ svm$bestTune$sigma, sep=""),
              paste("C=", svmcrossvalidation$bestTune$C, " ", "Sigma=",
                    ↪ svmcrossvalidation$bestTune$sigma, sep=""),
              paste("Size=", rna$bestTune$size, " ", "Decay=",
                    ↪ rna$bestTune$decay, sep=""),
              paste("Size=", rna_crossvalidation$bestTune$size, " ",
                    ↪ "Decay=", rna_crossvalidation$bestTune$decay,
                    ↪ sep=""),
              rf$bestTune$mtry,
              rf_cross$bestTune$mtry),
Acuracia = c(accuracy.knn,
             accuracy.svm_holdout,
             accuracy.svm_cross,
             accuracy.rna_holdout,
             accuracy.rna_cross,
             accuracy.rf_holdout,
             accuracy.rf_cross)
)
df

```

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – Hold-out	C= 0.25 Sigma= 0.09755161	0.7516	Prediction neg pos neg 93 31 pos 7 22
SVM – CV	C= 0.5 Sigma= 0.09755161	0.7451	Prediction neg pos neg 91 30 pos 9 23

RF – CV	mtry=2	0.7451	Prediction neg pos neg 90 29 pos 10 24
RF – Hold-out	mtry=2	0.7255	Prediction neg pos neg 89 31 pos 11 22
RNA – CV	size=11 decay=0.1	0.719	Prediction neg pos neg 87 30 pos 13 23
KNN	k=9	0.6883	Prediction neg pos neg 85 26 pos 22 21
RNA – Hold-out	size=1 decay=0.1	0.6471	Prediction neg pos neg 96 50 pos 4 3

Predição Diabetes

preg0 nt	gluco se	press ure	tricep s	insuli n	mass	pedig ree	age	diabet es
4	200	75	45	0	35.7	0.627	35	pos
5	80	60	0	0	28.1	0.351	25	neg
7	170	65	30	0	22.1	0.562	60	pos

REGRESSÃO

Para o experimento de Regressão:

- Ordenar por R2 decendente, 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

Regressão Admissão

```

if (!require("caret")) install.packages("caret")
if (!require("Metrics")) install.packages("Metrics")
if (!require("e1071")) install.packages("e1071")
if (!require("mlbench")) install.packages("mlbench")
if (!require("mice")) install.packages("mice")
if (!require("kernlab")) install.packages("kernlab")
if (!require("Fgmutils")) install.packages('Fgmutils')

library(caret)
library(Metrics)
library(e1071)
library(mlbench)
library(mice)
library(kernlab)
library(Fgmutils)

r2 <- function(predicted, watched) {
  return(1 - (sum((predicted-watched)^2) /
    ↪ sum((watched-mean(watched))^2)))
}

syx <- function(y_real, y_pred) {
  return (sqrt(sum((y_real - y_pred)^2) / (length(y_real) -2 )))
}

data <- read.csv("./csvs/9 - Admissao - Dados.csv", header=T)
View(data)

new_data <- read.csv("./csvs/9 - Admissao - NOVOS-CASOS - Dados.csv",
  ↪ header=T)
new_data$ChanceOfAdmit <- NULL

data$num <- NULL

```

```

set.seed(202490)

index <- createDataPartition(data$ChanceOfAdmit, p=0.80, list=FALSE)

data_train <- data[index,]
data_test <- data[-index,]

#####
### KNN ###
#####

tuneGrid <- expand.grid(k = c(1,3,5,7,9))

knn <- train(ChanceOfAdmit~., data=data_test, method="knn",
  ↪ tuneGrid=tuneGrid)
knn

predict.knn <- predict(knn, data_test)

result.knn.rmse = rmse(data_test$ChanceOfAdmit, predict.knn)
result.knn.r2 = r2(predict.knn, data_test$ChanceOfAdmit)
result.knn.syx = syx(data_test$ChanceOfAdmit, predict.knn)
result.knn.mae = mae(data_test$ChanceOfAdmit, predict.knn)
result.knn.pearson = cor(data_test$ChanceOfAdmit, predict.knn, method =
  ↪ "pearson")
View(new_data)

predict.knn_new <- predict(knn, new_data)

result.knn_new <- cbind(new_data, predict.knn_new)
View(result.knn_new)

```

```
#####
### RNA-Hold-out ###
#####
```

```
rna <- train(ChanceOfAdmit~., data=data_train, method="nnet", linout=T,
  ↪ trace=FALSE)
rna
```

```
predict.rna <- predict(rna, data_test)
```

```
result.rna.rmse = rmse(data_test$ChanceOfAdmit, predict.rna)
result.rna.r2 = r2(predict.rna, data_test$ChanceOfAdmit)
result.rna.syx = syx(data_test$ChanceOfAdmit, predict.rna)
result.rna.mae = mae(data_test$ChanceOfAdmit, predict.rna)
result.rna.pearson = cor(data_test$ChanceOfAdmit, predict.rna, method =
  ↪ "pearson")
```

```
predict.rna_new <- predict(rna, new_data)
result.rna_new <- cbind(new_data, predict.rna_new)
View(result.rna_new)
```

```
#####
### RNA Cross-validation ###
#####
```

```
control <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(size = seq(from = 1, to = 10, by = 1), decay =
  ↪ seq(from = 0.1, to = 0.9, by = 0.3))
```

```
rna_cv <- train(ChanceOfAdmit~.,
  data=data_train,
  method="nnet",
```

```

        trainControl=control,
        tuneGrid=tuneGrid,
        linout=T,
        MaxNWts=10000,
        maxit=2000,
        trace=F)

rna_cv

predict.rna_cv <- predict(rna_cv, data_test)
result.rna_cv.rmse = rmse(data_test$ChanceOfAdmit, predict.rna_cv)
result.rna_cv.r2 = r2(predict.rna_cv, data_test$ChanceOfAdmit)
result.rna_cv.syx = syx(data_test$ChanceOfAdmit, predict.rna_cv)
result.rna_cv.mae = mae(data_test$ChanceOfAdmit, predict.rna_cv)
result.rna_cv.pearson = cor(data_test$ChanceOfAdmit, predict.rna_cv,
  ↪ method = "pearson")

predict.rna_cv_new <- predict(rna_cv, new_data)
result.rna_cv_new <- cbind(new_data, predict.rna_cv_new)
View(result.rna_cv_new)

res <- resid(rna_cv)
res

plot(fitted(rna_cv), res, xlab="Chance Of Admit")
abline(0,0)

#####
### SVM Hold-out ###
#####

svm <- train(ChanceOfAdmit~., data=data_train, method = "svmRadial")
svm

```



```

predict.svm <- predict(svm, data_test)

result.svm.rmse = rmse(data_test$ChanceOfAdmit, predict.svm)
result.svm.r2 = r2(predict.svm, data_test$ChanceOfAdmit)
result.svm.syx = syx(data_test$ChanceOfAdmit, predict.svm)
result.svm.mae = mae(data_test$ChanceOfAdmit, predict.svm)
result.svm.pearson = cor(data_test$ChanceOfAdmit, predict.svm, method =
  ↪ "pearson")

#####
### SVM Cross-validation ###
#####

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

svm_cv <- train(ChanceOfAdmit~., data=data_train, method="svmRadial",
  ↪ trControl=ctrl)
svm_cv

predict.svm_cv <- predict(svm_cv, data_test)

rmse(data_test$ChanceOfAdmit, predict.svm_cv)
r2(predict.svm_cv, data_test$ChanceOfAdmit)
syx(data_test$ChanceOfAdmit, predict.svm_cv)
mae(data_test$ChanceOfAdmit, predict.svm_cv)
cor(data_test$ChanceOfAdmit, predict.svm_cv, method = "pearson")

#####

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

```

```

set.seed(1912)

svm_sigma <- train(ChanceOfAdmit~., data=data_train, method="svmRadial",
  ↪ trControl=ctrl, tuneGrid=tuneGrid)
svm_sigma

predict.svm_sigma <- predict(svm_sigma, data_test)
result.svm_sigma.rmse = rmse(data_test$ChanceOfAdmit, predict.svm_sigma)
result.svm_sigma.r2 = r2(predict.svm_sigma, data_test$ChanceOfAdmit)
result.svm_sigma.syx = syx(data_test$ChanceOfAdmit, predict.svm_sigma)
result.svm_sigma.mae = mae(data_test$ChanceOfAdmit, predict.svm_sigma)
result.svm_sigma.pearson = cor(data_test$ChanceOfAdmit,
  ↪ predict.svm_sigma, method = "pearson")

predict.new_svm_sigma <- predict(svm_sigma, new_data)
result.svm_sigma_new <- cbind(new_data, predict.new_svm_sigma)
View(result.svm_sigma_new)

#####
### RF Hold-out ###
#####

rf <- train(ChanceOfAdmit~., data=data_train, method="rf")
rf

predict.rf <- predict(rf, data_test)

result.rf.rmse = rmse(data_test$ChanceOfAdmit, predict.rf)
result.rf.r2 = r2(predict.rf, data_test$ChanceOfAdmit)
result.rf.syx = syx(data_test$ChanceOfAdmit, predict.rf)
result.rf.mae = mae(data_test$ChanceOfAdmit, predict.rf)
result.rf.pearson = cor(data_test$ChanceOfAdmit, predict.rf, method =
  ↪ "pearson")

```

```
#####
### RF CV ###
#####

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

rf_cv <- train(ChanceOfAdmit~., data=data_train, method="rf",
  ↪ trControl=ctrl)
rf_cv

predict.rf_cv <- predict(rf_cv, data_test)

rmse(data_test$ChanceOfAdmit, predict.rf_cv)
r2(predict.rf_cv, data_test$ChanceOfAdmit)
syx(data_test$ChanceOfAdmit, predict.rf_cv)
mae(data_test$ChanceOfAdmit, predict.rf_cv)
cor(data_test$ChanceOfAdmit, predict.rf_cv, method = "pearson")

#####

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

rf_mtry <- train(ChanceOfAdmit~., data=data_train, method="rf",
  ↪ trControl=ctrl, tuneGrid=tuneGrid)
rf_mtry

predict.rf_mtry <- predict(rf_mtry, data_test)

result.rf_mtry.rmse = rmse(data_test$ChanceOfAdmit, predict.rf_mtry)
result.rf_mtry.r2 = r2(predict.rf_mtry, data_test$ChanceOfAdmit)
result.rf_mtry.syx = syx(data_test$ChanceOfAdmit, predict.rf_mtry)
```

```

result.rf_mtry.mae = mae(data_test$ChanceOfAdmit, predict.rf_mtry)
result.rf_mtry.pearson = cor(data_test$ChanceOfAdmit, predict.rf_mtry,
  ↪ method = "pearson")

```

```

predict.new_rf_mtry <- predict(rf_mtry, new_data)
result.rf_mtry_new <- cbind(new_data, predict.new_rf_mtry)
View(result.rf_mtry_new)

```

```

df <- data.frame(
  Tecnica = c("KNN",
    "RNA - Hold-out",
    "RNA - CV",
    "SVM - Hold-Out",
    "SVM - CV",
    "RF - Hold-out",
    "RF - CV"),
  Parametro = c(knn$bestTune$k,
    paste("Size=", rna$bestTune$size, " ", "Decay=",
  ↪ rna$bestTune$decay, sep=""),
    paste("Size=", rna_cv$bestTune$size, " ", "Decay=",
  ↪ rna_cv$bestTune$decay, sep=""),
    paste("C=", svm$bestTune$C, " ", "Sigma=",
  ↪ svm$bestTune$sigma, sep=""),
    paste("C=", svm_sigma$bestTune$C, " ", "Sigma=",
  ↪ svm_sigma$bestTune$sigma, sep=""),
    rf$bestTune$mtry,
    rf_mtry$bestTune$mtry),
  R2 = c(result.knn.r2,
    result.rna.r2,
    result.rna_cv.r2,
    result.svm.r2,
    result.svm_sigma.r2,

```

```

        result.rf.r2,
        result.rf_mtry.r2),
SYX = c(result.knn.syx,
        result.rna.syx,
        result.rna_cv.syx,
        result.svm.syx,
        result.svm_sigma.syx,
        result.rf.syx,
        result.rf_mtry.syx),
Pearson = c(result.knn.pearson,
            result.rna.pearson,
            result.rna_cv.pearson,
            result.svm.pearson,
            result.svm_sigma.pearson,
            result.rf.pearson,
            result.rf_mtry.pearson),
RMSE = c(result.knn.rmse,
        result.rna.rmse,
        result.rna_cv.rmse,
        result.svm.rmse,
        result.svm_sigma.rmse,
        result.rf.rmse,
        result.rf_mtry.rmse),
MAE = c(result.knn.mae,
        result.rna.mae,
        result.rna_cv.mae,
        result.svm.mae,
        result.svm_sigma.mae,
        result.rf.mae,
        result.rf_mtry.mae)
)
df

```

Regressão de Biomassa

```

if (!require("caret")) install.packages("caret")
if (!require("Metrics")) install.packages("Metrics")
if (!require("e1071")) install.packages("e1071")
if (!require("mlbench")) install.packages("mlbench")
if (!require("mice")) install.packages("mice")
if (!require("kernlab")) install.packages("kernlab")

library(caret)
library(Metrics)
library(e1071)
library(mlbench)
library(mice)
library(kernlab)

r2 <- function(predicted, watched) {
  return(1 - (sum((predicted-watched)^2) /
    ↪ sum((watched-mean(watched))^2)))
}

syx <- function(y_real, y_pred) {
  return (sqrt(sum((y_real - y_pred)^2) / (length(y_real) -2 )))
}

#mae <- function(y_real, y_pred) {
#  return (1 / length(y_real) * (sum(y_real - y_pred)))
#}

#cor.test(iris$Sepal.Length, iris$Petal.Length, method = "pearson")

data <- read.csv("./csvs/5 - Biomassa - Dados.csv", header=T)
View(data)

```

```

new_data <- read.csv("./csvs/5 - Biomassa - NOVOS-CASOS - Dados.csv",
  ↪ header=T)
new_data$biomassa <- NULL

set.seed(202490)

index <- createDataPartition(data$biomassa, p=0.80, list=FALSE)

data_train <- data[index,]
data_test <- data[-index,]

#####
### KNN ###
#####

tuneGrid <- expand.grid(k = c(1,3,5,7,9))

knn <- train(biomassa~., data=data_test, method="knn", tuneGrid=tuneGrid)
knn

predict.knn <- predict(knn, data_test)

result.knn.rmse = rmse(data_test$biomassa, predict.knn)
result.knn.r2 = r2(predict.knn, data_test$biomassa)
result.knn.syx = syx(data_test$biomassa, predict.knn)
result.knn.mae = mae(data_test$biomassa, predict.knn)
result.knn.pearson = cor(data_test$biomassa, predict.knn, method =
  ↪ "pearson")

View(new_data)

```



```

predict.knn_new <- predict(knn, new_data)

result.knn_new <- cbind(new_data, predict.knn_new)
View(result.knn_new)

#####
### RNA-Hold-out ###
#####

rna <- train(biomassa~., data=data_train, method="nnet", linout=T,
  ↪ trace=FALSE)
rna

predict.rna <- predict(rna, data_test)

result.rna.rmse = rmse(data_test$biomassa, predict.rna)
result.rna.r2 = r2(predict.rna, data_test$biomassa)
result.rna.syx = syx(data_test$biomassa, predict.rna)
result.rna.mae = mae(data_test$biomassa, predict.rna)
result.rna.pearson = cor(data_test$biomassa, predict.rna, method =
  ↪ "pearson")

predict.rna_new <- predict(rna, new_data)
result.rna_new <- cbind(new_data, predict.rna_new)
View(result.rna_new)

#####
### RNA Cross-validation ###
#####

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

```



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

```
rna_cv <- train(biomassa~.,
               data=data_train,
               method="nnet",
               trainControl=control,
               tuneGrid=tuneGrid,
               linout=T,
               MaxNWts=10000,
               maxit=2000,
               trace=F)
```

```
rna_cv
```

```
predict.rna_cv <- predict(rna_cv, data_test)
```

```
result.rna_cv.rmse = rmse(data_test$biomassa, predict.rna_cv)
result.rna_cv.r2 = r2(predict.rna_cv, data_test$biomassa)
result.rna_cv.syx = syx(data_test$biomassa, predict.rna_cv)
result.rna_cv.mae = mae(data_test$biomassa, predict.rna_cv)
result.rna_cv.pearson = cor(data_test$biomassa, predict.rna_cv, method =
  ↪ "pearson")
```

```
predict.rna_cv_new <- predict(rna_cv, new_data)
result.rna_cv_new <- cbind(new_data, predict.rna_cv_new)
View(result.rna_cv_new)
```

```
#####
### SVM Hold-out ###
#####
```

```
svm <- train(biomassa~., data=data_train, method = "svmRadial")
```

```
svm
```

```
predict.svm <- predict(svm, data_test)
```

```
result.svm.rmse = rmse(data_test$biomassa, predict.svm)
```

```
result.svm.r2 = r2(predict.svm, data_test$biomassa)
```

```
result.svm.syx = syx(data_test$biomassa, predict.svm)
```

```
result.svm.mae = mae(data_test$biomassa, predict.svm)
```

```
result.svm.pearson = cor(data_test$biomassa, predict.svm, method =  
↪ "pearson")
```

```
#####
```

```
### SVM Cross-validation ###
```

```
#####
```

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

```
svm_cv <- train(biomassa~., data=data_train, method="svmRadial",  
↪ trControl=ctrl)
```

```
svm_cv
```

```
predict.svm_cv <- predict(svm_cv, data_test)
```

```
rmse(data_test$biomassa, predict.svm_cv)
```

```
r2(predict.svm_cv, data_test$biomassa)
```

```
syx(data_test$biomassa, predict.svm_cv)
```

```
mae(data_test$biomassa, predict.svm_cv)
```

```
cor(data_test$biomassa, predict.svm_cv, method = "pearson")
```

```
#####
```

```

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

set.seed(1912)

svm_sigma <- train(biomassa~., data=data_train, method="svmRadial",
  ↪ trControl=ctrl, tuneGrid=tuneGrid)

svm_sigma

predict.svm_sigma <- predict(svm_sigma, data_test)
result.svm_sigma.rmse = rmse(data_test$biomassa, predict.svm_sigma)
result.svm_sigma.r2 = r2(predict.svm_sigma, data_test$biomassa)
result.svm_sigma.syx = syx(data_test$biomassa, predict.svm_sigma)
result.svm_sigma.mae = mae(data_test$biomassa, predict.svm_sigma)
result.svm_sigma.pearson = cor(data_test$biomassa, predict.svm_sigma,
  ↪ method = "pearson")

predict.new_svm_sigma <- predict(svm_sigma, new_data)
result.svm_sigma_new <- cbind(new_data, predict.new_svm_sigma)
View(result.svm_sigma_new)

res <- ((data_test$biomassa - predict.svm_sigma)/data_test$biomassa) *100
res

plot(data_test$biomassa, res, xlab="Biomassa")
abline(0,0)

#####

### RF Hold-out ###

#####

rf <- train(biomassa~., data=data_train, method="rf")
rf

predict.rf <- predict(rf, data_test)
result.rf.rmse = rmse(data_test$biomassa, predict.rf)
result.rf.r2 = r2(predict.rf, data_test$biomassa)
result.rf.syx = syx(data_test$biomassa, predict.rf)
result.rf.mae = mae(data_test$biomassa, predict.rf)

```

```

result.rf.pearson = cor(data_test$biomassa, predict.rf, method =
  ↪ "pearson")

#####

### RF CV ###

#####

ctrl <- trainControl(method = "cv", number = 10)
rf_cv <- train(biomassa~., data=data_train, method="rf", trControl=ctrl)
rf_cv
predict.rf_cv <- predict(rf_cv, data_test)
rmse(data_test$biomassa, predict.rf_cv)
r2(predict.rf_cv, data_test$biomassa)
syx(data_test$biomassa, predict.rf_cv)
mae(data_test$biomassa, predict.rf_cv)
cor(data_test$biomassa, predict.rf_cv, method = "pearson")

#####

tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
rf_mtry <- train(biomassa~., data=data_train, method="rf",
  ↪ trControl=ctrl, tuneGrid=tuneGrid)
rf_mtry
predict.rf_mtry <- predict(rf_mtry, data_test)

result.rf_mtry.rmse = rmse(data_test$biomassa, predict.rf_mtry)
result.rf_mtry.r2 = r2(predict.rf_mtry, data_test$biomassa)
result.rf_mtry.syx = syx(data_test$biomassa, predict.rf_mtry)
result.rf_mtry.mae = mae(data_test$biomassa, predict.rf_mtry)
result.rf_mtry.pearson = cor(data_test$biomassa, predict.rf_mtry, method
  ↪ = "pearson")

```

```

predict.new_rf_mtry <- predict(rf_mtry, new_data)
result.rf_mtry_new <- cbind(new_data, predict.new_rf_mtry)
View(result.rf_mtry_new)

df <- data.frame(
  Tecnica = c("KNN",
              "RNA - Hold-out",
              "RNA - CV",
              "SVM - Hold-Out",
              "SVM - CV",
              "RF - Hold-out",
              "RF - CV"),
  Parametro = c(knn$bestTune$k,
                paste("Size=", rna$bestTune$size, " ", "Decay=",
                      ↪ rna$bestTune$decay, sep=""),
                paste("Size=", rna_cv$bestTune$size, " ", "Decay=",
                      ↪ rna_cv$bestTune$decay, sep=""),
                paste("C=", svm$bestTune$C, " ", "Sigma=",
                      ↪ svm$bestTune$sigma, sep=""),
                paste("C=", svm_sigma$bestTune$C, " ", "Sigma=",
                      ↪ svm_sigma$bestTune$sigma, sep=""),
                rf$bestTune$mtry,
                rf_mtry$bestTune$mtry),
  R2 = c(result.knn.r2,
        result.rna.r2,
        result.rna_cv.r2,
        result.svm.r2,
        result.svm_sigma.r2,
        result.rf.r2,
        result.rf_mtry.r2),
  SYX = c(result.knn.syx,
          result.rna.syx,

```

```

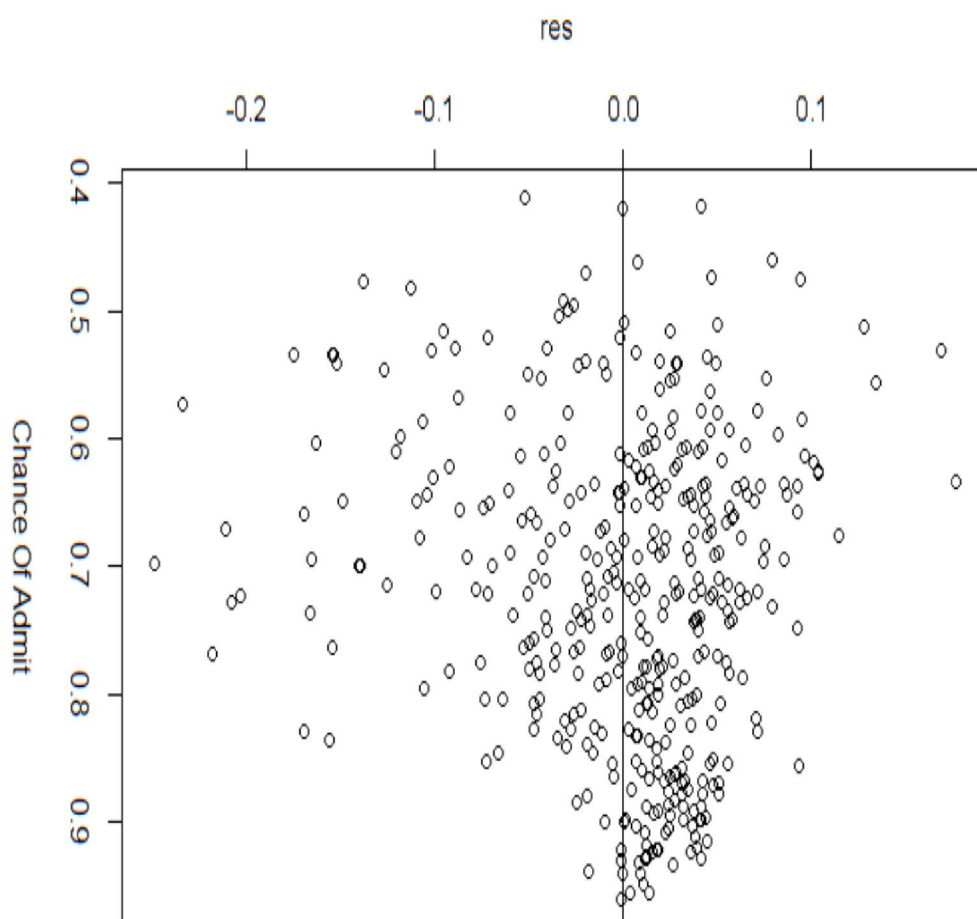
    result.rna_cv.syx,
    result.svm.syx,
    result.svm_sigma.syx,
    result.rf.syx,
    result.rf_mtry.syx),
Pearson = c(result.knn.pearson,
            result.rna.pearson,
            result.rna_cv.pearson,
            result.svm.pearson,
            result.svm_sigma.pearson,
            result.rf.pearson,
            result.rf_mtry.pearson),
RMSE = c(result.knn.rmse,
         result.rna.rmse,
         result.rna_cv.rmse,
         result.svm.rmse,
         result.svm_sigma.rmse,
         result.rf.rmse,
         result.rf_mtry.rmse),
MAE = c(result.knn.mae,
        result.rna.mae,
        result.rna_cv.mae,
        result.svm.mae,
        result.svm_sigma.mae,
        result.rf.mae,
        result.rf_mtry.mae)
)
df

```

Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RNA – CV	size=10 decay=0.1	0.8370628	0.05419331	0.9179804	0.05363747	0.04011106
RF – CV	mtry=2	0.8277202	0.05572535	0.91234	0.05515379	0.03889543
SVM – Hold-out	C=0.5 Sigma=0.1730883	0.8251234	0.05614375	0.909074	0.05556791	0.03977015

RNA – Hold-out	size=5 decay=0.1	0.8242099	0.05629019	0.9110211	0.05571284	0.04164856
RF – Hold-out	mtry=2	0.8234332	0.05641442	0.9103035	0.0558358	0.03989843
SVM – CV	C= 1 Sigma=0.1710585	0.8224601	0.05656965	0.9070763	0.05598944	0.03940272
KNN	K=7	0.7776499	0.06330737	0.8823635	0.06265805	0.04883382

Gráfico de resíduos

Predição Admissão

GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Predicted Chance Of Admitt
300	108	3	3.5	3.5	7.77	0	0.6033419
354	119	4	5.0	4.5	7.50	1	0.6536553
200	120	5	5.0	5.0	8.00	1	0.6758379

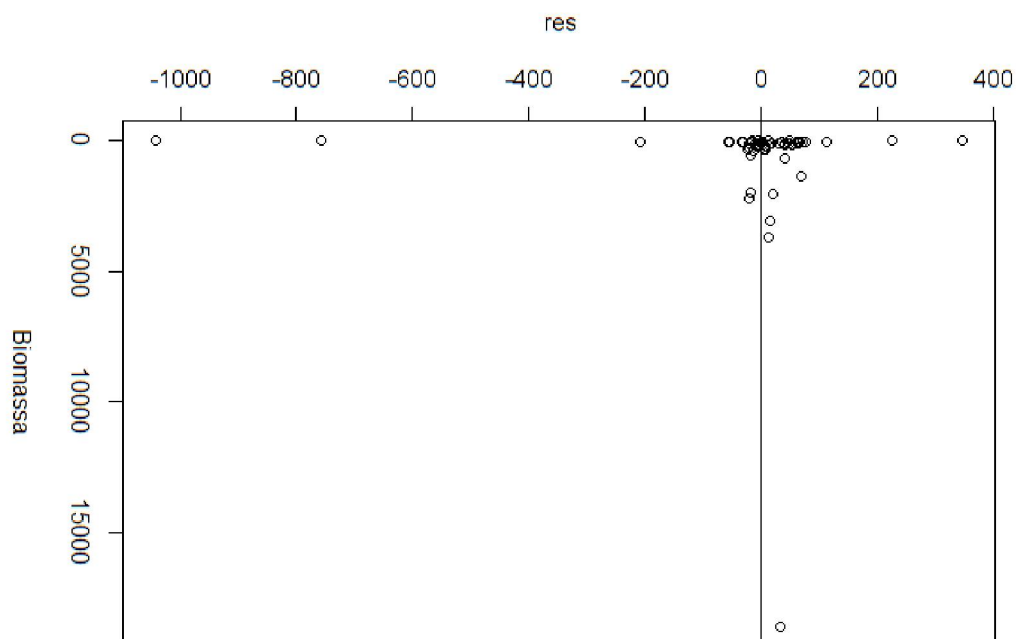
Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV	C=50 Sigma=0.01	0.8822678	855.3966	0.9894328	841.0192	189.6646
RNA – Hold-out	size=5 decay=0.1	0.7580049	1226.3749	0.9898464	1205.7621	291.6157
RNA – CV	size=7 decay=0.4	0.7019301	1361.0656	0.9671596	1338.1889	256.3490
KNN	K=3	0.6902221	1387.5390	0.9560010	1364.2173	260.6396
RF – Hold-out	mtry=2	0.6590796	1455.6148	0.9586842	1431.1490	258.2948
RF – CV	mtry=2	0.6515169	1471.6714	0.9533632	1446.9357	260.0076
SVM – Hold-out	C=1 Sigma=0.744	0.1290280	2326.6020	0.4455273	2287.4966	406.4971

Predição Biomassa

dap	h	Me	Predicted biomassa
6.6	7.5	2.00	546.69398
5.3	7.5	0.50	69.04666
17.5	17.5	1.04	310.29891

Gráfico de resíduos



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

```
> library(klaR)
> dados <- read.csv("C:/Users/prisc/Downloads/Pós IAA/Aprendizado de
Maquina/6 - Veiculos - Dados.csv", header=T)
> View(dados)
> dados$a <- NULL
> set.seed(202490)
> cluster.results <- kmodes(dados, 10, iter.max = 10, weighted = FALSE)
Warning message:
In kmodes(dados, 10, iter.max = 10, weighted = FALSE) :
  data has numeric columns with more than 30 different levels!
> cluster.results
```

```
K-modes clustering with 10 clusters of sizes 109, 68, 66, 104, 100, 79,
114, 76, 68, 62
```

Cluster modes:

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect
1	85	43	68	120	54	7	150	46	19
145		169		341	171				

2	89	47	85	162	64	11	157	43	20
160		173		354	186				
3	90	38	75	169	68	6	161	41	20
131		184		388	137				
4	107	54	103	197	62	11	220	31	24
162		229		635	214				
5	104	52	101	197	64	10	208	31	24
161		228		706	218				
6	89	42	66	155	64	7	149	45	19
144		167		327	158				
7	86	37	66	150	59	7	133	50	18
127		159		246	145				
8	86	39	72	140	56	7	151	44	19
134		170		339	139				
9	100	51	104	201	67	10	201	32	23
158		214		307	186				
10	91	46	76	176	63	9	157	40	20
148		181		373	151				
SkewMaxis Skewmaxis Kurtmaxis KurtMaxis HollRa tipo									
1		85		4	14		180	184	bus
2		75		1	9		183	195	van
3		71		1	6		200	204	bus
4		70		0	11		189	199	saab
5		74		0	6		187	197	opel
6		72		1	13		188	201	saab
7		64		1	7		186	183	van
8		80		0	21		183	187	opel
9		72		5	11		192	196	bus
10		67		0	10		192	198	opel

Clustering vector:

```
[1] 5 6 9 10 6 3 3 7 7 4 7 3 3 6 3 9 2 9 5 5 2 7
1 9 9 6 7 4 3 6 2 6 7 5
[35] 10 10 1 8 5 8 9 7 2 10 4 2 7 1 2 10 6 4 5 9 1 7
4 2 4 7 5 8 7 6 9 8 1 5
[69] 1 1 5 5 10 7 8 10 6 8 5 1 8 4 7 8 1 3 6 7 3 7
4 3 5 2 1 5 6 1 10 1 6 6
[103] 7 3 1 4 5 8 1 3 1 2 8 8 1 2 9 6 1 9 7 2 6 2
9 2 6 6 2 9 9 6 6 7 5 2
[137] 2 7 7 4 10 1 5 1 10 10 2 9 6 7 5 1 8 5 1 1 6 7
8 8 7 5 10 7 5 4 6 4 1 7
[171] 4 4 1 5 2 3 8 1 3 7 4 3 1 8 4 9 3 10 5 10 10 4
6 1 10 1 4 8 1 1
```

[reached getOption("max.print") -- omitted 646 entries]

Within cluster simple-matching distance by cluster:

```
[1] 1652 1012 1032 1563 1526 1266 1829 1195 1056 980
```

Available components:

```
[1] "cluster" "size" "modes" "withindiff" "iterations"
"weighted"
```

```
> resultado <- cbind(dados, cluster.results$cluster)
> options(max.print = 200)
> resultado
```

Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong
1	95	48	83	178	72	10	162
2	91	41	84	141	57	9	149
3	104	50	106	209	66	10	207

4	93	41	82	159	63	9	144	46
5	85	44	70	205	103	52	149	45
6	107	57	106	172	50	6	255	26
7	97	43	73	173	65	6	153	42
8	90	43	66	157	65	9	137	48
9	86	34	62	140	61	7	122	54
10	93	44	98	197	62	11	183	36
	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr			
1		20	159	176	379	184		
2		19	143	170	330	158		
3		23	158	223	635	220		
4		19	143	160	309	127		
5		19	144	241	325	188		
6		28	169	280	957	264		
7		19	143	176	361	172		
8		18	146	162	281	164		
9		17	127	141	223	112		
10		22	146	202	505	152		
	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo		
1		70	6	16	187	197	van	
2		72	9	14	189	199	van	
3		73	14	9	188	196	saab	
4		63	6	10	199	207	van	
5		127	9	11	180	183	bus	
6		85	5	9	181	183	bus	
7		66	13	1	200	204	bus	
8		67	3	3	193	202	van	
9		64	2	14	200	208	van	
10		64	4	14	195	204	saab	
	cluster.results\$cluster							
1			5					
2			6					
3			9					
4			10					
5			6					
6			3					
7			3					
8			7					
9			7					
10			4					
[reached 'max' / getOption("max.print") -- omitted 836 rows]								

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

```
> library(arules)
> library(datasets)
>
> dados <- read.transactions("C:/Users/prisc/Downloads/Pós
IAA/Aprendizado de Maquina/2 - Musculacao - Dados.csv", sep = ";",
format = "basket")
Warning message:
In asMethod(object) : removing duplicated items in transactions
>
> inspect(dados[1:5])
```

```

items
[1] {Afundo,Crucifixo,Gemeos,LegPress}
[2] {Agachamento,Gemeos,LegPress}
[3] {Afundo,Agachamento,Gemeos,LegPress}
[4] {Adutor,Agachamento,LegPress}
[5] {Afundo,Bicicleta,Gemeos,LegPress}

```

```

> set.seed(202490)
>
> rules <- apriori(dados, parameter = list(supp = 0.3, conf = 0.75,
target = "rules"))

```

Apriori

Parameter specification:

confidence	minval	smax	arem	aval	originalSupport
0.75	0.1	1	none	FALSE	TRUE
maxtime	support	minlen	maxlen	target	ext
5	0.3	1	10	rules	TRUE

Algorithmic control:

filter	tree	heap	memopt	load	sort	verbose
0.1	TRUE	TRUE	FALSE	TRUE	2	TRUE

Absolute minimum support count: 7

```

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[11 item(s), 26 transaction(s)] done [0.00s].
sorting and recoding items ... [8 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [16 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].

```

```

> inspect(rules)

```

	lhs		rhs	support
	confidence	coverage	lift count	
[1]	{}		=> {LegPress}	0.8076923
	0.8076923	1.0000000	1.000000 21	
[2]	{Agachamento}		=> {LegPress}	0.3076923
	1.0000000	0.3076923	1.238095 8	
[3]	{Afundo}		=> {Gemeos}	0.3461538
	1.0000000	0.3461538	1.529412 9	
[4]	{AgachamentoSmith}		=> {Esteira}	0.3076923
	0.8000000	0.3846154	1.733333 8	
[5]	{AgachamentoSmith}		=> {Extensor}	0.3461538
	0.9000000	0.3846154	1.800000 9	
[6]	{AgachamentoSmith}		=> {Bicicleta}	0.3076923
	0.8000000	0.3846154	1.485714 8	
[7]	{Esteira}		=> {Extensor}	0.4230769
	0.9166667	0.4615385	1.833333 11	
[8]	{Extensor}		=> {Esteira}	0.4230769
	0.8461538	0.5000000	1.833333 11	
[9]	{Esteira}		=> {Bicicleta}	0.3846154
	0.8333333	0.4615385	1.547619 10	
[10]	{Extensor}		=> {Bicicleta}	0.4615385

```

0.9230769 0.5000000 1.714286    12
[11] {Bicicleta}          => {Extensor}  0.4615385
0.8571429 0.5384615 1.714286    12
[12] {AgachamentoSmith,Extensor}=> {Bicicleta} 0.3076923
0.8888889 0.3461538 1.650794     8
[13] {AgachamentoSmith,Bicicleta}=> {Extensor}  0.3076923
1.0000000 0.3076923 2.000000     8
[14] {Esteira,Extensor}      => {Bicicleta} 0.3846154
0.9090909 0.4230769 1.688312    10
[15] {Bicicleta,Esteira}     => {Extensor}
0.3846154 1.0000000 0.3846154 2.000000    10
[16] {Bicicleta,Extensor}    => {Esteira}
0.3846154 0.8333333 0.4615385 1.805556    10

```

APÊNDICE 8 – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B - RESOLUÇÃO

1 - Classificação de Imagens (CNN)

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

```
[ ]: cifar10 = tf.keras.datasets.cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
[ ]: x_train, x_test = x_train/255.0, x_test/255.0

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

print("x_train.shape", x_train.shape)
print("y_train.shape", y_train.shape)
print("x_test.shape", x_test.shape)
print("y_test.shape", y_test.shape)
```

```
x_train.shape (50000, 32, 32, 3)
y_train.shape (50000,)
x_test.shape (10000, 32, 32, 3)
y_test.shape (10000,)
```

```
[ ]: K = len(set(y_train))

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)
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)
```

```
[ ]: model.summary()
```

```
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 162)	0
embedding (Embedding)	(None, 162, 20)	144,520
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6
Total params: 145,046 (566.59 KB)		
Trainable params: 145,046 (566.59 KB)		
Non-trainable params: 0 (0.00 B)		

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

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

Non-trainable params: 0 (0.00 B)

```
[ ]: model.compile(optimizer="adam", loss="sparse_categorical_crossentropy",
    ↪metrics=["accuracy"])
r = model.fit(x_train, y_train, validation_data=(x_text, y_test),
    ↪epochs=15)
```

Epoch 1/15

1563/1563 [CONCLUIDO] 16s 7ms/step -

accuracy: 0.3566 - loss: 1.7506 - val_accuracy: 0.5311 - val_loss: 1.3112

...

Epoch 1/15

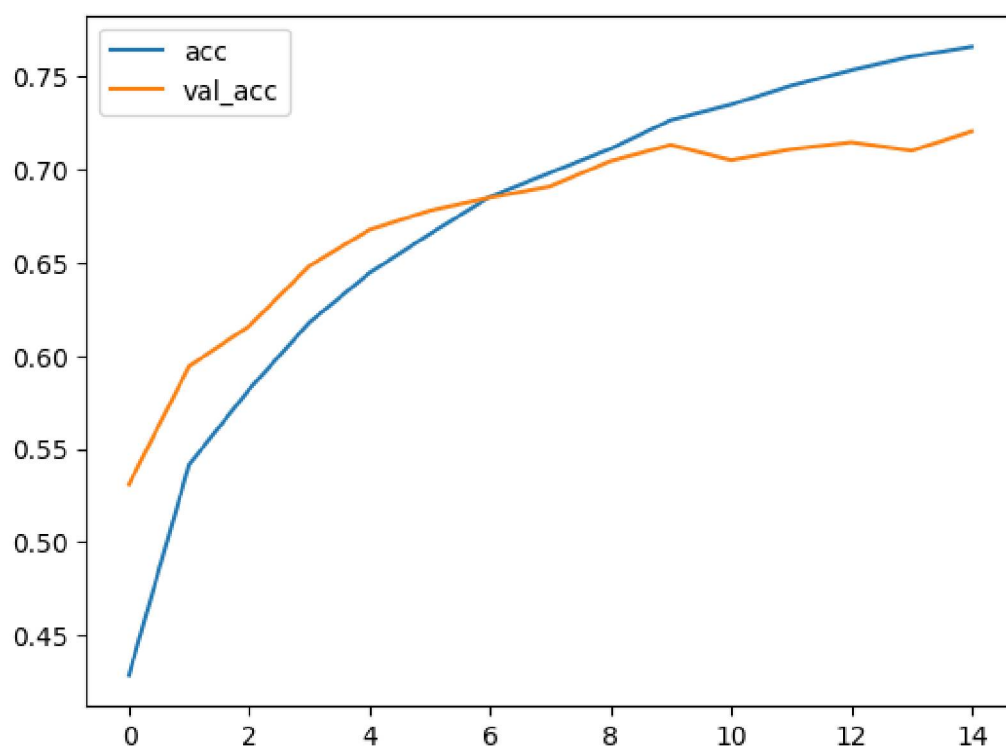
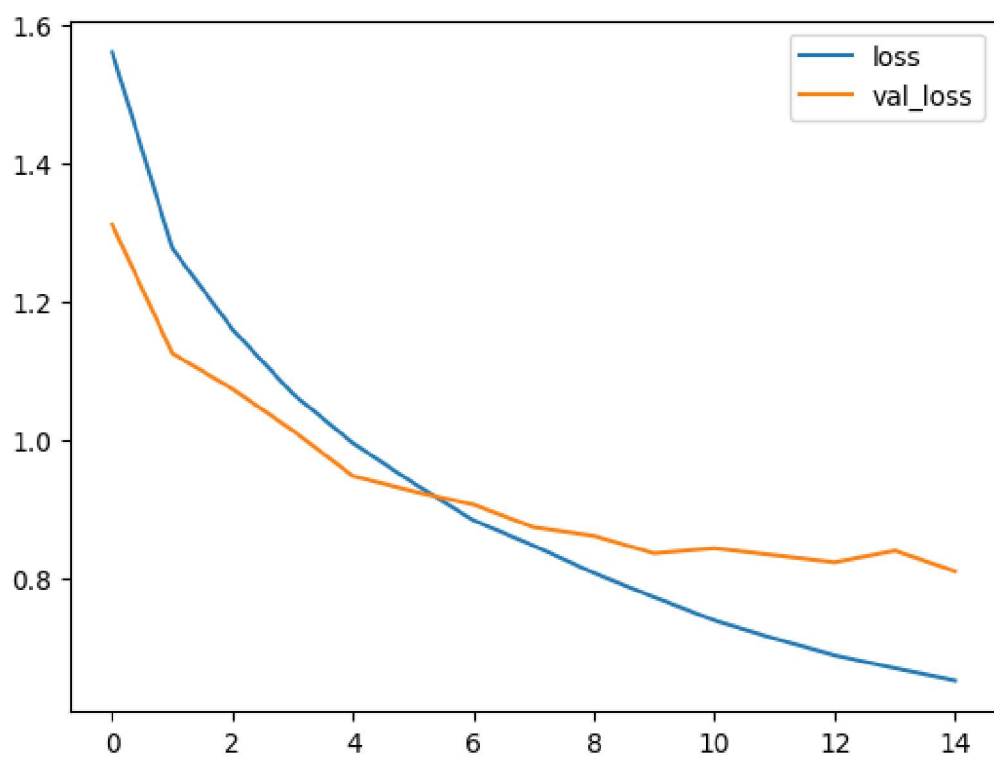
1563/1563 [CONCLUIDO] 16s 7ms/step -

accuracy: 0.3566 - loss: 1.7506 - val_accuracy: 0.5311 - val_loss: 1.3112

...

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

plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
plt.show()
```

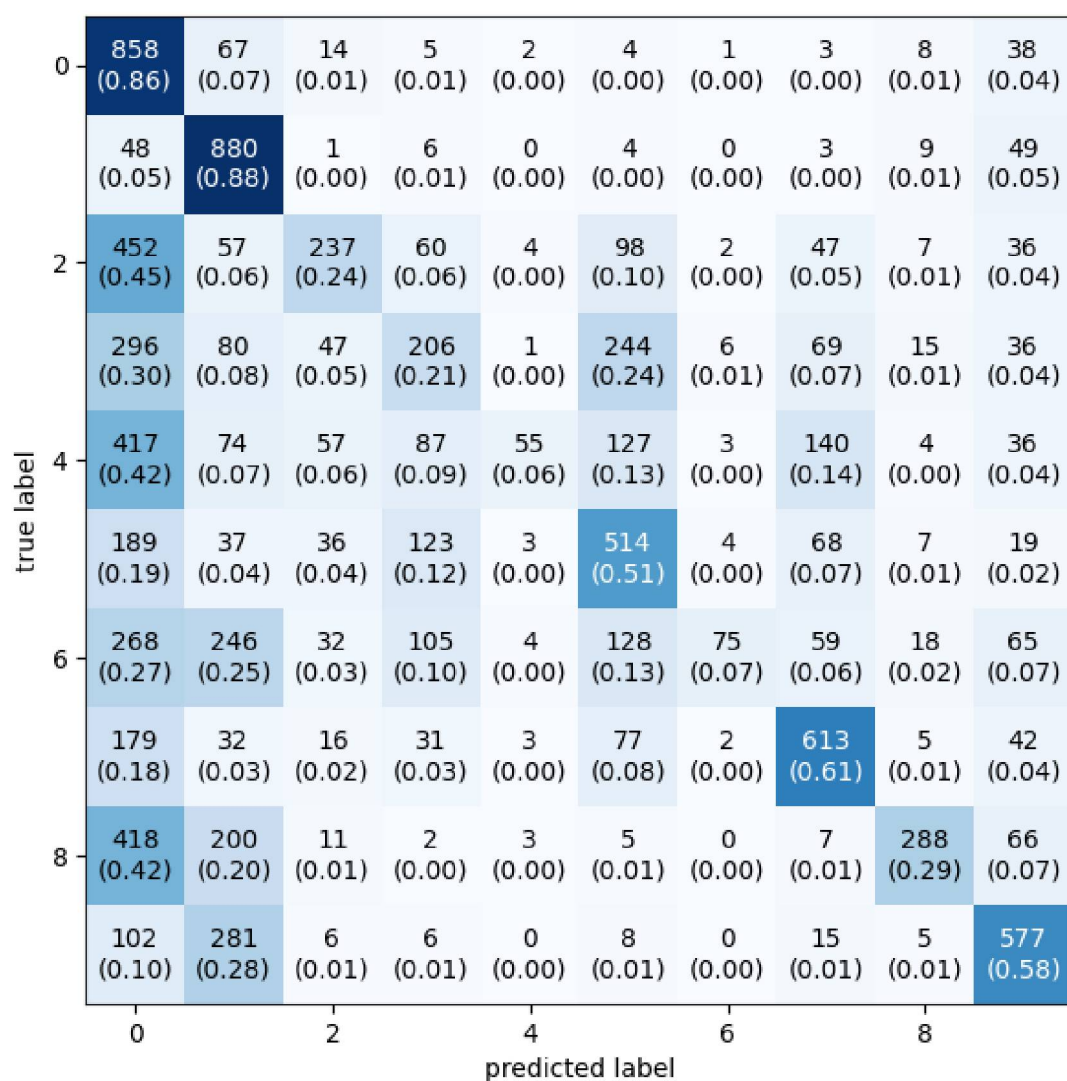



```
[ ]: y_pred = model.predict(x_test).argmax(axis=1)
```

```
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7,7), show_normed=True)
```

313/313 ----- 1s 2ms/step

```
[ ]: (<Figure size 700x700 with 1 Axes>,
      <Axes: xlabel='predicted label', ylabel='true label'>)
```



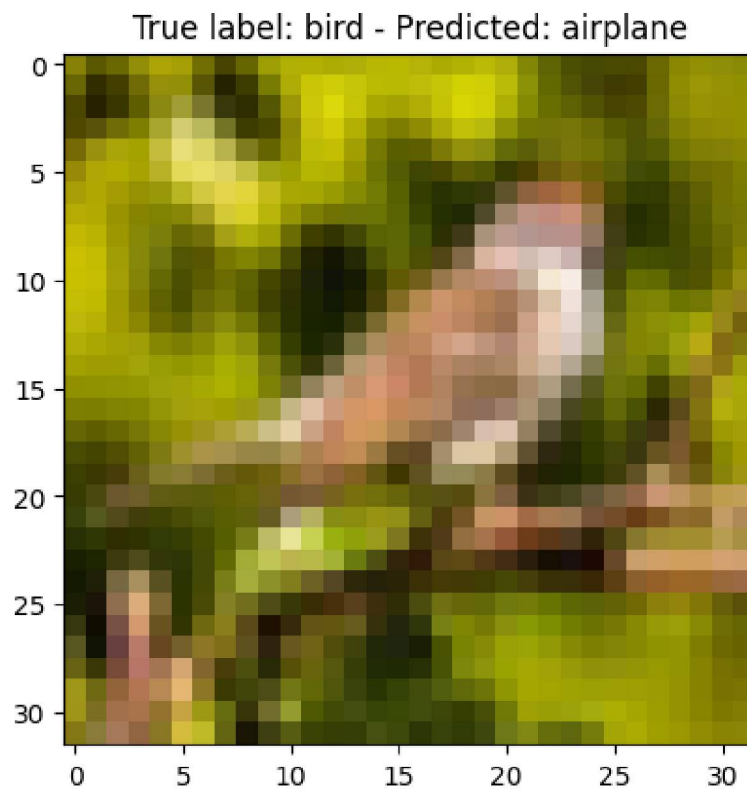
```
[ ]: labels = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

misclassified = np.where(y_pred!=y_test)[0]
```

```
i = np.random.choice(misclassified)

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

```
[ ]: Text(0.5, 1.0, 'True label: bird - Predicted: airplane')
```



2 - Detector de SPAM (RNN)

```
[ ]: import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.preprocessing.text import Tokenizer
```

```
[ ]: !wget http://www.razer.net.br/datasets/spam.csv

df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()

df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ['label', 'data']

df["b_label"] = df["label"].map({"ham":0, "spam":1})
y = df["b_label"].values

--2024-08-11 18:27:49-- http://www.razer.net.br/datasets/spam.csv
Resolving www.razer.net.br (www.razer.net.br)... 178.128.150.229
Connecting to www.razer.net.br (www.razer.net.br)|178.128.150.229|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 503663 (492K) [text/csv]
Saving to: 'spam.csv'

spam.csv          100%[=====>] 491.86K  --.-KB/s    in 0.1s

2024-08-11 18:27:50 (3.50 MB/s) - 'spam.csv' saved [503663/503663]
```

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

```
[ ]: num_words = 20000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train)
```

```

sequences_test = tokenizer.texts_to_sequences(x_test)
word2index = tokenizer.word_index
V = len(word2index)
print(f"{V}s tokens")

```

7225s tokens

```

[ ]: data_train = pad_sequences(sequences_train)
      T = data_train.shape[1]
      print(f"Tamanho da sequencia treino: {T}")

      data_test = pad_sequences(sequences_test,maxlen= T)
      U = data_test.shape[1]
      print(f"Tamanho da sequencia teste: {U}")

```

Tamanho da sequencia treino: 162

Tamanho da sequencia teste: 162

```

[ ]: print(data_train.shape)
      print(data_test.shape)

```

(3733, 162)

(1839, 162)

```

[ ]: # Tamanho do Embedding
      D = 20
      #Quantidade de LSTM
      M = 5
      #Camada de entrada
      i = Input(shape=(T,))
      #Criando camada de embedding, essa camada vai ser treinada tambem
      x = Embedding(V + 1, D)(i)
      # Camada LSTM
      x = LSTM(M)(x)
      #Camada Densa
      x = Dense(1, activation="sigmoid")(x)
      # Modelo
      model = Model(i, x)

```

```

[ ]: model.summary()

```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 162)	0
embedding (Embedding)	(None, 162, 20)	144,520
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6

Total params: 145,046 (566.59 KB)

Trainable params: 145,046 (566.59 KB)

Non-trainable params: 0 (0.00 B)

```
[ ]: 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))
```

Epoch 1/5

117/117 ----- 11s 71ms/step -

accuracy: 0.8341 - loss: 0.5878 - val_accuracy: 0.9233 - val_loss: 0.3001

Epoch 2/5

117/117 ----- 8s 69ms/step -

accuracy: 0.9608 - loss: 0.2497 - val_accuracy: 0.9668 - val_loss: 0.1772

Epoch 3/5

117/117 ----- 6s 55ms/step -

accuracy: 0.9843 - loss: 0.1439 - val_accuracy: 0.9821 - val_loss: 0.1188

Epoch 4/5

117/117 ----- 9s 76ms/step -

accuracy: 0.9904 - loss: 0.0958 - val_accuracy: 0.9859 - val_loss: 0.0924

Epoch 5/5

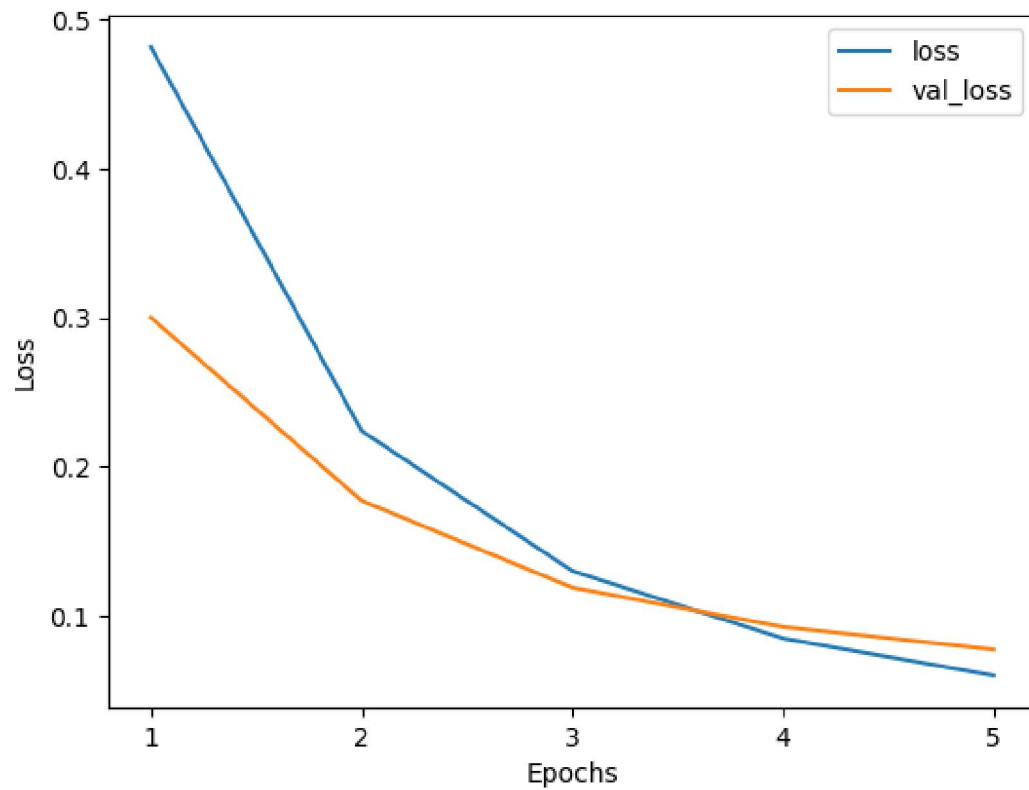
117/117 ----- 8s 56ms/step -

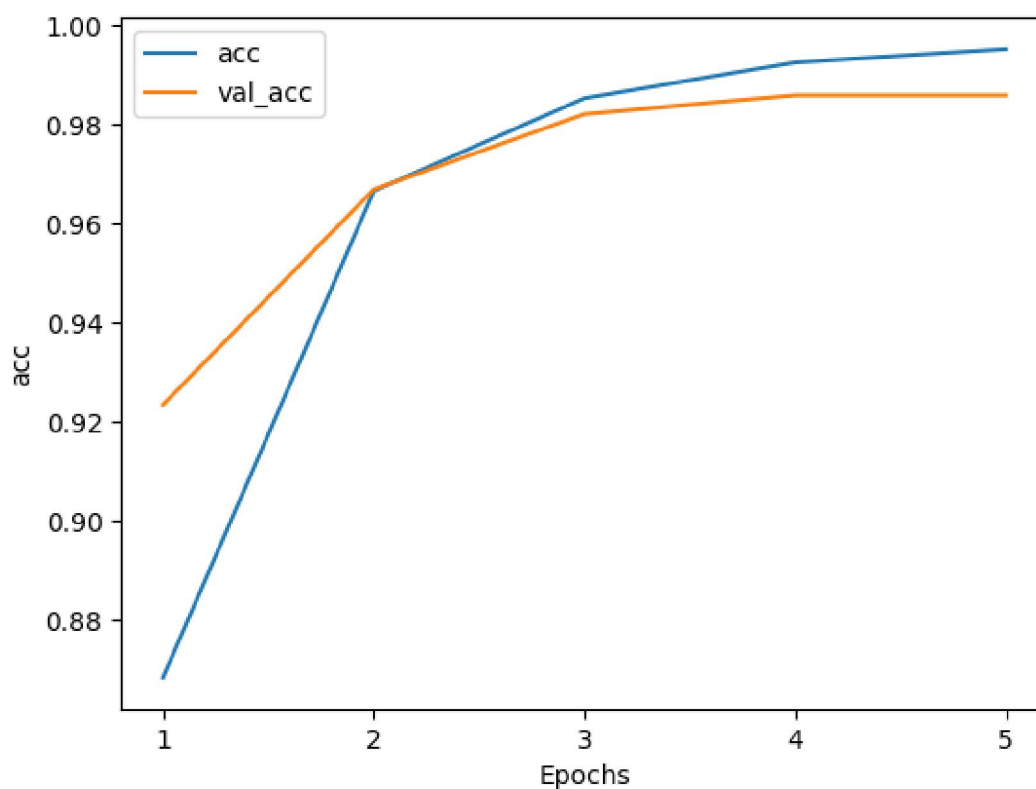
accuracy: 0.9965 - loss: 0.0619 - val_accuracy: 0.9859 - val_loss: 0.0773

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

plt.plot(r.history['accuracy'], label='acc')
```

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





```
[ ]: texto = "Is your car dirty? Discover our new product. Free for all. Click_
    ↳the link."
seq_texto = tokenizer.texts_to_sequences([texto])
data_texto = pad_sequences(seq_texto, maxlen= T)

pred = model.predict(data_texto)
print(pred)
print("SPAM" if pred >=0.5 else "OK")
```

1/1 ----- 0s 178ms/step

[[0.5046611]]

SPAM

3 - Gerador de Dígitos Fake (GAN)

```
[ ]: !pip install imageio
```

```
Requirement already satisfied: imageio in /usr/local/lib/python3.10/
  ↳ dist-packages (2.34.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/
  ↳ dist-packages (from imageio) (1.26.4)
Requirement already satisfied: pillow>=8.3.2 in /usr/local/lib/python3.10/
  ↳ dist-packages (from imageio) (9.4.0)
```

```
[ ]: !pip install git+https://github.com/tensorflow/docs
```

```
Collecting git+https://github.com/tensorflow/docs
  Cloning https://github.com/tensorflow/docs to /tmp/pip-req-build-i8_uu6q4
  Running command git clone --filter=blob:none --quiet https://github.com/
  ↳ tensorflow/docs /tmp/pip-req-build-i8_uu6q4
  Resolved https://github.com/tensorflow/docs to commit
  ↳ 7d4187e1624afaf0639de9ca5b7ccd5ad19e3
  Preparing metadata (setup.py) ... done
Collecting astor (from tensorflow-docs==2024.7.15.51478)
  Downloading astor-0.8.1-py2.py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/
  ↳ dist-packages (from tensorflow-docs==2024.7.15.51478) (1.4.0)
...
Successfully installed astor-0.8.1 tensorflow-docs-2024.7.15.51478
```

```
[ ]: 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
```

```
[ ]: (train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()

train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).
  ↳ astype('float32')

train_images = (train_images - 127.5) / 127.5
```

```

Downloading data from https://storage.googleapis.com/tensorflow/
  ↳tf-keras-datasets/mnist.npz
11490434/11490434 ----- 2s
0us/step

```

```

[ ]: BUFFER_SIZE = 60000
    BATCH_SIZE = 256

```

```

[ ]: train_dataset = tf.data.Dataset.from_tensor_slices(train_images).
  ↳shuffle(BUFFER_SIZE).batch(BATCH_SIZE)

```

```

[ ]: def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)

    model.add(layers.Conv2DTranspose(128,(5,5), strides=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

```

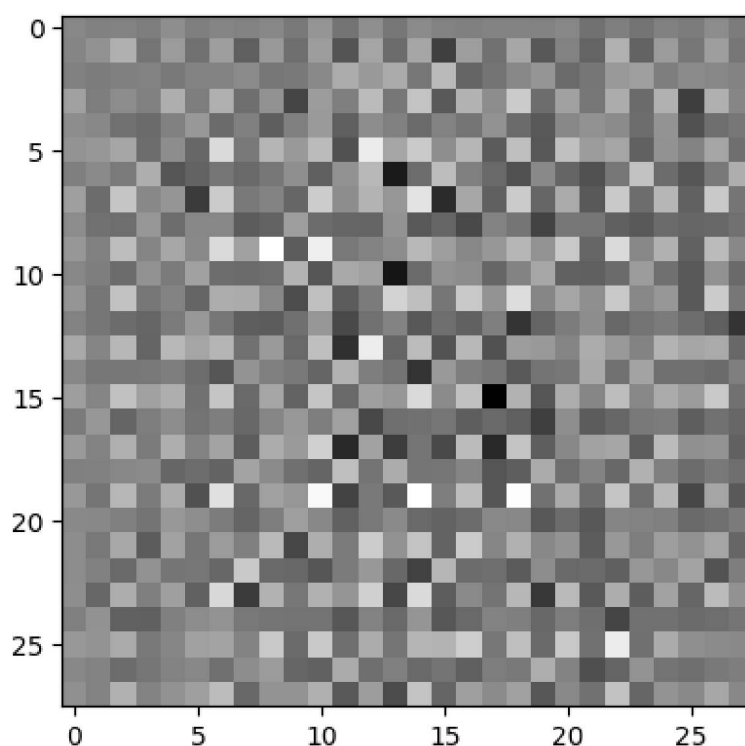
```

[ ]: generator = make_generator_model()

noise = tf.random.normal([1,100])
generated_image = generator(noise, training=False)
plt.imshow(generated_image[0,:,:,:0], cmap='gray')

```

```
[ ]: <matplotlib.image.AxesImage at 0x7b549c4122f0>
```



```
[ ]: 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
```

```
[ ]: discriminator = make_discriminator_model()
decision = discriminator(generated_image)
```

```
print(decision)
```

```
tf.Tensor([[0.00391983]], shape=(1, 1), dtype=float32)
```

```
[ ]: cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_output = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_output
    return total_loss
```

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

```
[ ]: generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
```

```
[ ]: 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)
```

```
[ ]: EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16

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

```
[ ]: @tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

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

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

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

```

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

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

```

```

[ ]: 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)

```

```

[ ]: 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()

```



```

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

```

Cria rede feed-forward pontual

```
[ ]: 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)
    ])
```

Camada do Codificador

```
[ ]: 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
```

Camada do Decodificador

```
[ ]: 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
```


Encoder Completo

```
[ ]: 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)
```

Decoder Completo

```
[ ]: 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

```

Transformer Completo

```

[ ]: 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

```
[ ]: # 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')
```

0.1 Treinamento

```
[ ]: 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
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:
            if batch == 0:
                print(f'Epoch {epoch+1} Batch {batch} Loss {train_loss.result():.4f} Accuracy {train_accuracy.result():.4f}')
            ↪ elif batch == 50:
                print(f'Epoch {epoch+1} Batch {batch} Loss {train_loss.result():.4f} Accuracy {train_accuracy.result():.4f}')
            ↪ else:
                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')

```

```

Epoch 1 Batch    0 Loss 8.8823 Accuracy 0.0000
Epoch 1 Batch   50 Loss 8.8116 Accuracy 0.0127
Epoch 1 Batch  100 Loss 8.7075 Accuracy 0.0298
Epoch 1 Batch  150 Loss 8.5943 Accuracy 0.0361
Epoch 1 Batch  200 Loss 8.4554 Accuracy 0.0393
Epoch 1 Batch  250 Loss 8.2874 Accuracy 0.0421
Epoch 1 Batch  300 Loss 8.0976 Accuracy 0.0449
Epoch 1 Batch  350 Loss 7.8994 Accuracy 0.0510
Epoch 1 Batch  400 Loss 7.7098 Accuracy 0.0591
Epoch 1 Batch  450 Loss 7.5438 Accuracy 0.0657
Epoch 1 Batch  500 Loss 7.3972 Accuracy 0.0727
Epoch 1 Batch  550 Loss 7.2665 Accuracy 0.0794
Epoch 1 Batch  600 Loss 7.1445 Accuracy 0.0869
Epoch 1 Batch  650 Loss 7.0294 Accuracy 0.0945
Epoch 1 Batch  700 Loss 6.9226 Accuracy 0.1014
Epoch 1 Batch  750 Loss 6.8224 Accuracy 0.1078
Epoch 1 Batch  800 Loss 6.7284 Accuracy 0.1139
Epoch 1 Loss 6.7121 Accuracy 0.1150
Time taken for 1 epoch: 172.86 secs

```

```

Epoch 2 Batch    0 Loss 5.3843 Accuracy 0.1926
Epoch 2 Batch   50 Loss 5.2308 Accuracy 0.2108
Epoch 2 Batch  100 Loss 5.1956 Accuracy 0.2149
Epoch 2 Batch  150 Loss 5.1750 Accuracy 0.2175
Epoch 2 Batch  200 Loss 5.1523 Accuracy 0.2207
Epoch 2 Batch  250 Loss 5.1326 Accuracy 0.2227
Epoch 2 Batch  300 Loss 5.1051 Accuracy 0.2255
Epoch 2 Batch  350 Loss 5.0805 Accuracy 0.2282
Epoch 2 Batch  400 Loss 5.0533 Accuracy 0.2310
Epoch 2 Batch  450 Loss 5.0310 Accuracy 0.2332
Epoch 2 Batch  500 Loss 5.0114 Accuracy 0.2350
Epoch 2 Batch  550 Loss 4.9896 Accuracy 0.2372
Epoch 2 Batch  600 Loss 4.9686 Accuracy 0.2393
Epoch 2 Batch  650 Loss 4.9482 Accuracy 0.2412
Epoch 2 Batch  700 Loss 4.9331 Accuracy 0.2424
Epoch 2 Batch  750 Loss 4.9174 Accuracy 0.2437

```

```
Epoch 20 Batch 650 Loss 1.4316 Accuracy 0.6828
Epoch 20 Batch 700 Loss 1.4326 Accuracy 0.6827
Epoch 20 Batch 750 Loss 1.4372 Accuracy 0.6819
Epoch 20 Batch 800 Loss 1.4396 Accuracy 0.6816
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-4
Epoch 20 Loss 1.4401 Accuracy 0.6815
Time taken for 1 epoch: 97.21 secs
```

Tradutor

```
[ ]: 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}')

```

Prediction : b'i read about tattoos in enclopedia .'

APÊNDICE 9 – BIG DATA

A – ENUNCIADO

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

B - RESOLUÇÃO

Resumo da Implementação de Big Data no Setor Financeiro

Neste projeto, realizamos a migração de uma vasta quantidade de dados sigilosos (valor) de uma empresa do setor financeiro, anteriormente armazenados na AWS, para um ambiente on-premise, utilizando a plataforma Hortonworks com HBase e Hive, como fonte de dados. O objetivo principal era aumentar a segurança e reduzir custos, sem comprometer a performance e a integridade dos dados. O projeto envolveu a migração de dados estruturados, semiestruturados e não estruturados (Variedade), exigindo um planejamento cuidadoso para garantir a integridade, veracidade e disponibilidade dos dados durante e após a migração (Veracidade).

A empresa mantinha seus dados em um ambiente de nuvem na AWS, utilizando o S3 para armazenamento e o DynamoDB para gerenciar dados relacionais e não relacionais. No entanto, devido à natureza confidencial dos dados financeiros, foi necessário migrá-los para um ambiente on-premise, onde o controle e a segurança poderiam ser aprimorados. A escolha da plataforma Hortonworks, com HBase e Hive, foi feita devido à sua capacidade de lidar com grandes volumes de dados (Volume), variedade de formatos e a necessidade de manter a integridade dos dados.

A migração envolveu a transferência de dados de várias naturezas:

- **Dados Estruturados:** Informações organizadas em formatos tabulares, com relações bem definidas entre as entidades.
- **Dados Semiestruturados:** Incluindo arquivos JSON e XML, que, embora possuam alguma estrutura, não são tão rigidamente organizados como os dados relacionais.

- **Dados Não Estruturados:** Arquivos de texto, documentos e outras informações que não seguem uma estrutura pré-definida.

A complexidade da migração foi aumentada pela necessidade de adaptar modelos de dados relacionais para o HBase e Hive, que são mais orientados a colunas e não impõem restrições rígidas quanto à forma dos dados. Além disso, a volumetria foi um desafio significativo, uma vez que o banco de dados DynamoDB já continha uma vasta quantidade de registros, muitos dos quais eram dados sensíveis de clientes.

Os principais desafios enfrentados incluíram:

1. **Segurança e Confidencialidade dos Dados:** A migração de dados sigilosos exigiu criptografia de ponta a ponta e verificações rigorosas de integridade para garantir que os dados não fossem corrompidos ou expostos durante o processo.
2. **Adaptação dos Modelos Relacionais:** A conversão de modelos relacionais para o HBase e Hive exigiu revisões de design e ajustes para garantir que a performance fosse mantida e que as consultas pudessem ser executadas eficientemente.
3. **Gerenciamento de Volume de Dados:** A volumetria foi um fator crítico, exigindo otimização dos processos de ingestão e indexação para garantir que o ambiente Hortonworks pudesse lidar com a carga sem comprometer a performance.
4. **Veracidade e Confiabilidade:** A precisão e a integridade dos dados eram fundamentais, especialmente em um ambiente financeiro. A implementação incluiu mecanismos de verificação e validação para garantir que os dados migrados fossem consistentes e confiáveis.
5. **Redução de Custos:** A migração para um ambiente on-premise resultou em economia de custos significativa, uma vez que o gerenciamento local de grandes volumes de dados era mais econômico a longo prazo do que manter esses dados na nuvem.
6. **Maior Segurança:** Com os dados mantidos em um ambiente controlado, a empresa pôde implementar medidas de segurança mais rigorosas, garantindo que os dados confidenciais dos clientes fossem protegidos contra acessos não autorizados.
7. **Manutenção da Performance:** Embora a velocidade de acesso aos dados não tenha sido drasticamente impactada pela migração, a implementação de um ambiente otimizado garantiu que a performance fosse mantida em níveis aceitáveis, mesmo com a complexidade e variedade dos dados (Velocidade).
8. **Flexibilidade no Gerenciamento de Dados:** A adoção do HBase e Hive permitiu à empresa lidar com diferentes tipos de dados de maneira eficiente, oferecendo flexibilidade para atender às necessidades do negócio.

A implementação de Big Data no ambiente on-premise da empresa do setor financeiro foi um sucesso, atendendo às exigências de segurança, redução de custos e integridade dos dados. A migração para a plataforma Hortonworks, com HBase e Hive, proporcionou à empresa uma solução robusta e escalável, capaz de gerenciar grandes volumes de dados variados, mantendo a performance e confiabilidade necessárias para suas operações.

SoR, SoT e Spec

Em um projeto de outro setor financeiro, os dados armazenados on-premise estão sendo migrados para a AWS, um processo denominado “democratização dos dados”. Essa democratização tem como objetivo criar um data lake para facilitar o compartilhamento de dados entre diversos setores da empresa.

O processo de democratização envolve 3 camadas:

1. **SoR – Primeira Camada da Democratização dos Dados:** Nesta camada, as informações são idênticas aos dados da tabela on-premise, sem nenhuma adição, alteração ou remoção. Esta camada não pode ser compartilhada devido a possibilidade de conter dados sensíveis ou sigilosos.
2. **SoT – Segunda Camada da Democratização dos Dados:** Nesta camada, utilizam-se as informações da camada SoR, resultando em dados mais “polidos”. Alterações são feitas, se necessário, para tornar os dados mais “amigáveis”, facilitando no desenvolvimento da Spec e evitando retrabalho. Esta camada não pode ser compartilhada devido a possibilidade de conter dados sensíveis ou sigilosos.
3. **Spec – Terceira Camada da Democratização dos Dados:** Nesta camada, os dados são preparados especificamente para atender às necessidades do usuário final. Na empresa em questão, a área responsável solicita ao dono do dado uma tabela contendo as informações necessárias.

APÊNDICE 10 – VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B - RESOLUÇÃO

```
[ ]: # Libraries
import numpy as np, cv2, sys, pylab, matplotlib, PIL, scipy
from numpy import histogram
from scipy import stats
from PIL import Image
import matplotlib.pyplot as plt
import skimage
from skimage.feature import local_binary_pattern
from skimage import data, morphology

from skimage.transform import rotate
from skimage import data
from skimage.color import label2rgb
from skimage import data, exposure, util

from skimage.util import dtype

import os
import random
```

Q1 Extração de Características

Q1T1. Carregue a base de dados de Treino

```
[ ]: from google.colab import files,drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
[ ]: # Leitura e extração Zip - Dados de Treinamento
from google.colab import files

!unzip Train_Warwick.zip
```

unzip: cannot find or open Train_Warwick.zip, Train_Warwick.zip.zip or Train_Warwick.zip.ZIP.

Q1T2. Crie partições contendo 80% para treino e 20% para validação (atenção aos pacientes).

```
[ ]: # Set the path to your data
data_train = '/content/drive/MyDrive/Colab Notebooks/Visão Computacional/
↳Train_4cls_amostra/Pasta Treino'
data_val = '/content/drive/MyDrive/Colab Notebooks/Visão Computacional/
↳Train_4cls_amostra/Pasta Validacao'
```

```
[ ]: def gravar_lista_imagens(data_path):
    # Lista para armazenar os caminhos das imagens

    all_files = []

    for class_folder in os.listdir(data_path):
        class_path = os.path.join(data_path, class_folder)
        if os.path.isdir(class_path):
            for img_file in os.listdir(class_path):
                if img_file.endswith('.png'):
                    file_path = os.path.join(class_path, img_file)
                    all_files.append(file_path)

    return all_files

train_files = gravar_lista_imagens(data_train)
valid_files = gravar_lista_imagens(data_val )
```

```
[ ]: print(f"Total train images: {len(train_files)}")
print(f"Total validation images: {len(valid_files)}")
```

Total train images: 473

Total validation images: 120

```
[ ]: # Example: print the first few file paths in train and test sets
print("Train files:", train_files[:5])
print("Validation files:", valid_files[:5])
```

Train files: ['/content/drive/MyDrive/Colab Notebooks/Visão Computacional/Train_4cls_amostra/Pasta Treino/0/13_HER2_11272.png',
'/content/drive/MyDrive/Colab Notebooks/Visão Computacional/Train_4cls_amostra/Pasta Treino/0/13_HER2_10874.png',
'/content/drive/MyDrive/Colab Notebooks/Visão

```

Computacional/Train_4cls_amostra/Pasta Treino/0/13_HER2_10494.png',
'/content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Train_4cls_amostra/Pasta Treino/0/13_HER2_10366.png',
'/content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Train_4cls_amostra/Pasta Treino/0/01_HER2_7144.png']
Validation files: ['/content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Train_4cls_amostra/Pasta Validacao/0/57_HER2_4420.png',
'/content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Train_4cls_amostra/Pasta Validacao/0/57_HER2_4666.png',
'/content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Train_4cls_amostra/Pasta Validacao/0/57_HER2_4898.png',
'/content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Train_4cls_amostra/Pasta Validacao/0/57_HER2_6349.png',
'/content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Train_4cls_amostra/Pasta Validacao/0/57_HER2_4660.png']

```

```

[ ]: # Testing if the split into train and validation worked
def check_patient_overlap(train_list, valid_list):
    # Extract the patient IDs from the train and test lists
    train_patients = set([os.path.basename(file).split('_')[0] for file_
↪in train_list])
    valid_patients = set([os.path.basename(file).split('_')[0] for file_
↪in valid_list])

    # Find the common patient IDs in both lists
    overlap = train_patients.intersection(valid_patients)

    # Check if there's any overlap
    if overlap:
        return f"Patients in both train and validation sets:
↪{sorted(list(overlap))}"
    else:
        return "No patients are present in both train and validation sets.
↪"

result = check_patient_overlap(train_files, valid_files)
print(result)

```

```

Patients in both train and validation sets: ['01', '04', '11', '14', '15',
↪'16',
'19', '25', '26', '36', '46', '57']

```


Q1T3.1 Extraia características utilizando LBP (gerando um csv).

```
[ ]: ## Extracting LBP characteristics for train and validation and saving to
      →file
import os
import cv2
import numpy as np
from skimage.feature import local_binary_pattern
from skimage import morphology
import csv

# Funções de extração adaptadas
def lbp_riu(npmat_img, label):
    radius = 1
    n_points = 8
    lbp = local_binary_pattern(npmat_img, n_points, radius,
    →method='uniform')

    array_lbp = np.zeros(11, dtype=int) # Dez posições do P+2 e mais uma
    →para o label
    rows, cols = npmat_img.shape
    for ci in range(rows):
        for cj in range(cols):
            aux = int(lbp[ci][cj])
            array_lbp[aux + 1] += 1

    array_lbp[0] = label # Label da imagem (agora um número)
    return array_lbp # Retorna os vetores extraídos

def lbp_tophat(npmat_img, label):
    radius = 1
    n_points = 8
    lbp = local_binary_pattern(npmat_img, n_points, radius,
    →method='uniform')

    array_lbp = np.zeros(11, dtype=int) # Dez posições do P+2 e mais uma
    →para o label
    rows, cols = npmat_img.shape
    for ci in range(rows):
        for cj in range(cols):
            aux = int(lbp[ci][cj])
            array_lbp[aux + 1] += 1

    se = morphology.disk(5)
```

```

white = morphology.white_tophat(npmat_img, se)
black = morphology.black_tophat(npmat_img, se)
wmean = np.mean(white)
wsd = np.std(white)
bmean = np.mean(black)
bsd = np.std(black)

top_hat_features = np.array([wmean, wsd, bmean, bsd])
# return np.concatenate((array_lbp, top_hat_features)) # Combina as
↳ características LBP e tophat
return array_lbp, top_hat_features

def tophat(npmat_img, label):
    se = morphology.disk(5)
    white = morphology.white_tophat(npmat_img, se)
    black = morphology.black_tophat(npmat_img, se)
    wmean = np.mean(white)
    wsd = np.std(white)
    bmean = np.mean(black)
    bsd = np.std(black)

    top_hat_features = np.array([label, wmean, wsd, bmean, bsd])
    return top_hat_features # Retorna as características extraídas

# Função para calcular características e salvar em CSV
def Calcula_caracteristicas(file_list, csv_name):
    with open(csv_name, mode='w', newline='') as csv_file:
        csv_writer = csv.writer(csv_file)

        header = ['Nome do Arquivo', 'Label'] + [f'LBP_{i}' for i in
↳ range(1, 11)] + \
                [f'TopHat_wmean', f'TopHat_wsd', f'TopHat_bmean',
↳ f'TopHat_bsd'] + \
                [f'LBP_TopHat_{i}' for i in range(1, 11)] + \
                [f'TopHat_wmean2', f'TopHat_wsd2', f'TopHat_bmean2',
↳ f'TopHat_bsd2']
        csv_writer.writerow(header)

        for image_path in file_list:
            file_name = os.path.basename(image_path)
            class_label = int(os.path.basename(os.path.
↳ dirname(image_path)))
            img = cv2.imread(image_path, 0)

```

```

        if img is not None:
            lbp_features = lbp_riu(img, class_label)
            lbp_tophat_features, tophat_features1 = lbp_tophat(img,
↳class_label)
            tophat_features2 = tophat(img, class_label)

            # Combine all features
            combined_features = np.concatenate(([file_name], #
↳Include filename here

                                                    lbp_features,
                                                    tophat_features1,
                                                    lbp_tophat_features[1:],
↳# Exclude the label
                                                    tophat_features2[1:] #
↳Exclude the label
                                                    ))

            csv_writer.writerow(combined_features)

```

```

[ ]: # Chamando para os datasets de treino e teste
Calcula_caracteristicas(train_files, 'train_lbp_features.csv')
Calcula_caracteristicas(valid_files, 'valid_lbp_features.csv')

print("Extração de características LBP e TopHat concluída e salva nos
↳arquivos CSV.")

```

Extração de características LBP e TopHat concluída e salva nos arquivos CSV.

Q1T3.2 Extraia características utilizando CNN VGG16 (gerando um csv).

```

[ ]: !pip install keras

```

```

Requirement already satisfied: keras in /usr/local/lib/python3.11/
↳dist-packages
(3.8.0)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-
packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/
↳dist-packages
(from keras) (2.0.2)
Requirement already satisfied: rich in /usr/local/lib/python3.11/
↳dist-packages
(from keras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/
↳dist-packages

```

```

3    0.000000  0.0  0.202915  0.000000  0.730596  0.0  0.0  0.000000  0.0
4    0.669177  0.0  0.092328  0.000000  0.000000  0.0  0.0  0.000000  0.0
..
115  0.502435  0.0  0.000000  0.115867  0.000000  0.0  0.0  0.332782  0.0
116  0.340461  0.0  0.311162  0.420919  0.000000  0.0  0.0  0.000000  0.0
117  0.354492  0.0  0.937881  0.407781  0.337284  0.0  0.0  0.000000  0.0
118  0.981852  0.0  0.726162  0.703347  0.307817  0.0  0.0  0.204648  0.0
119  0.014179  0.0  0.139641  0.226677  0.000000  0.0  0.0  0.369366  0.0

```

```

          9  ...      25080      25081      25082      25083  25084      25085  ↵
↪ \
0    0.000000  ...  0.451090  0.114031  0.000000  0.000000      0.0  0.00000
1    0.000000  ...  0.706679  0.000000  0.191406  0.447078      0.0  0.00000
2    0.000000  ...  0.045687  0.000000  0.215100  0.000000      0.0  0.00000
3    0.000000  ...  0.598636  0.000000  0.256484  0.454856      0.0  0.00000
4    0.000000  ...  0.785919  0.000000  0.130940  0.000000      0.0  0.00000
..
115  0.000000  ...  0.603611  0.000000  0.000000  0.000000      0.0  0.00000
116  0.000000  ...  0.755744  0.198884  0.000000  0.000000      0.0  0.00974
117  0.159063  ...  0.284996  0.000000  0.000000  0.000000      0.0  0.00000
118  0.000000  ...  0.798354  0.000000  0.502949  0.000000      0.0  0.00000
119  0.000000  ...  0.052298  0.000000  0.830387  0.000000      0.0  0.00000

```

```

          25086      25087  Label  paciente
0    1.286761  0.000000      0      57
1    0.889122  0.000000      0      57
2    0.996628  0.000000      0      57
3    1.160086  0.000000      0      57
4    0.802790  0.000000      0      57
..
115  0.084894  0.000000      3      11
116  0.592438  0.000000      3      19
117  0.000000  0.006357      3      11
118  0.669602  0.000000      3      19
119  0.223745  0.000000      3      11

```

[120 rows x 25090 columns]

Q1T4. Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos.

[]:


```

# ler os arquivos train_lbp_features.csv e valid_lbp_features.csv,
↳ transformar a coluna label para inteiro, ajustar modelos RF, RNA e
↳ SVM nos dados de treinamento e testar os ajustes nos dados de
↳ validação, mostrar matriz de confusão colorida para a performance de
↳ cada modelo nos dados de teste, salvar e mostrar um df com acurácia,
↳ sensibilidade, especificidade, F1Score, ordenar por maior F1Score

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score,
↳ recall_score, f1_score, precision_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler # Import StandardScaler

# Load the CSV files
train_df_lbp = pd.read_csv('train_lbp_features.csv')
valid_df_lbp = pd.read_csv('valid_lbp_features.csv')

# Convert 'Label' column to integer type
train_df_lbp['Label'] = train_df_lbp['Label'].astype(int)
valid_df_lbp['Label'] = valid_df_lbp['Label'].astype(int)

# Separate features and labels for train and test sets
X_train_lbp = train_df_lbp.drop(columns=['Nome do Arquivo', 'Label'])
y_train_lbp = train_df_lbp['Label']
X_valid_lbp = valid_df_lbp.drop(columns=['Nome do Arquivo', 'Label'])
y_valid_lbp = valid_df_lbp['Label']

# Normalize LBP features
scaler = StandardScaler()
X_train_lbp_scaled = scaler.fit_transform(X_train_lbp)
X_valid_lbp_scaled = scaler.transform(X_valid_lbp)

# Initialize models
rf_model_lbp = RandomForestClassifier()
rna_model_lbp = MLPClassifier()
svm_model_lbp = SVC()

# Train the models
rf_model_lbp.fit(X_train_lbp, y_train_lbp)

```

```

rna_model_lbp.fit(X_train_lbp, y_train_lbp)
svm_model_lbp.fit(X_train_lbp, y_train_lbp)

# Make predictions on the valid set
rf_pred_lbp = rf_model_lbp.predict(X_valid_lbp)
rna_pred_lbp = rna_model_lbp.predict(X_valid_lbp)
svm_pred_lbp = svm_model_lbp.predict(X_valid_lbp)

# Function to calculate metrics and create confusion matrix
def evaluate_model(y_true, y_pred_lbp, model_name):
    cm = confusion_matrix(y_true, y_pred_lbp)
    accuracy = accuracy_score(y_true, y_pred_lbp)
    tpr = recall_score(y_true, y_pred_lbp, average='weighted')
    precision = precision_score(y_true, y_pred_lbp, average='weighted')
    f1 = f1_score(y_true, y_pred_lbp, average='weighted')
    specificity = recall_score(y_true, y_pred_lbp, average='weighted')

    # Create a colored confusion matrix
    plt.figure(figsize=(6, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.show()

    return accuracy, tpr, specificity, f1

# Evaluate and display confusion matrices for each model
rf_accuracy_lbp, rf_tpr_lbp, rf_specificity_lbp, rf_f1_lbp = _
    ↪ evaluate_model(y_valid_lbp, rf_pred_lbp, 'Random Forest LBP VALID')
rna_accuracy_lbp, rna_tpr_lbp, rna_specificity_lbp, rna_f1_lbp = _
    ↪ evaluate_model(y_valid_lbp, rna_pred_lbp, 'RNA LBP VALID')
svm_accuracy_lbp, svm_tpr_lbp, svm_specificity_lbp, svm_f1_lbp = _
    ↪ evaluate_model(y_valid_lbp, svm_pred_lbp, 'SVM LBP VALID')

# Create a DataFrame to store the results
results_df_lbp = pd.DataFrame({
    'Model': ['Random Forest LBP VALID', 'RNA LBP VALID', 'SVM LBP_
    ↪ VALID'],
    'Accuracy': [rf_accuracy_lbp, rna_accuracy_lbp, svm_accuracy_lbp],
    'Sensitivity': [rf_tpr_lbp, rna_tpr_lbp, svm_tpr_lbp],
    'Specificity': [rf_specificity_lbp, rna_specificity_lbp, _
    ↪ svm_specificity_lbp],

```

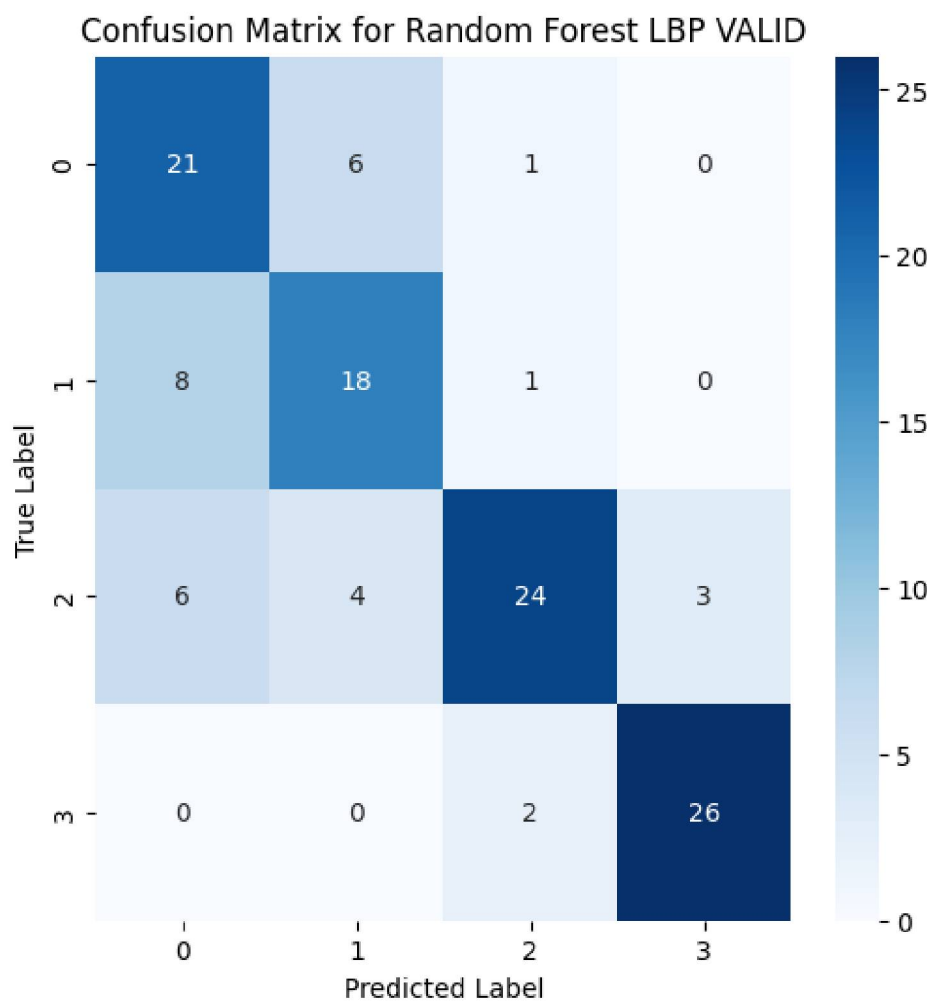
```

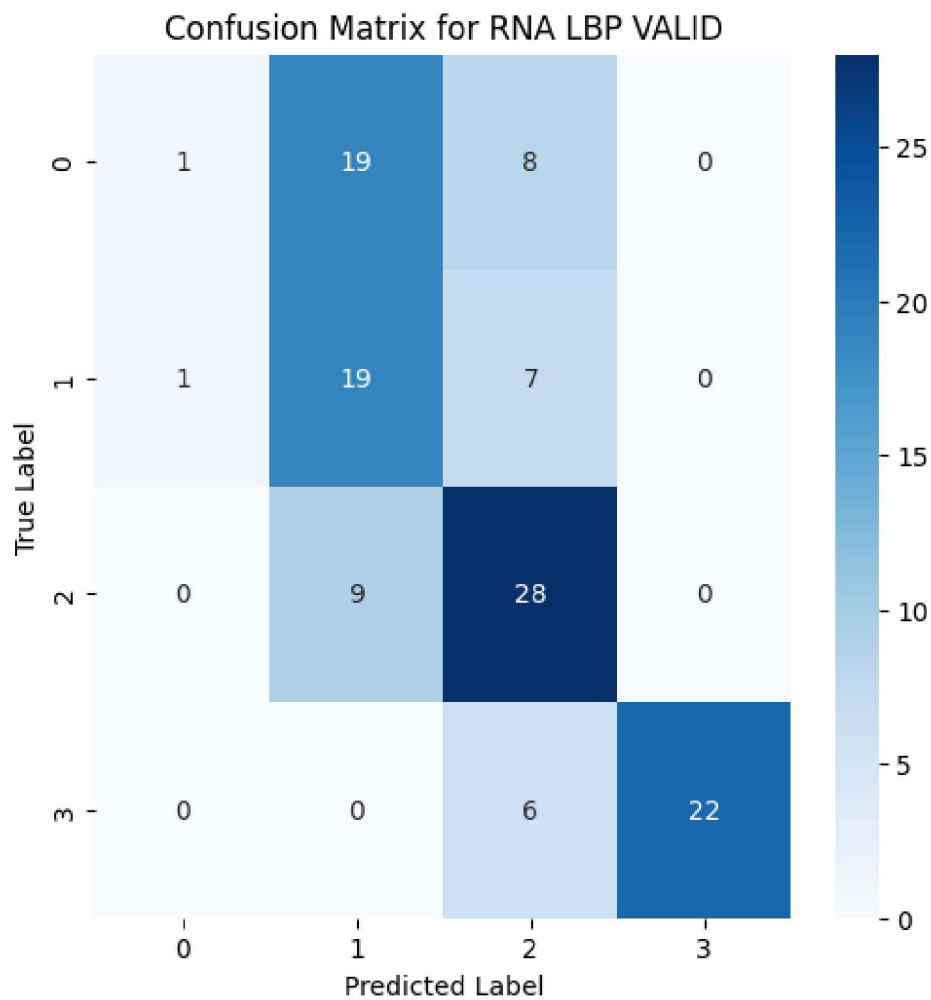
    'F1Score': [rf_f1_lbp, rna_f1_lbp, svm_f1_lbp]
})

# Sort the DataFrame by F1Score in descending order
results_df_lbp = results_df_lbp.sort_values('F1Score', ascending=False)

# Display the results
print(results_df_lbp)

```

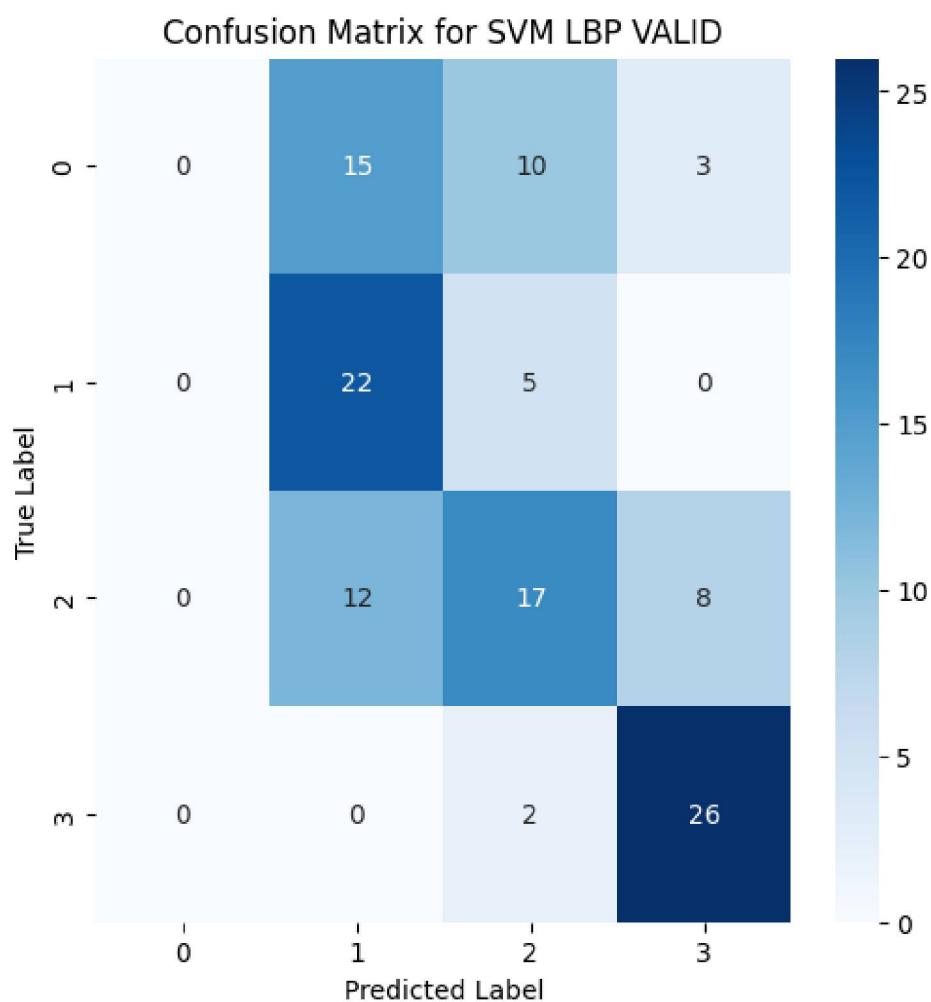




```

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:
1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

```

	Model	Accuracy	Sensibility	Specificity	F1Score
0	Random Forest LBP VALID	0.741667	0.741667	0.741667	0.743386
1	RNA LBP VALID	0.583333	0.583333	0.583333	0.537205
2	SVM LBP VALID	0.541667	0.541667	0.541667	0.464582

```
[ ]: # Import necessary classes from scikit-learn
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score, \
    recall_score, precision_score, f1_score
import seaborn as sns
import matplotlib.pyplot as plt
```

```

# Load the CSV files
train_df_vgg = pd.read_csv('train_vgg16_features.csv')
valid_df_vgg = pd.read_csv('valid_vgg16_features.csv')

# Convert 'Label' column to integer type
train_df_vgg['Label'] = train_df_vgg['Label'].astype(int)
valid_df_vgg['Label'] = valid_df_vgg['Label'].astype(int)

# For the training set:
X_train_vgg = train_df_vgg.iloc[:, :-2] # Select all rows, all columns,
↳ except the last two
y_train_vgg = train_df_vgg['Label']

# For the validation set:
X_valid_vgg = valid_df_vgg.iloc[:, :-2] # Same logic for validation set
y_valid_vgg = valid_df_vgg['Label']

# Initialize models
rf_model_vgg = RandomForestClassifier()
rna_model_vgg = MLPClassifier()
svm_model_vgg = SVC()

# Train the models
rf_model_vgg.fit(X_train_vgg, y_train_vgg)
rna_model_vgg.fit(X_train_vgg, y_train_vgg)
svm_model_vgg.fit(X_train_vgg, y_train_vgg)

# Make predictions on the valid set
rf_pred_vgg = rf_model_vgg.predict(X_valid_vgg)
rna_pred_vgg = rna_model_vgg.predict(X_valid_vgg)
svm_pred_vgg = svm_model_vgg.predict(X_valid_vgg)

# Function to calculate metrics and create confusion matrix
def evaluate_model(y_true, y_pred_vgg, model_name):
    cm = confusion_matrix(y_true, y_pred_vgg)
    accuracy = accuracy_score(y_true, y_pred_vgg)
    tpr = recall_score(y_true, y_pred_vgg, average='weighted')
    precision = precision_score(y_true, y_pred_vgg, average='weighted')
    f1 = f1_score(y_true, y_pred_vgg, average='weighted')
    specificity = recall_score(y_true, y_pred_vgg, average='weighted')

# Create a colored confusion matrix
plt.figure(figsize=(6, 6))

```

```

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title(f'Confusion Matrix for {model_name}')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

return accuracy, tpr, specificity, f1

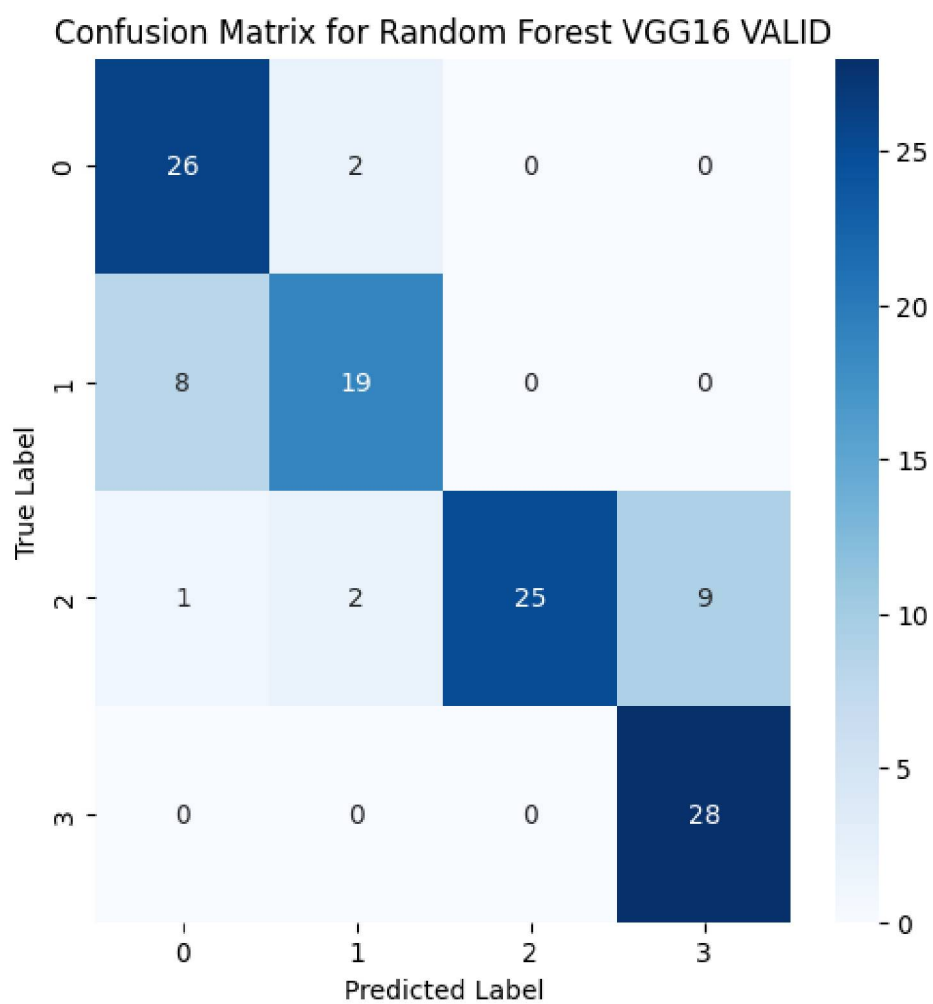
# Evaluate and display confusion matrices for each model
rf_vgg_accuracy, rf_vgg_tpr, rf_vgg_specificity, rf_vgg_f1 = _
    evaluate_model(y_valid_vgg, rf_pred_vgg, 'Random Forest VGG16 VALID')
rna_vgg_accuracy, rna_vgg_tpr, rna_vgg_specificity, rna_vgg_f1 = _
    evaluate_model(y_valid_vgg, rna_pred_vgg, 'RNA VGG16 VALID')
svm_vgg_accuracy, svm_vgg_tpr, svm_vgg_specificity, svm_vgg_f1 = _
    evaluate_model(y_valid_vgg, svm_pred_vgg, 'SVM VGG16 VALID')

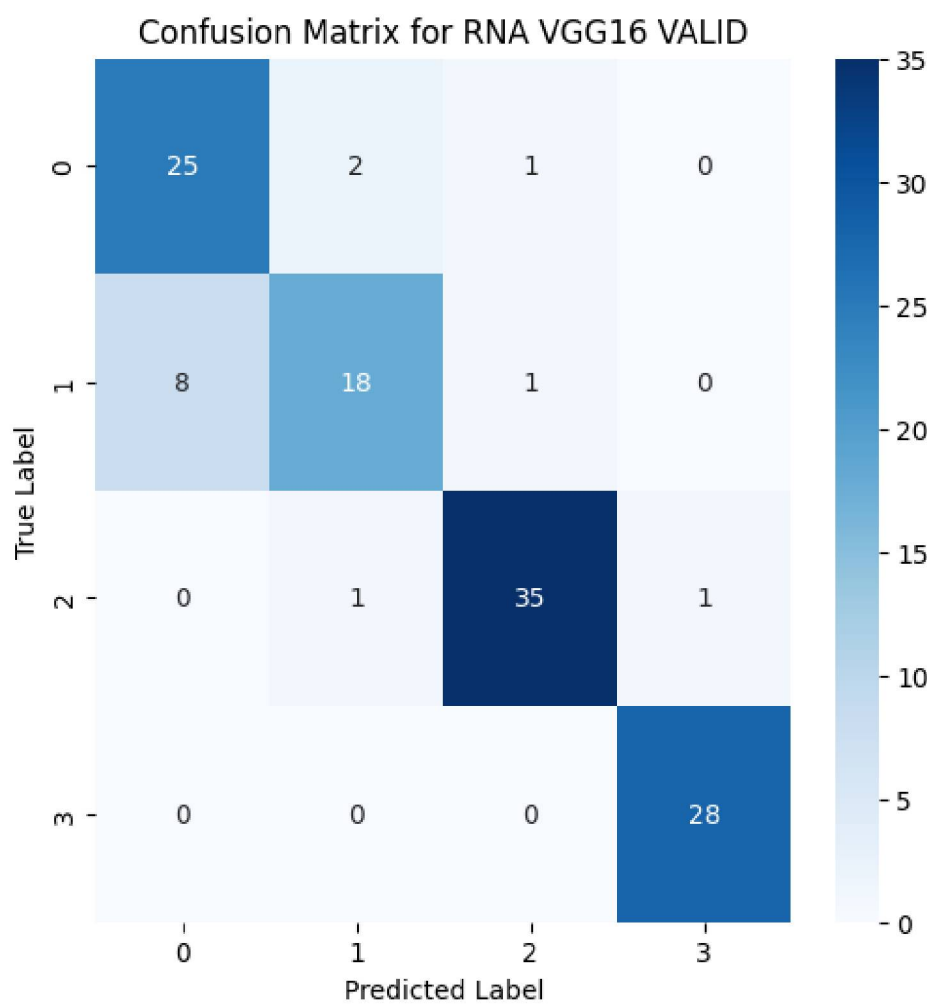
# Create a DataFrame to store the results
results_df_vgg = pd.DataFrame({
    'Model': ['Random Forest VGG16 VALID', 'RNA VGG16 VALID', 'SVM VGG16',
    evaluate_model(y_valid_vgg, svm_pred_vgg, 'SVM VGG16 VALID')],
    'Accuracy': [rf_vgg_accuracy, rna_vgg_accuracy, svm_vgg_accuracy],
    'Sensitivity': [rf_vgg_tpr, rna_vgg_tpr, svm_vgg_tpr],
    'Specificity': [rf_vgg_specificity, rna_vgg_specificity,
    evaluate_model(y_valid_vgg, svm_pred_vgg, 'SVM VGG16 VALID')],
    'F1Score': [rf_vgg_f1, rna_vgg_f1, svm_vgg_f1]
})

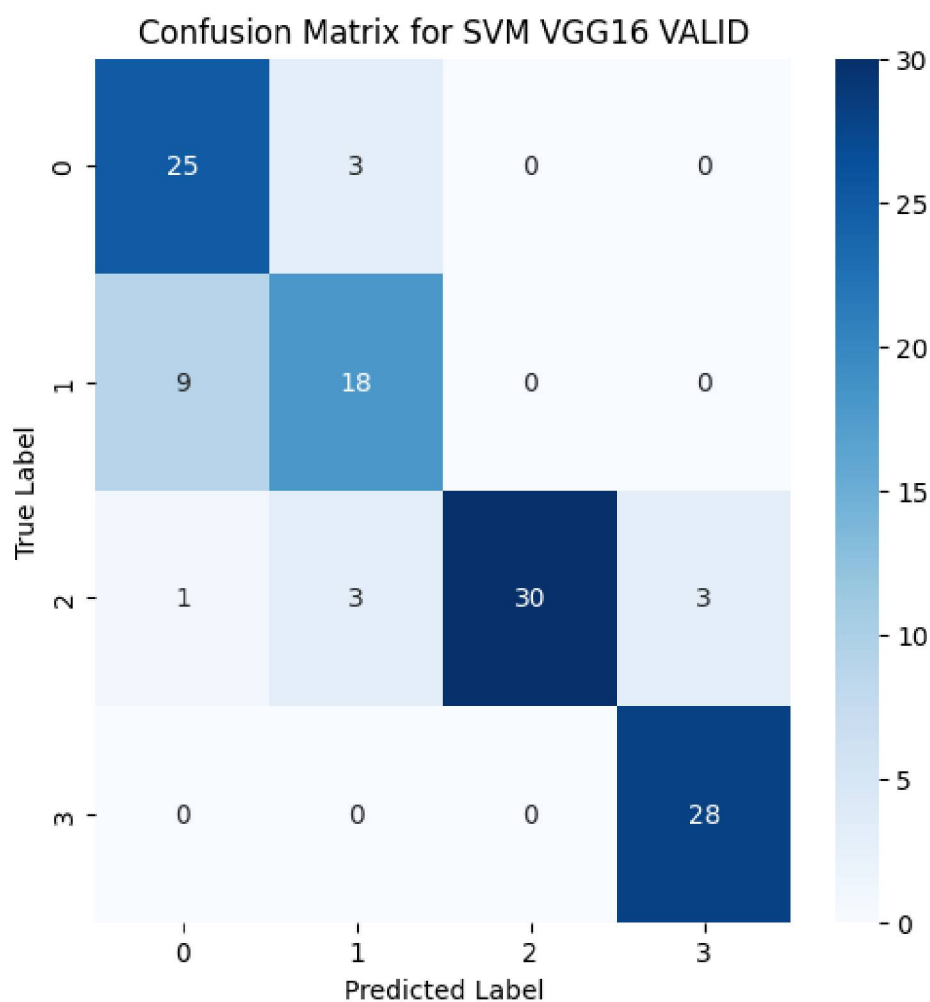
# Sort the DataFrame by F1Score in descending order
results_df_vgg = results_df_vgg.sort_values('F1Score', ascending=False)

# Display the results
print(results_df_vgg)

```







		Model	Accuracy	Sensibility	Specificity	F1Score
1	RNA	VGG16 VALID	0.883333	0.883333	0.883333	0.880913
2	SVM	VGG16 VALID	0.841667	0.841667	0.841667	0.841597
0	Random Forest	VGG16 VALID	0.816667	0.816667	0.816667	0.813274

```
[ ]: # prompt: juntar os dataframes de resultados acima e ordenar por maior
      ↳ f1score

# Concatenate the two DataFrames
combined_results_df = pd.concat([results_df_lbp, results_df_vgg],
      ↳ ignore_index=True)

# Sort the combined DataFrame by F1Score in descending order
```

```
combined_results_df = combined_results_df.sort_values('F1Score',
↳ascending=False)
```

```
# Display the combined and sorted results
print(combined_results_df)
```

		Model	Accuracy	Sensibility	Specificity	F1Score
3	RNA VGG16	VALID	0.883333	0.883333	0.883333	0.880913
4	SVM VGG16	VALID	0.841667	0.841667	0.841667	0.841597
5	Random Forest VGG16	VALID	0.816667	0.816667	0.816667	0.813274
0	Random Forest LBP	VALID	0.741667	0.741667	0.741667	0.743386
1	RNA LBP	VALID	0.583333	0.583333	0.583333	0.537205
2	SVM LBP	VALID	0.541667	0.541667	0.541667	0.464582

T5 Carregue a base de Teste e execute a tarefa 3 nesta base.

```
[ ]: from google.colab import files
```

```
!unzip '/content/drive/MyDrive/Colab Notebooks/Visão Computacional/
↳Test_Warwick.zip'
```

```
Archive: /content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Test_Warwick.zip
```

```
creating: Test_4cl_amostra/
```

```
creating: Test_4cl_amostra/0/
```

```
inflating: Test_4cl_amostra/0/70_HER2_23828.png
```

```
inflating: Test_4cl_amostra/0/70_HER2_23827.png
```

```
...
```

```
inflating: Test_4cl_amostra/1/34_HER2_18120.png
```

```
inflating: Test_4cl_amostra/1/34_HER2_17287.png
```

```
inflating: Test_4cl_amostra/1/34_HER2_18288.png
```

```
inflating: Test_4cl_amostra/1/34_HER2_6302.png
```

```
[ ]: # Set the path to your data
```

```
data_path = '/content/Test_4cl_amostra'
```

```
import os
```

```
# Dictionary to store image paths grouped by patient ID
```

```
patient_dict = {}
```

```
# List to store all image file paths
```

```
all_files = []
```



```

# Assume data_path is defined earlier in your code
# data_path = "path/to/your/data/directory"

# Loop through class folders and images to extract patient ID and store
→ image paths
for class_folder in os.listdir(data_path):
    class_path = os.path.join(data_path, class_folder)
    if os.path.isdir(class_path):
        for img_file in os.listdir(class_path):
            if img_file.endswith('.png'):
                patient_id = img_file.split('_')[0] # Extract the
→ patient number XX
                full_path = os.path.join(class_path, img_file)

                # Add to patient dictionary
                if patient_id not in patient_dict:
                    patient_dict[patient_id] = []
                patient_dict[patient_id].append(full_path)

                # Add to all_files list
                all_files.append(full_path)

# Print some statistics
print(f"Total number of images: {len(all_files)}")
print(f"Number of unique patients: {len(patient_dict)}")

# If you need to use the files later in your code, you can use the
→ all_files list
# For example:
# for file_path in all_files:
#     # Process each file
#     pass

```

Total number of images: 371

Number of unique patients: 13

```
[ ]: Calcula_caracteristicas(all_files, 'testfinal_lbp_features.csv')
```

```
[ ]: imagens_all, rotulos_all, pacientes_all =carregar_dados_imagens(data_path)
treino(imagens_all, rotulos_all, pacientes_all, 'testfinal_vgg16_features.
→ csv')
```

WARNING:tensorflow:5 out of the last 20 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.
→ one_step_on_data_distributed at


```

4      0.697009      0.0      1      34
..      ...      ...      ...
366    1.133308      0.0      0      66
367    0.955719      0.0      0      73
368    0.621897      0.0      0      68
369    0.749184      0.0      0      68
370    0.695996      0.0      0      68

```

```
[371 rows x 25090 columns]
```

Q1T6. Aplique os modelos treinados nos dados de teste

Q1T7. Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.

```
[ ]: test_df_lbpfinal = pd.read_csv('testfinal_lbp_features.csv')

# Convert 'Label' column to integer type
test_df_lbpfinal['Label'] = test_df_lbpfinal['Label'].astype(int)

# Separate features and labels for train and test sets
X_test_lbpfinal = test_df_lbpfinal.drop(columns=['Nome do Arquivo',
↪ 'Label'])
y_test_lbpfinal = test_df_lbpfinal['Label']

# Make predictions on the test set
rf_pred_lbpfinal = rf_model_lbp.predict(X_test_lbpfinal)
rna_pred_lbpfinal = rna_model_lbp.predict(X_test_lbpfinal)
svm_pred_lbpfinal = svm_model_lbp.predict(X_test_lbpfinal)

# Function to calculate metrics and create confusion matrix
def evaluate_model(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    accuracy = accuracy_score(y_true, y_pred)
    tpr = recall_score(y_true, y_pred, average='weighted')
    precision = precision_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    specificity = recall_score(y_true, y_pred, average='weighted')

# Create a colored confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title(f'Confusion Matrix for {model_name}')
plt.xlabel('Predicted Label')
```

```

plt.ylabel('True Label')
plt.show()

return accuracy, tpr, specificity, f1

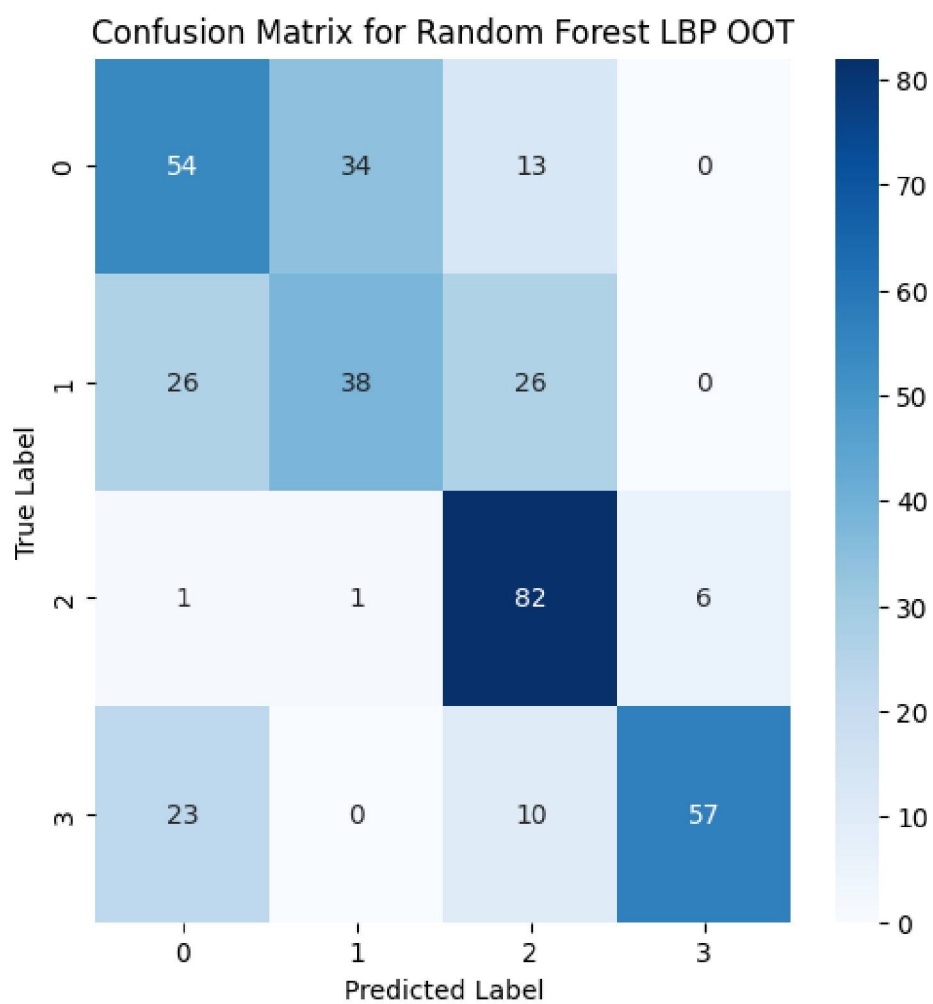
# Evaluate and display confusion matrices for each model
rf_lbpfinal_accuracy, rf_lbpfinal_tpr, rf_lbpfinal_specificity,
    ↪rf_lbpfinal_f1 = evaluate_model(y_test_lbpfinal, rf_pred_lbpfinal,
    ↪'Random Forest LBP OOT')
rna_lbpfinal_accuracy, rna_lbpfinal_tpr, rna_lbpfinal_specificity,
    ↪rna_lbpfinal_f1 = evaluate_model(y_test_lbpfinal, rna_pred_lbpfinal,
    ↪'RNA LBP OOT')
svm_lbpfinal_accuracy, svm_lbpfinal_tpr, svm_lbpfinal_specificity,
    ↪svm_lbpfinal_f1 = evaluate_model(y_test_lbpfinal, svm_pred_lbpfinal,
    ↪'SVM LBP OOT')

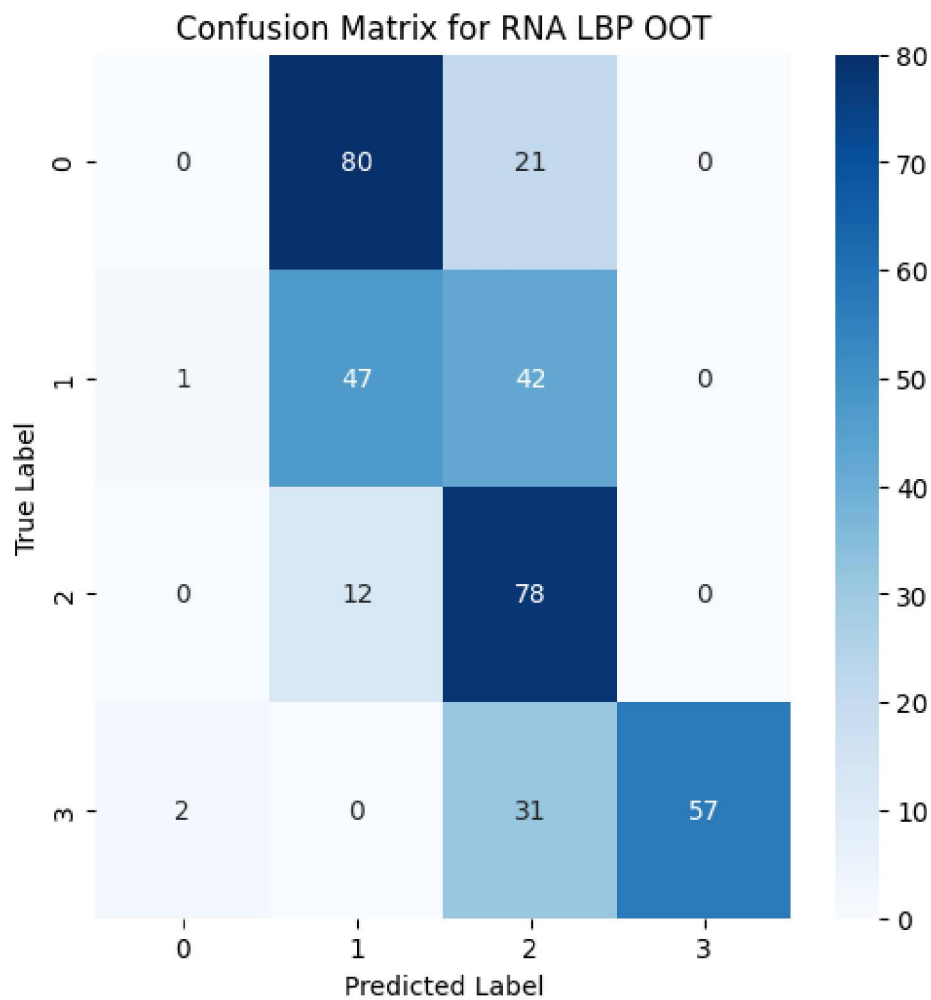
# Create a DataFrame to store the results
results_df_vggfinal = pd.DataFrame({
    'Model': ['Random Forest LBP OOT', 'RNA LBP OOT', 'SVM LBP OOT'],
    'Accuracy': [rf_lbpfinal_accuracy, rna_lbpfinal_accuracy,
    ↪svm_lbpfinal_accuracy],
    'Sensitivity': [rf_lbpfinal_tpr, rna_lbpfinal_tpr, svm_lbpfinal_tpr],
    'Specificity': [rf_lbpfinal_specificity, rna_lbpfinal_specificity,
    ↪svm_lbpfinal_specificity],
    'F1Score': [rf_lbpfinal_f1, rna_lbpfinal_f1, svm_lbpfinal_f1]
})

# Sort the DataFrame by F1Score in descending order
results_df_lbpfinal = results_df_vggfinal.sort_values('F1Score',
    ↪ascending=False)

# Display the results
print(results_df_lbpfinal)

```

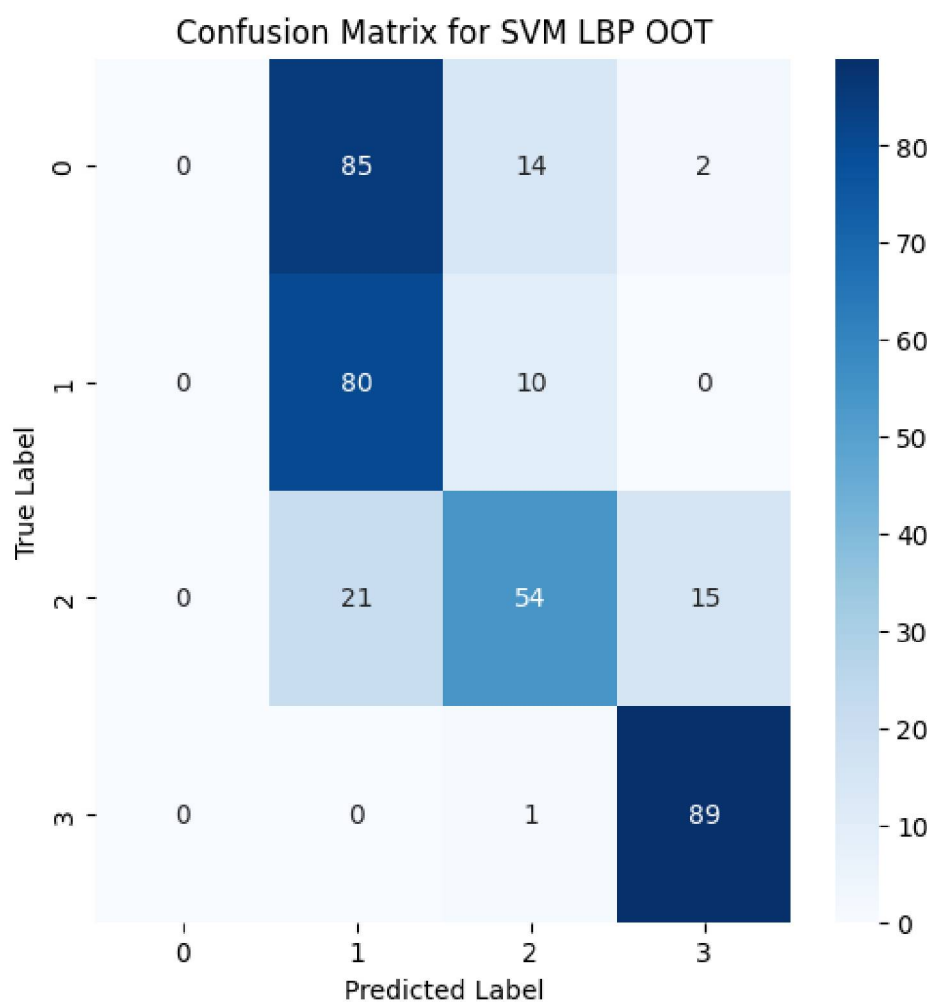




```

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:
1565:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in
labels
with no predicted samples. Use `zero_division` parameter to control this
behavior.
_warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))

```



	Model	Accuracy	Sensibility	Specificity	F1Score
0	Random Forest LBP OOT	0.622642	0.622642	0.622642	0.617302
2	SVM LBP OOT	0.601078	0.601078	0.601078	0.515966
1	RNA LBP OOT	0.490566	0.490566	0.490566	0.432148

```
[ ]: test_df_vggfinal = pd.read_csv('testfinal_vgg16_features.csv')

# Convert 'Label' column to integer type
test_df_vggfinal['Label'] = test_df_vggfinal['Label'].astype(int)

# Separate features and labels for train and test sets
X_test_vggfinal = test_df_vggfinal.iloc[:, :-2]
y_test_vggfinal = test_df_vggfinal['Label']
```

```

# Make predictions on the test set
rf_pred_vggfinal = rf_model_vgg.predict(X_test_vggfinal)
rna_pred_vggfinal = rna_model_vgg.predict(X_test_vggfinal)
svm_pred_vggfinal = svm_model_vgg.predict(X_test_vggfinal)

# Function to calculate metrics and create confusion matrix
def evaluate_model(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    accuracy = accuracy_score(y_true, y_pred)
    tpr = recall_score(y_true, y_pred, average='weighted')
    precision = precision_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    specificity = recall_score(y_true, y_pred, average='weighted')

    # Create a colored confusion matrix
    plt.figure(figsize=(6, 6))
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.show()

    return accuracy, tpr, specificity, f1

# Evaluate and display confusion matrices for each model
rf_vggfinal_accuracy, rf_vggfinal_tpr, rf_vggfinal_specificity, \
    rf_vggfinal_f1 = evaluate_model(y_test_vggfinal, rf_pred_vggfinal, \
    'Random Forest VGG16 OOT')
rna_vggfinal_accuracy, rna_vggfinal_tpr, rna_vggfinal_specificity, \
    rna_vggfinal_f1 = evaluate_model(y_test_vggfinal, rna_pred_vggfinal, \
    'RNA VGG16 OOT')
svm_vggfinal_accuracy, svm_vggfinal_tpr, svm_vggfinal_specificity, \
    svm_vggfinal_f1 = evaluate_model(y_test_vggfinal, svm_pred_vggfinal, \
    'SVM VGG16 OOT')

# Create a DataFrame to store the results
results_df_vggfinal = pd.DataFrame({
    'Model': ['Random Forest VGG16 OOT', 'RNA VGG16 OOT', 'SVM VGG16 OOT'],
    'Accuracy': [rf_vggfinal_accuracy, rna_vggfinal_accuracy, \
    svm_vggfinal_accuracy],
    'Sensitivity': [rf_vggfinal_tpr, rna_vggfinal_tpr, svm_vggfinal_tpr],

```

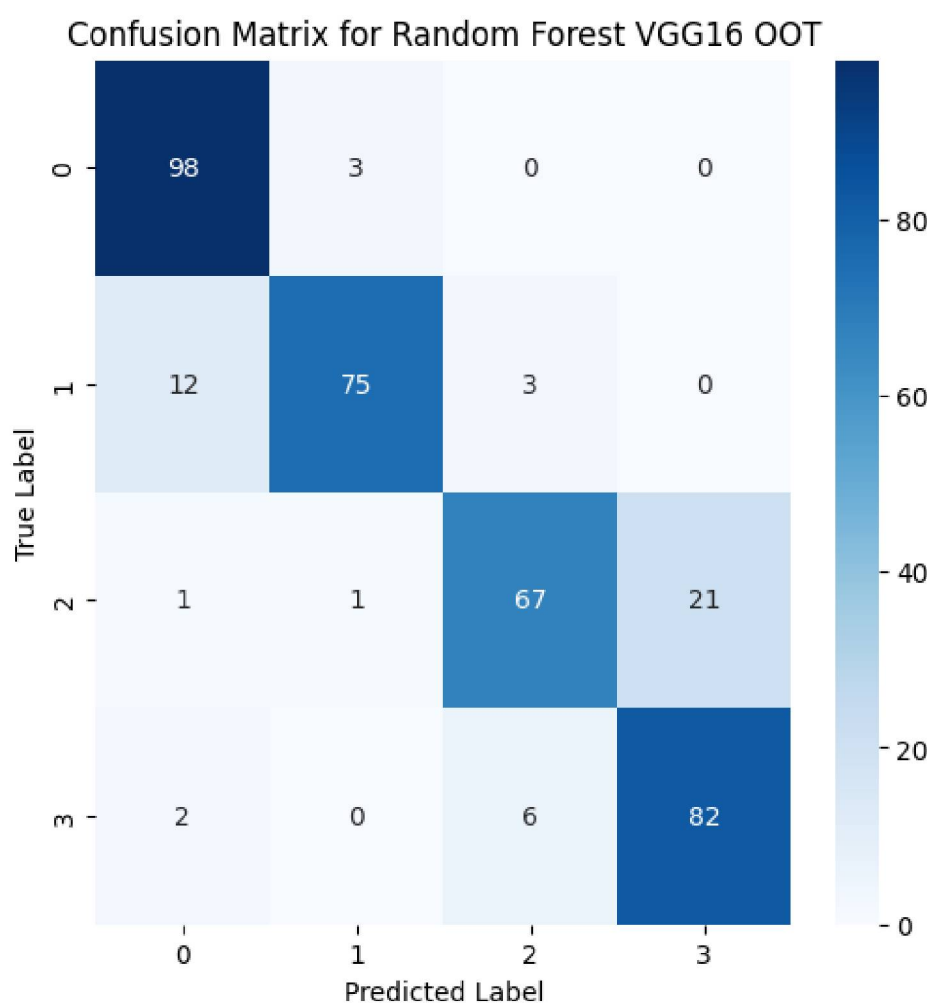
```

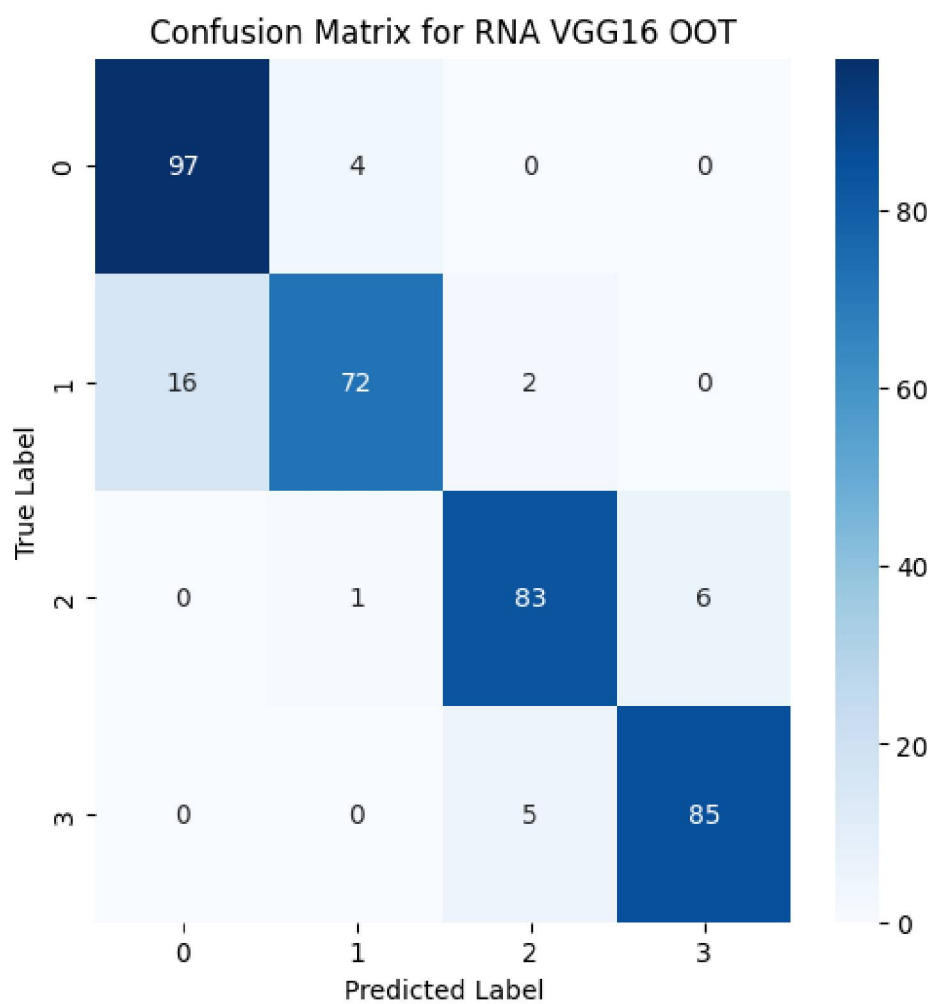
    'Specificity': [rf_vggfinal_specificity, rna_vggfinal_specificity,
    ↪svm_vggfinal_specificity],
    'F1Score': [rf_vggfinal_f1, rna_vggfinal_f1, svm_vggfinal_f1]
})

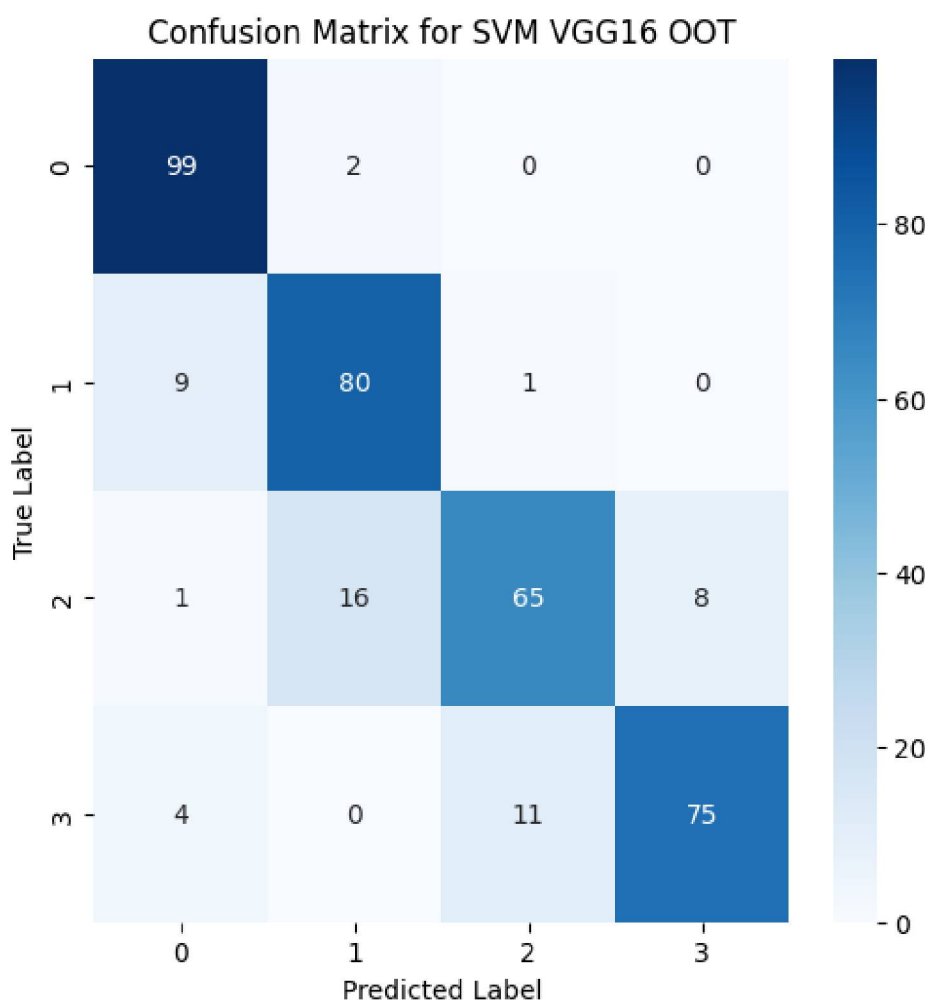
# Sort the DataFrame by F1Score in descending order
results_df_vggfinal = results_df_vggfinal.sort_values('F1Score',
    ↪ascending=False)

# Display the results
print(results_df_vggfinal)

```







	Model	Accuracy	Sensibility	Specificity	F1Score
1	RNA VGG16 OOT	0.908356	0.908356	0.908356	0.907536
0	Random Forest VGG16 OOT	0.867925	0.867925	0.867925	0.866614
2	SVM VGG16 OOT	0.859838	0.859838	0.859838	0.857517

Q1T8. Indique qual modelo dá o melhor o resultado e a métrica utilizada

```
[ ]: # prompt: juntar os dataframes de resultados acima e ordenar por maior
      ↳ f1score

# Concatenate the two DataFrames
combined_results_dffinal = pd.concat([results_df_lbpfinal,
      ↳ results_df_vggfinal], ignore_index=True)
```

```
# Sort the combined DataFrame by F1Score in descending order
combined_results_dffinal = combined_results_dffinal.
    ↪sort_values('F1Score', ascending=False)

# Display the combined and sorted results
print(combined_results_dffinal)
```

	Model	Accuracy	Sensibility	Specificity	F1Score
3	RNA VGG16 OOT	0.908356	0.908356	0.908356	0.907536
4	Random Forest VGG16 OOT	0.867925	0.867925	0.867925	0.866614
5	SVM VGG16 OOT	0.859838	0.859838	0.859838	0.857517
0	Random Forest LBP OOT	0.622642	0.622642	0.622642	0.617302
1	SVM LBP OOT	0.601078	0.601078	0.601078	0.515966
2	RNA LBP OOT	0.490566	0.490566	0.490566	0.432148

RESULTADO Q1: Os modelos SVM e Random Forest treinados com as características extraídas do modelo CNN VGG16 foram os melhores resultados por apresentarem maior F1-Score, ou seja, além de conseguir classificar corretamente maior proporção dos pacientes que realmente tinham câncer de mama, também tem a melhor especificidade ao classificar corretamente os pacientes que não têm a doença.

0.1 Q2. Redes Neurais Artificiais

Q2T1 Utilize a base de dados de Treino já separadas em treino e validação do exercício anterior

```
[ ]: import os
from keras.models import Model
from keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.layers import Dense, Dropout, Flatten
from keras.applications.vgg16 import VGG16
from pathlib import Path
import numpy as np
import pandas as pd

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

Collecting livelossplot

Downloading livelossplot-0.5.6-py3-none-any.whl.metadata (8.9 kB)

Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (from livelossplot) (3.10.0)

```
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->livelossplot)
(2.9.0.post0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from Jinja2>=2.
  ↳9->bokeh->livelossplot)
(3.0.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/
  ↳dist-
packages (from pandas>=1.2->bokeh->livelossplot) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
  ↳dist-
packages (from pandas>=1.2->bokeh->livelossplot) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.7->matplotlib->livelossplot) (1.17.0)
Downloading livelossplot-0.5.6-py3-none-any.whl (23 kB)
Installing collected packages: livelossplot
Successfully installed livelossplot-0.5.6
```

```
[ ]: !pip install tensorflow==2.15
```

```
Collecting tensorflow==2.15
  Downloading tensorflow-2.15.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.4 kB)
...
```

```
Successfully installed keras-2.15.0 ml-dtypes-0.2.0 tensorboard-2.15.2
tensorflow-2.15.0 tensorflow-estimator-2.15.0 wrapt-1.14.1
tf-keras 2.17.0 requires tensorflow<2.18,>=2.17, but you have tensorflow 2.
  ↳15.0
```

Q2T3 Treine modelos VGG16 e Resnet50 adaptadas com e sem Data Augmentation

Data Augmentation O ImageDataGenerator vai alterar as imagens de acordo com as transformações indicada dentro dela. Isso irá criar novas imagens com essas alterações. A quantidade de imagens é definida pelo batch_size que define para cada ciclo de retreinamento o número de imagens criadas. A opção rescale ajuda no treinamento, pois normaliza os valores da imagem entre 0 e 1.

```
[ ]: # Leitura e extração Zip - Dados de Treinamento
from google.colab import files
```

```
!unzip '/content/drive/MyDrive/Colab Notebooks/Visão Computacional/
↳Test_Warwick.zip'
```

```
Archive: /content/drive/MyDrive/Colab Notebooks/Visão
Computacional/Test_Warwick.zip
replace Test_4cl_amostra/0/70_HER2_23828.png? [y]es, [n]o, [A]ll, [N]one,
[r]ename:
```

```
[ ]: data_teste = '/content/Test_4cl_amostra'
data_train = '/content/drive/MyDrive/Colab Notebooks/Visão Computacional/
↳Train_4cls_amostra/Pasta Treino'
data_val = '/content/drive/MyDrive/Colab Notebooks/Visão Computacional/
↳Train_4cls_amostra/Pasta Validacao'
```

```
[ ]: from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.applications.resnet50 import preprocess_input #_
↳Import ResNet50 preprocess_input

batch_size = 32

def create_resnet_augmentation_generators(train_df, valid_df, test_df):

    train_generator = ImageDataGenerator(
        rotation_range=90,
        brightness_range=[0.1, 0.7],
        width_shift_range=0.5,
        height_shift_range=0.5,
        horizontal_flip=True,
        vertical_flip=True,
        preprocessing_function=preprocess_input)_
    ↳# Use ResNet50 preprocess_input

    valid_generator = _
    ↳ImageDataGenerator(preprocessing_function=preprocess_input) # Use_
    ↳ResNet50 preprocess_input

    test_generator = _
    ↳ImageDataGenerator(preprocessing_function=preprocess_input) # Use_
    ↳ResNet50 preprocess_input

    traingen = train_generator.flow_from_dataframe(
        train_df,
```

```

        x_col='filename',
        y_col='class',
        target_size=(224,224),
        batch_size=batch_size,
        class_mode='categorical',
        classes=class_subset,
        shuffle=True,
        seed=42)

validgen = valid_generator.flow_from_dataframe(
    valid_df,
    x_col='filename',
    y_col='class',
    target_size=(224,224),
    batch_size=batch_size,
    class_mode='categorical',
    classes=class_subset,
    shuffle=True,
    seed=42)

testgen = test_generator.flow_from_dataframe(test_df,
    target_size=(224,224),
    batch_size=batch_size,
    class_mode='categorical',
    classes=class_subset,
    shuffle=False,
    seed=42)

return traingen, validgen, testgen

```

```

[ ]: class_subset = ['0','1','2','3']

def create_dataframe(file_list):
    labels = []
    for file_path in file_list:
        # Extract the label (folder name) from the file path
        label = os.path.basename(os.path.dirname(file_path))
        labels.append(label)

    df = pd.DataFrame({
        'filename': file_list,
        'class': labels
    })

```



```

    # Convert numeric labels to strings (if necessary)
    # df['class'] = df['class'].astype(str) # This line might be
    ↳ redundant if labels are already strings

    return df

train_files = gravar_lista_imagens(data_train)
valid_files = gravar_lista_imagens(data_val)
test_files = gravar_lista_imagens(data_teste)

train_df = create_dataframe(train_files)
valid_df = create_dataframe(valid_files)
test_df = create_dataframe(test_files)

```

```

[ ]: traingen, validgen, testgen =
    ↳ create_resnet_augmentation_generators(train_df, valid_df, test_df)

```

Found 473 validated image filenames belonging to 4 classes.

Found 120 validated image filenames belonging to 4 classes.

Found 371 validated image filenames belonging to 4 classes.

Resnet treinada do zero

```

[ ]: from tensorflow.keras.applications.resnet50 import ResNet50

    # A opção include_top=False não inclui as camadas de aprendizado da rede
    ↳ original
    resnet_0 = ResNet50(input_shape=(224,224,3), weights='imagenet',
    ↳ include_top=False)

    # treinar os pesos existentes
    for layer in resnet_0.layers:
        layer.trainable = False

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 5s
0us/step

Criação das camadas Fully Connected para o treinamento

```

[ ]: # camadas próprias - você pode colocar mais se quiser
    # A saída da resnet será a entrada da camada criada
    x = Flatten()(resnet_0.output)

```

```
# camada de classificação com as 4 classes utilizadas
prediction = Dense(len(class_subset), activation='softmax')(x)

# Criação do Objeto Modelo (a parte da resnet + as camadas Fully
↳connected criadas)
model_0 = Model(inputs=resnet_0.input, outputs=prediction)
```

```
[ ]: model_0.summary() # Impressão das arquitetura da rede
```

Model: "functional_3"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (None, 7, 7, (BatchNormalizatio...	(None, 224, 8,192 2048)		conv5_block3_3_c...
...			
dense (Dense)	(None, 4)	401,412	flatten[0][0]

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

Trainable params: 401,412 (1.53 MB)

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

Resnet50 com Data Augmentation

```
[ ]: %%time
# Otimizador propagação da Raiz quadrada da média ao quadrado (Root Mean
↳Squared Propagation)
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras
from tensorflow.keras.utils import to_categorical

steps_per_epoch = traingen.samples // batch_size
val_steps = validgen.samples // batch_size

n_epochs = 10
```

```

optimizer = RMSprop(learning_rate=0.0001)

model_0.compile(loss='categorical_crossentropy', optimizer=optimizer,
                 metrics=['accuracy'])

#Trainning the model
# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer = ModelCheckpoint(filepath='img_model_0.weights.best.keras',
                               verbose=1,
                               save_best_only=True)

# Para o treinamento para prevenir o overfitting
early_stop = EarlyStopping(monitor='val_loss',
                           patience=10,
                           restore_best_weights=True,
                           mode='min')

```

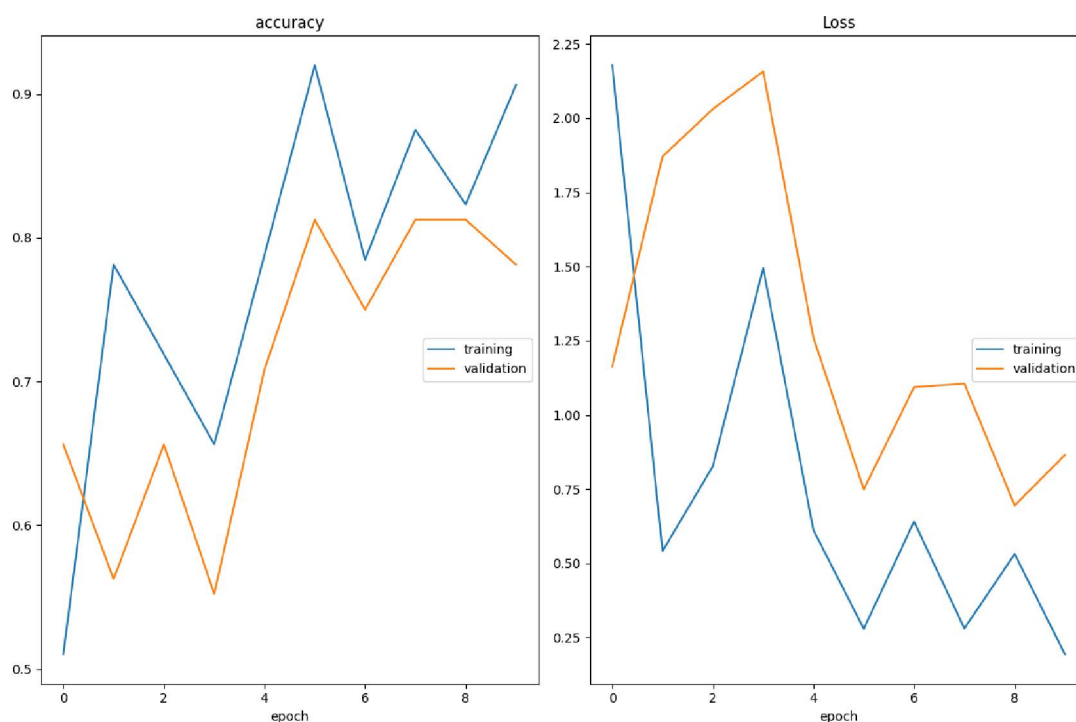
CPU times: user 14.8 ms, sys: 2.98 ms, total: 17.8 ms

Wall time: 17.6 ms

```

[ ]: history_0 = model_0.fit(training,
                             epochs=n_epochs,
                             steps_per_epoch=steps_per_epoch,
                             validation_data=validation_data,
                             validation_steps=val_steps,
                             callbacks=[checkpointer, PlotLossesKeras()],
                             verbose=False)

```

```

accuracy
    training      (min: 0.510, max: 0.920, cur: 0.
↪906)
    validation    (min: 0.552, max: 0.812, cur: 0.
↪781)
Loss
    training      (min: 0.194, max: 2.179, cur: 0.
↪194)
    validation    (min: 0.695, max: 2.157, cur: 0.
↪865)

```

####Resnet sem data augmentation

```

[ ]: # Utiliza os pesos treinados na base imagenet
resnet_sem_augment = ResNet50(input_shape=(224,224,3),
↪weights='imagenet', include_top=False)

for layer in resnet_sem_augment.layers:
    layer.trainable = False

```

Rede igual ao treinamento do zero, porém utilizando os pesos da Imagenet.

```
[ ]: x_sem_augment = Flatten()(resnet_sem_augment.output)

prediction = Dense(len(class_subset), activation='softmax')(x_sem_augment)

model_sem_augment = Model(inputs=resnet_sem_augment.input,
    ↳ outputs=prediction)

model_sem_augment.summary()
```

Model: "functional_4"

Layer (type)	Output Shape	Param #	Connected to
input_layer_4 (InputLayer)	(None, 224, 224, 3)	0	-
...			
dense_1 (Dense)	(None, 4)	401,412	flatten_1[0][0]

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

Trainable params: 401,412 (1.53 MB)

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

#####Treinamento sem data augmentation

```
[ ]: #Gerando imagem sem data augmentation
def create_resnet_sem_augmentation_generators(train_df, valid_df):
    # Create training data generator
    train_sem_augmentation_gen =
    ↳ ImageDataGenerator(preprocessing_function=preprocess_input) # Use
    ↳ ResNet50 preprocess_input
    valid_sem_augmentation_gen =
    ↳ ImageDataGenerator(preprocessing_function=preprocess_input) # Use
    ↳ ResNet50 preprocess_input

    traingen = train_sem_augmentation_gen.flow_from_dataframe(
        train_df,
```

```

        x_col='filename',
        y_col='class',
        target_size=(224,224),
        batch_size=64,
        class_mode='categorical',
        classes=class_subset,
        shuffle=True,
        seed=42)

validgen = valid_sem_augmentation_gen.flow_from_dataframe(
    valid_df,
    x_col='filename',
    y_col='class',
    target_size=(224,224),
    batch_size=64,
    class_mode='categorical',
    classes=class_subset,
    shuffle=True,
    seed=42)

return traingen, validgen

```

```

[ ]: # Corrected ResNet without Data Augmentation training setup
from tensorflow.keras.applications.resnet50 import ResNet50 # Import
    →ResNet50 again for clarity

# Utiliza os pesos treinados na base imagenet
resnet_sem_augment = ResNet50(input_shape=(224,224,3),
    →weights='imagenet', include_top=False)

# Freeze the pre-trained layers
for layer in resnet_sem_augment.layers:
    layer.trainable = False

```

```

[ ]: %%time

from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras
from tensorflow.keras.utils import to_categorical

steps_per_epoch = traingen.samples // batch_size

```

```

val_steps = validgen.samples // batch_size

n_epochs = 10

optimizer = RMSprop(learning_rate=0.0001)

model_sem_augment.compile(loss='categorical_crossentropy',
    ↳optimizer=optimizer, metrics=['accuracy'])

checkpointer = ModelCheckpoint(filepath='img_model_sem_data_augmentation.
    ↳weights.best.keras',
                                verbose=1,
                                save_best_only=True)

early_stop = EarlyStopping(monitor='val_loss',
                            patience=10,
                            restore_best_weights=True,
                            mode='min')

```

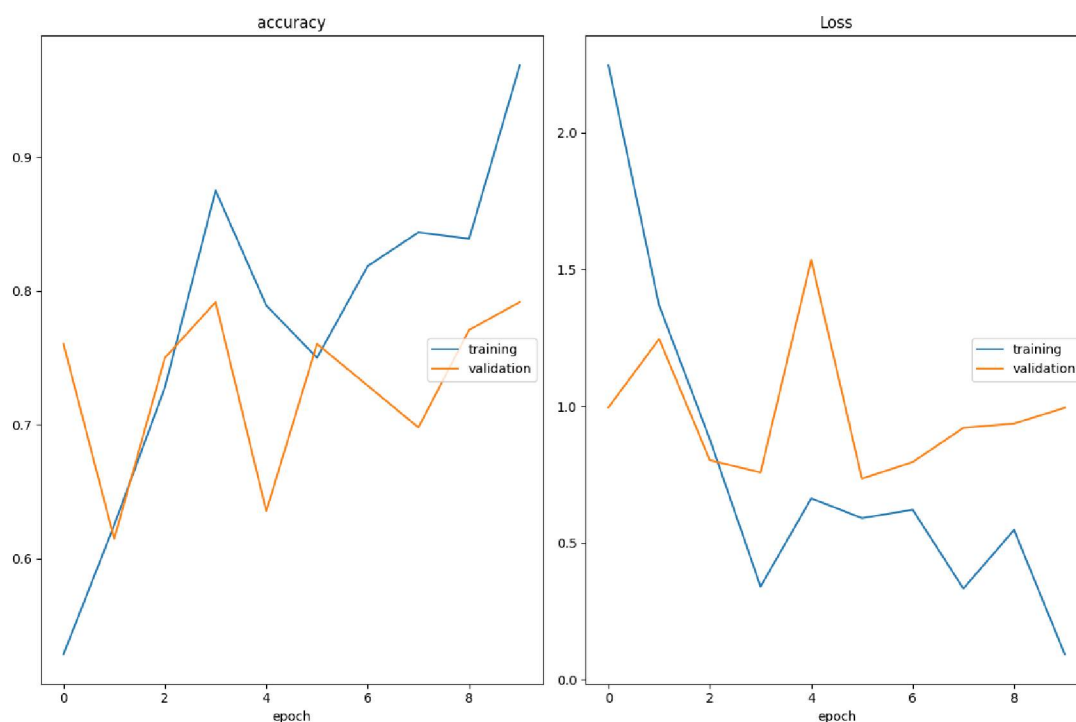
CPU times: user 6.1 ms, sys: 3 µs, total: 6.1 ms

Wall time: 5.75 ms

```

[ ]: history_t1 = model_sem_augment.fit(training,
    epochs=n_epochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=validgen,
    validation_steps=val_steps,
    callbacks=[checkpointer, PlotLossesKeras()],
    verbose=True)

```



```

accuracy
  training      (min: 0.528, max: 0.969, cur: 0.
↳969)
  validation    (min: 0.615, max: 0.792, cur: 0.
↳792)
Loss
  training      (min: 0.093, max: 2.247, cur: 0.
↳093)
  validation    (min: 0.735, max: 1.535, cur: 0.
↳995)

```

```

14/14          1s 95ms/step -
accuracy: 0.9688 - loss: 0.0934 - val_accuracy: 0.7917 - val_loss: 0.9947

```

```

####VGG16 com data augmentation

```

```

[ ]: from tensorflow.keras.applications.vgg16 import preprocess_input #
↳Import VGG16 preprocess_input

def create_vgg16_augmentation_generators(train_df, valid_df, test_df):

    train_generator = ImageDataGenerator(

```

```

rotation_range=90,
brightness_range=[0.1, 0.7],
width_shift_range=0.5,
height_shift_range=0.5,
horizontal_flip=True,
vertical_flip=True,
preprocessing_function=preprocess_input)

→# Use VGG16 preprocess_input

valid_generator =
→ImageDataGenerator(preprocessing_function=preprocess_input) # Use
→VGG16 preprocess_input

test_generator =
→ImageDataGenerator(preprocessing_function=preprocess_input) # Use
→VGG16 preprocess_input

traingen = train_generator.flow_from_dataframe(
    train_df,
    x_col='filename',
    y_col='class',
    target_size=(224,224),
    batch_size=batch_size,
    class_mode='categorical',
    classes=class_subset,
    shuffle=True,
    seed=42)

validgen = valid_generator.flow_from_dataframe(
    valid_df,
    x_col='filename',
    y_col='class',
    target_size=(224,224),
    batch_size=batch_size,
    class_mode='categorical',
    classes=class_subset,
    shuffle=True,
    seed=42)

testgen = test_generator.flow_from_dataframe(test_df,
    target_size=(224,224),
    batch_size=batch_size,
    class_mode='categorical',

```

```

        classes=class_subset,
        shuffle=False,
        seed=42)

    return traingen, validgen, testgen

```

```

[ ]: from tensorflow.keras.applications.vgg16 import preprocess_input #
    ↳ Import VGG16 preprocess_input
    from keras.applications.vgg16 import VGG16 # Import VGG16 again for
    ↳ clarity

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

    # treinar os pesos existentes
    for layer in vgg_augmentation.layers:
        layer.trainable = False

    x_vgg = Flatten()(vgg_augmentation.output)

    # camada de classificação com as 04 classes utilizadas
    prediction = Dense(4, activation='softmax')(x_vgg)

    # Criação do Objeto Modelo
    vgg_augmentation = Model(inputs=vgg_augmentation.input,
    ↳ outputs=prediction)

    vgg_augmentation.summary()

```

Model: "functional_6"

Layer (type)	Output Shape	Param #
input_layer_7 (InputLayer)	(None, 224, 224, 3)	0
...		
dense_3 (Dense)	(None, 4)	100,356

Total params: 14,815,044 (56.51 MB)

Trainable params: 100,356 (392.02 KB)

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

```
[ ]: traingen, validgen, testgen =   
      ↪ create_vgg16_augmentation_generators(train_df, valid_df, test_df)
```

Found 473 validated image filenames belonging to 4 classes.

Found 120 validated image filenames belonging to 4 classes.

Found 371 validated image filenames belonging to 4 classes.

```
[ ]: %%time
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras
from tensorflow.keras.utils import to_categorical

steps_per_epoch = traingen.samples // batch_size
val_steps = validgen.samples // batch_size

n_epochs = 10

optimizer = RMSprop(learning_rate=0.0001)

vgg_augmentation.compile(loss='categorical_crossentropy',   
      ↪ optimizer=optimizer, metrics=['accuracy'])

# Treinamento do Modelo

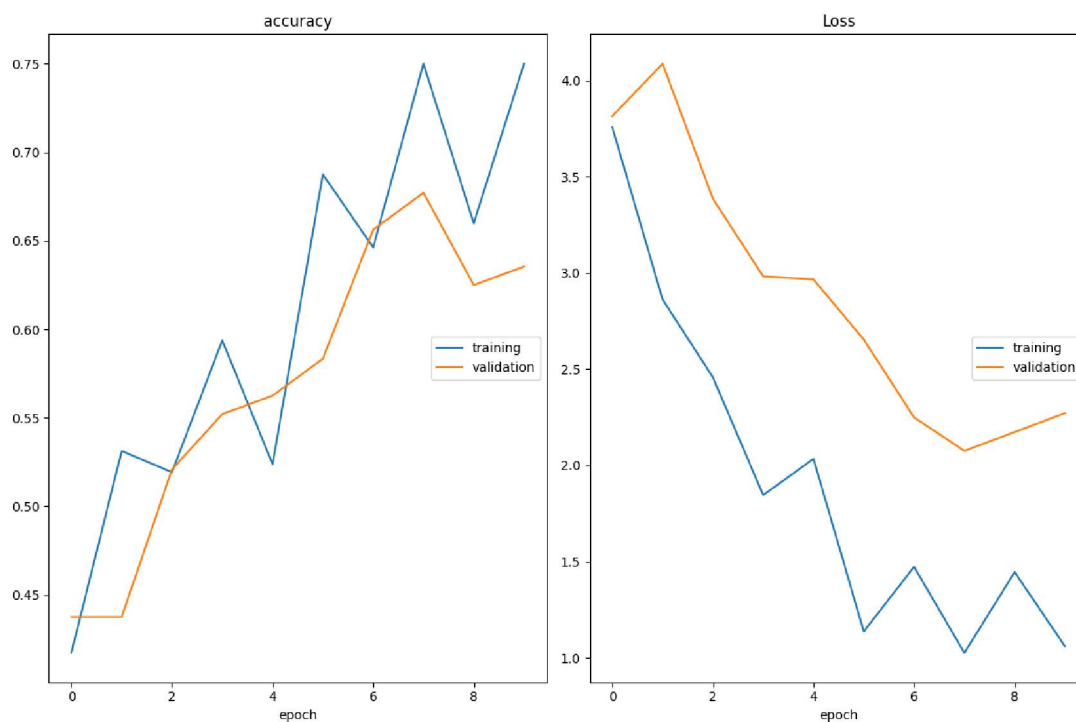
# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer = ModelCheckpoint(filepath='img_vgg_augmentation.weights.   
      ↪ best.keras',
                                verbose=1,
                                save_best_only=True)

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

CPU times: user 6.38 ms, sys: 0 ns, total: 6.38 ms

Wall time: 5.91 ms

```
[ ]: history_0 = vgg_augmentation.fit(training,
                                     epochs=n_epochs,
                                     steps_per_epoch=steps_per_epoch,
                                     validation_data=validation_data,
                                     validation_steps=val_steps,
                                     callbacks=[checkpointer, PlotLossesKeras()],
                                     verbose=True)
```



```
accuracy
  training      (min: 0.417, max: 0.750, cur: 0.750)
  validation    (min: 0.438, max: 0.677, cur: 0.635)
Loss
  training      (min: 1.026, max: 3.757, cur: 1.060)
  validation    (min: 2.076, max: 4.087, cur: 2.272)
```

14/14 2s 104ms/step -
 accuracy: 0.7500 - loss: 1.0600 - val_accuracy: 0.6354 - val_loss: 2.2717
 #####VGG16 Sem data augmentation

```
[ ]: from tensorflow.keras.applications.vgg16 import preprocess_input #
      ↳ Import VGG16 preprocess_input
def create_vgg16_sem_augmentation_generators(train_df, valid_df):
    # Create training data generator
    train_sem_augmentation_gen =
      ↳ ImageDataGenerator(preprocessing_function=preprocess_input) # Use
      ↳ VGG16 preprocess_input
    valid_sem_augmentation_gen =
      ↳ ImageDataGenerator(preprocessing_function=preprocess_input) # Use
      ↳ VGG16 preprocess_input

    traingen = train_sem_augmentation_gen.flow_from_dataframe(
        train_df,
        x_col='filename',
        y_col='class',
        target_size=(224,224),
        batch_size=64,
        class_mode='categorical',
        classes=class_subset,
        shuffle=True,
        seed=42)

    validgen = valid_sem_augmentation_gen.flow_from_dataframe(
        valid_df,
        x_col='filename',
        y_col='class',
        target_size=(224,224),
        batch_size=64,
        class_mode='categorical',
        classes=class_subset,
        shuffle=True,
        seed=42)

    return traingen, validgen

[ ]: from keras.applications.vgg16 import VGG16 # Import VGG16 again for
      ↳ clarity
```

```
vgg_sem_augment = VGG16(input_shape=(224,224,3), weights='imagenet',
    ↳include_top=False)
```

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

```
[ ]: x_vgg_sem_augment = Flatten()(vgg_sem_augment.output)

# camada de classificação com as 04 classes utilizadas
prediction = Dense(4, activation='softmax')(x_vgg_sem_augment)

# Criação do Objeto Modelo
vgg_sem_augment = Model(inputs=vgg_sem_augment.input, outputs=prediction)
```

```
[ ]: sem_aug_traingen, sem_aug_validgen=
    ↳create_vgg16_sem_augmentation_generators(train_df, valid_df)
```

Found 473 validated image filenames belonging to 4 classes.

Found 120 validated image filenames belonging to 4 classes.

```
[ ]: %%time
# Otimizador propagação da Raiz quadrada da média ao quadrado (Root Mean
    ↳Squared Propagation)
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras
from tensorflow.keras.utils import to_categorical

steps_per_epoch = sem_aug_traingen.samples // batch_size
val_steps = sem_aug_validgen.samples // batch_size

n_epochs = 10

optimizer = RMSprop(learning_rate=0.0001)

vgg_sem_augment.compile(loss='categorical_crossentropy',
    ↳optimizer=optimizer, metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer = ModelCheckpoint(filepath='img_model_vgg_sem_augment.
    ↳weights.best.keras',
                                verbose=1,
```

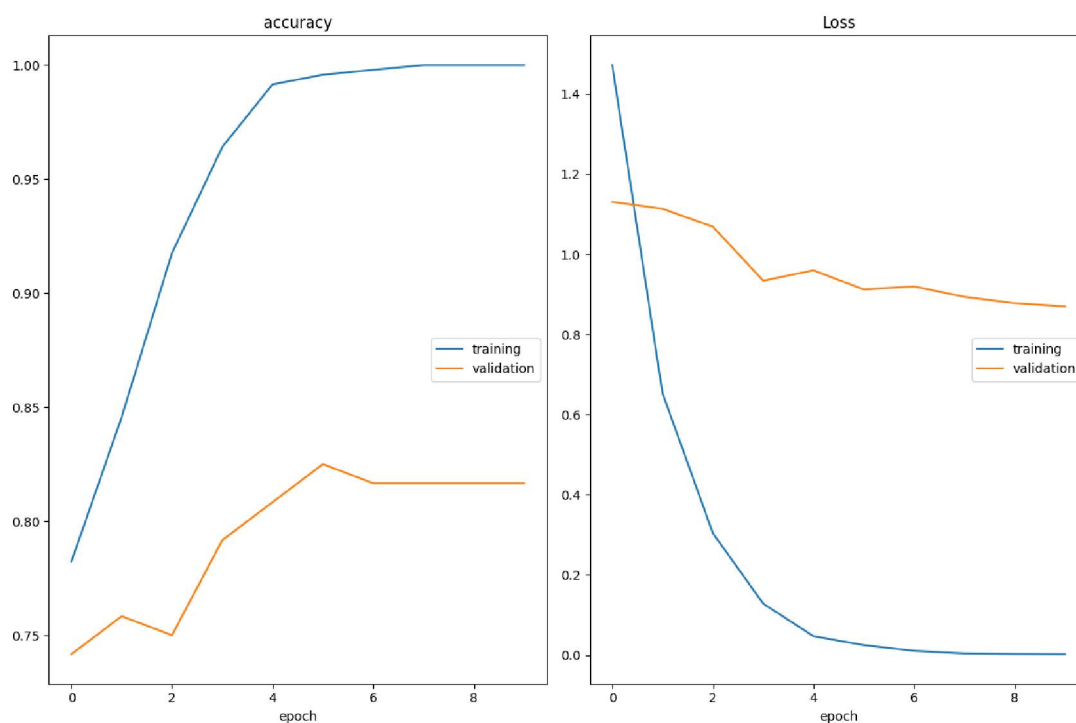
```
save_best_only=True)
```

```
# Para o treinamento para prevenir o overfitting
early_stop = EarlyStopping(monitor='val_loss',
                             patience=10,
                             restore_best_weights=True,
                             mode='min')
```

CPU times: user 3.62 ms, sys: 821 µs, total: 4.44 ms

Wall time: 4.46 ms

```
[ ]: history_0 = vgg_augmentation.fit(sem_aug_traingen,
                                     epochs=n_epochs,
                                     steps_per_epoch=steps_per_epoch,
                                     validation_data=sem_aug_validgen,
                                     validation_steps=val_steps,
                                     callbacks=[checkpointer, PlotLossesKeras()],
                                     verbose=True)
```



accuracy

```

    training          (min:    0.782, max:    1.000, cur:    1.
    000)
    validation         (min:    0.742, max:    0.825, cur:    0.
    817)
Loss
    training          (min:    0.002, max:    1.472, cur:    0.
    002)
    validation         (min:    0.870, max:    1.131, cur:    0.
    870)

```

```

14/14          7s 447ms/step -
accuracy: 1.0000 - loss: 0.0018 - val_accuracy: 0.8167 - val_loss: 0.8698

```

```
[ ]: test_df.head()
```

```

[ ]:
      filename class
0  /content/Test_4cl_amostra/1/39_HER2_22299.png      1
1  /content/Test_4cl_amostra/1/39_HER2_25631.png      1
2  /content/Test_4cl_amostra/1/39_HER2_24516.png      1
3  /content/Test_4cl_amostra/1/34_HER2_12127.png      1
4  /content/Test_4cl_amostra/1/34_HER2_17118.png      1

```

###Q2T3. Aplique os modelos treinados nas imagens da base de Teste

Predições dos Modelos com Data Augmentation Resnet50

```
[ ]: true_classes = testgen.classes
     true_classes2 = test_df['class']
```

```
[ ]: if np.array_equal(true_classes, true_classes2):
      print("true_classes é igual a true_classes2")
      else:
      print("true_classes não é igual a true_classes2")

```

true_classes não é igual a true_classes2

```
[ ]: from sklearn.metrics import accuracy_score, recall_score,
      precision_score, f1_score, confusion_matrix
      import pandas as pd
      import numpy as np

      # Função para calcular a especificidade em problemas multiclases
      def specificity_score_multiclass(y_true, y_pred):
          cm = confusion_matrix(y_true, y_pred)
          # Inicializa a lista para armazenar a especificidade por classe
          specificity_per_class = []

```



```

    # Itera sobre cada classe para calcular a especificidade
    for i in range(len(cm)):
        tn = np.sum(np.delete(np.delete(cm, i, axis=0), i, axis=1)) #
        ↪ Remove a linha e coluna da classe 'i'
        fp = np.sum(cm[:, i]) - cm[i, i] # Falsos positivos para a
        ↪ classe 'i'
        specificity = tn / (tn + fp) if (tn + fp) > 0 else 0 # Evitar
        ↪ divisão por zero
        specificity_per_class.append(specificity)
    # Retorna a média da especificidade sobre todas as classes
    return np.mean(specificity_per_class)

# Função para calcular todas as métricas
def calculate_metrics(true_classes, pred_classes, model_name):
    acc = accuracy_score(true_classes, pred_classes)
    sens = recall_score(true_classes, pred_classes, average='macro') #
    ↪ Sensibilidade média por classe
    spec = specificity_score_multiclass(true_classes, pred_classes) #
    ↪ Especificidade média por classe
    f1 = f1_score(true_classes, pred_classes, average='macro') #
    ↪ F1-Score médio por classe
    return [model_name, acc, sens, spec, f1]

```

```

[ ]: from sklearn.metrics import accuracy_score, recall_score,
    ↪ precision_score, f1_score, confusion_matrix
import pandas as pd
import numpy as np

# Carregar os pesos dos modelos
model_0.load_weights('img_model_0.weights.best.keras')

# Gerar as previsões
preds_0 = model_0.predict(testgen)
pred_classes_0 = np.argmax(preds_0, axis=1)

# Calcular as métricas para cada modelo
metrics_0 = calculate_metrics(true_classes, pred_classes_0, "Modelo_
    ↪ ResNet50 com Data Augmentation Treinado do Zero")

# Criar DataFrame com os resultados
df_metrics = pd.DataFrame([metrics_0],
    ↪ columns=["Modelo", "Acurácia", "Sensibilidade",
    ↪ "Especificidade", "F1-Score"])

```



```
# Ordenar pelo maior F1-Score
df_metrics = df_metrics.sort_values(by="F1-Score", ascending=False)

# Exibir o DataFrame ordenado
print(df_metrics)
```

```
/usr/local/lib/python3.11/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super()`.
↳ __init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
```

```
self._warn_if_super_not_called()
```

```
12/12          13s 643ms/step
```

```
Modelo  Acurácia
```

```
↳ Sensibilidade \
```

```
0  Modelo ResNet50 com Data Augmentation Treinado...  0.789757      0.783333
```

```
    Especificidade  F1-Score
```

```
0          0.929336  0.761894
```

```
[ ]: df_metrics
```

```
[ ]:          Modelo  Acurácia
```

```
↳ Sensibilidade \
```

```
0  Modelo ResNet50 com Data Augmentation Treinado...  0.606469      0.
```

```
↳ 611689
```

```
    Especificidade  F1-Score
```

```
0          0.869962  0.585786
```

CNN VGG16 com Data Augmentation

```
[ ]: from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Definir o tamanho da imagem (mesmo que nos geradores)
img_size = (224, 224)
num_classes = len(class_subset) # Número de classes a serem
↳ classificadas
```

```

# Carregar o modelo base VGG16 pré-treinado no ImageNet
base_model = VGG16(weights='imagenet', include_top=False,
    ↳input_shape=(img_size[0], img_size[1], 3))

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

# Adicionar camadas de classificação no topo
x = Flatten()(base_model.output) # Achata as ativações das camadas
    ↳convolucionais
x = Dense(512, activation='relu')(x) # Primeira camada totalmente
    ↳conectada
x = Dropout(0.5)(x) # Dropout para evitar overfitting
x = Dense(num_classes, activation='softmax')(x) # Camada de saída com
    ↳número de classes

# Criar o modelo final
model = Model(inputs=base_model.input, outputs=x)

# Compilar o modelo
model.compile(optimizer=Adam(learning_rate=1e-4),
    ↳loss='categorical_crossentropy', metrics=['accuracy'])

# Definir callbacks: checkpoint para salvar o melhor modelo e early
    ↳stopping
checkpoint = ModelCheckpoint('img_vgg_augmentation.keras',
    ↳monitor='val_loss', save_best_only=True, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)

# Treinar o modelo
history = model.fit(
    traingen,
    validation_data=validgen,
    epochs=10,
    steps_per_epoch=traingen.samples // traingen.batch_size,
    validation_steps=validgen.samples // validgen.batch_size,
    callbacks=[checkpoint, early_stopping]
)

```

Epoch 1/10

14/14 0s 793ms/step -

accuracy: 0.2943 - loss: 6.8849

```

# Converter as previsões em classes
pred_classes = np.argmax(preds, axis=1)

# Classes verdadeiras
true_classes = testgen.classes

# Verificar se o número de previsões corresponde ao número de amostras
↳no conjunto de teste
print(f"Número de classes verdadeiras: {len(true_classes)}")
print(f"Número de classes previstas: {len(pred_classes)}")

# Calcular métricas
acc = accuracy_score(true_classes, pred_classes)
sens = recall_score(true_classes, pred_classes, average='macro') #↳
↳Sensibilidade média
spec = specificity_score_multiclass(true_classes, pred_classes) #↳
↳Especificidade média
f1 = f1_score(true_classes, pred_classes, average='macro') # F1-Score↳
↳médio

```

```

12/12          12s 1s/step
Número de classes verdadeiras: 371
Número de classes previstas: 371

```

```

[ ]: # Criar DataFrame com as métricas
df_metrics1 = pd.DataFrame({
    'Modelo': "VGG16 Data Augmentation",
    'Acurácia': [acc],
    'Sensibilidade': [sens],
    'Especificidade': [spec],
    'F1-Score': [f1]
})

# Exibir o DataFrame com os resultados
print(df_metrics1)

```

```

          Modelo  Acurácia  Sensibilidade  Especificidade  ↳
↳F1-Score
0  VGG16 Data Augmentation  0.654987          0.644444          0.883113  0.
↳588581

```

```

[ ]: # prompt: unir os dataframes df_metrics e df_metrics1 e ordenar pelo
↳maior F1 score

```

```
# Concatenar os DataFrames df_metrics e df_metrics1
df_combined = pd.concat([df_metrics, df_metrics1], ignore_index=True)

# Ordenar o DataFrame combinado pelo maior F1-Score
df_combined = df_combined.sort_values(by='F1-Score', ascending=False)

# Exibir o DataFrame ordenado
print(df_combined)
```

	Modelo	Acurácia
↳ Sensibilidade \		
0	Modelo ResNet50 com Data Augmentation Treinado...	0.789757 0.783333
1	VGG16 Data Augmentation	0.654987 0.
↳ 644444		

	Especificidade	F1-Score
0	0.929336	0.761894
1	0.883113	0.588581

Resnet50 sem Data augmentation

```
[ ]: import pandas as pd
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping
from livelossplot import PlotLossesKeras

# Parâmetros
img_size = (224, 224) # Tamanho das imagens
batch_size = 32
n_epochs = 10

# Criar geradores de dados
train_datagen = ImageDataGenerator(rescale=1./255) # Normalização
valid_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_dataframe(
    train_df,
    x_col='filename',
    y_col='class',
    target_size=img_size,
```

```

        batch_size=batch_size,
        class_mode='categorical', # Para múltiplas classes
        shuffle=True
    )

valid_generator = valid_datagen.flow_from_dataframe(
    valid_df,
    x_col='filename',
    y_col='class',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

test_generator = test_datagen.flow_from_dataframe(
    test_df,
    x_col='filename',
    y_col='class',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

# Carregar o modelo ResNet50
base_model = ResNet50(weights='imagenet', include_top=False,
    ↪input_shape=(img_size[0], img_size[1], 3))

# Congelar as camadas do modelo base
for layer in base_model.layers:
    layer.trainable = False

# Adicionar camadas personalizadas
x = Flatten()(base_model.output)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)

# Definir a camada de saída com o número correto de classes
x = Dense(len(train_generator.class_indices), activation='softmax')(x)

# Criar o modelo final
model = Model(inputs=base_model.input, outputs=x)

```



```

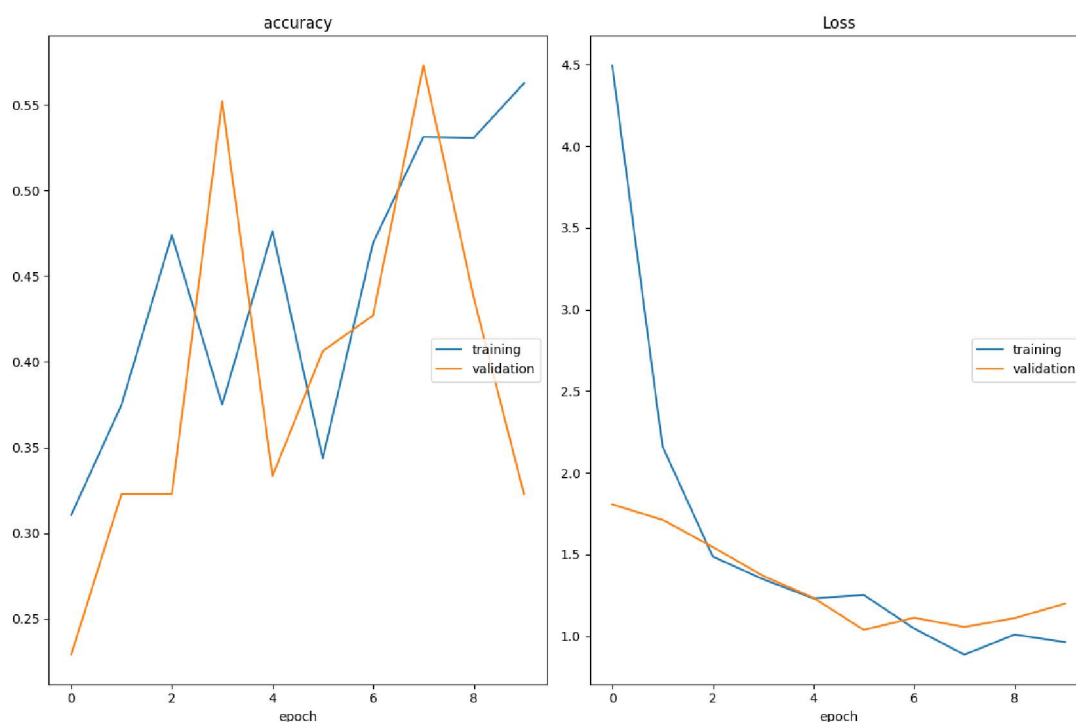
# Compilar o modelo
optimizer = RMSprop(learning_rate=0.0001)
model.compile(loss='categorical_crossentropy', optimizer=optimizer,
    metrics=['accuracy'])

# Definir callbacks
checkpointer = ModelCheckpoint(filepath='resnet50_best_weights.keras',
    verbose=1, save_best_only=True)
early_stop = EarlyStopping(monitor='val_loss', patience=10,
    restore_best_weights=True, mode='min')

# Ajustar o modelo
steps_per_epoch = train_generator.samples // batch_size
val_steps = valid_generator.samples // batch_size

history = model.fit(
    train_generator,
    epochs=n_epochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=valid_generator,
    validation_steps=val_steps,
    callbacks=[checkpointer, PlotLossesKeras()],
    verbose=1
)

```



```

accuracy
    training          (min:    0.311, max:    0.562, cur:    0.
↪562)
    validation        (min:    0.229, max:    0.573, cur:    0.
↪323)
Loss
    training          (min:    0.887, max:    4.495, cur:    0.
↪963)
    validation        (min:    1.038, max:    1.807, cur:    1.
↪198)

```

14/14 1s 103ms/step -

accuracy: 0.5625 - loss: 0.9632 - val_accuracy: 0.3229 - val_loss: 1.1984

```

[ ]: from sklearn.metrics import accuracy_score, recall_score,
    ↪precision_score, f1_score, confusion_matrix
import pandas as pd
import numpy as np

# Gerar as previsões
preds = model.predict(testgen)
pred_classes = np.argmax(preds, axis=1)

# Calcular as métricas para cada modelo
metrics = calculate_metrics(true_classes, pred_classes, "Modelo ResNet50_
↪sem Data Augmentation")

# Criar DataFrame com os resultados
df_metrics3 = pd.DataFrame([metrics],
                           columns=["Modelo", "Acurácia", "Sensibilidade",
↪"Especificidade", "F1-Score"])

# Ordenar pelo maior F1-Score
df_metrics3 = df_metrics3.sort_values(by="F1-Score", ascending=False)

# Exibir o DataFrame ordenado
print(df_metrics3)

```

12/12 13s 730ms/step

Modelo Acurácia Sensibilidade \

0 Modelo ResNet50 sem Data Augmentation 0.250674 0.258333

Especificidade F1-Score

0 0.752669 0.123172

CNN VGG16 Sem Data Augmentation

```
[ ]: from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

# Definir o tamanho da imagem (mesmo que nos geradores)
img_size = (224, 224)
num_classes = len(class_subset) # Número de classes a serem
    ↳classificadas

# Carregar o modelo base VGG16 pré-treinado no ImageNet
base_model = VGG16(weights='imagenet', include_top=False,
    ↳input_shape=(img_size[0], img_size[1], 3))

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

# Adicionar camadas de classificação no topo
x = Flatten()(base_model.output) # Achata as ativações das camadas
    ↳convolucionais
x = Dense(512, activation='relu')(x) # Primeira camada totalmente
    ↳conectada
x = Dropout(0.5)(x) # Dropout para evitar overfitting
x = Dense(num_classes, activation='softmax')(x) # Camada de saída com
    ↳número de classes

# Criar o modelo final
model = Model(inputs=base_model.input, outputs=x)

# Compilar o modelo
model.compile(optimizer=Adam(learning_rate=1e-4),
    ↳loss='categorical_crossentropy', metrics=['accuracy'])

# Definir callbacks: checkpoint para salvar o melhor modelo e early
    ↳stopping
```

```

checkpoint = ModelCheckpoint('img_model_vgg_sem_augment.keras',
    monitor='val_loss', save_best_only=True, verbose=1)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)

# Treinar o modelo
history = model.fit(
    traingen,
    validation_data=validgen,
    epochs=10,
    steps_per_epoch=traingen.samples // traingen.batch_size,
    validation_steps=validgen.samples // validgen.batch_size,
    callbacks=[checkpoint, early_stopping]
)

```

Epoch 1/10

14/14 0s 777ms/step -

accuracy: 0.3385 - loss: 5.9763

Epoch 1: val_loss improved from inf to 6.20825, saving model to

img_model_vgg_sem_augment.keras

14/14 17s 1s/step -

accuracy: 0.3426 - loss: 5.9117 - val_accuracy: 0.4479 - val_loss: 6.2083

Epoch 2/10

1/14 2s 166ms/step -

accuracy: 0.5000 - loss: 3.6972

/usr/local/lib/python3.11/dist-

packages/keras/src/trainers/epoch_iterator.py:107: UserWarning: Your input

ran

out of data; interrupting training. Make sure that your dataset or

generator can

generate at least `steps\_per\_epoch \* epochs` batches. You may need to use

the

`.repeat()` function when building your dataset.

self._interrupted_warning()

Epoch 2: val_loss did not improve from 6.20825

14/14 3s 253ms/step -

accuracy: 0.5000 - loss: 3.6972 - val_accuracy: 0.4792 - val_loss: 6.2454

Epoch 3/10

14/14 0s 668ms/step -

accuracy: 0.5620 - loss: 3.0239

Epoch 3: val_loss improved from 6.20825 to 4.18646, saving model to

img_model_vgg_sem_augment.keras

14/14 24s 2s/step -

```
img_model_vgg_sem_augment.keras
14/14          5s 350ms/step -
accuracy: 0.6875 - loss: 0.6068 - val_accuracy: 0.7188 - val_loss: 2.2371
```

```
[ ]: from sklearn.metrics import accuracy_score, recall_score, \
      precision_score, f1_score, confusion_matrix
import pandas as pd
import numpy as np

# Avaliar o modelo no conjunto de teste
test_loss, test_acc = model.evaluate(testgen, steps=testgen.samples // \
      testgen.batch_size)
print(f'Acurácia no conjunto de teste: {test_acc * 100:.2f}%')
```

```
11/11          2s 199ms/step -
accuracy: 0.4900 - loss: 3.9278
Acurácia no conjunto de teste: 71.31%
```

Q2T4 Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão

```
[ ]: # Gerar previsões no conjunto de teste (sem o parâmetro 'steps' para
      ↳prever todas as amostras)
      preds = model.predict(testgen)

      # Converter as previsões em classes
      pred_classes = np.argmax(preds, axis=1)

      # Classes verdadeiras
      true_classes = testgen.classes

      # Verificar se o número de previsões corresponde ao número de amostras
      ↳no conjunto de teste
      print(f"Número de classes verdadeiras: {len(true_classes)}")
      print(f"Número de classes previstas: {len(pred_classes)}")

      # Calcular métricas
      acc = accuracy_score(true_classes, pred_classes)
      sens = recall_score(true_classes, pred_classes, average='macro') #
      ↳Sensibilidade média
      spec = specificity_score_multiclass(true_classes, pred_classes) #
      ↳Especificidade média
      f1 = f1_score(true_classes, pred_classes, average='macro') # F1-Score
      ↳médio
```

```
12/12          5s 327ms/step
Número de classes verdadeiras: 371
Número de classes previstas: 371
```

```
[ ]: # Criar DataFrame com as métricas
      df_metrics4 = pd.DataFrame({
          'Modelo': "VGG16 Sem Data Augmentation",
          'Acurácia': [acc],
          'Sensibilidade': [sens],
          'Especificidade': [spec],
          'F1-Score': [f1]
      })

      # Exibir o DataFrame com os resultados
      print(df_metrics4)
```

	Modelo	Acurácia	Sensibilidade	Especificidade	\
0	VGG16 Sem Data Augmentation	0.727763	0.719444	0.907714	
	F1-Score				
0	0.675051				

```
[ ]: # Concatenar os DataFrames
df_combined = pd.concat([df_metrics,
    ↪df_metrics1,df_metrics3,df_metrics4], ignore_index=True)

# Ordenar o DataFrame combinado pelo maior F1-Score
df_combined = df_combined.sort_values(by='F1-Score', ascending=False)

# Exibir o DataFrame ordenado
print(df_combined)
```

	Modelo	Acurácia	
↪Sensibilidade \			
0	Modelo ResNet50 com Data Augmentation Treinado...	0.789757	0.783333
3	VGG16 Sem Data Augmentation	0.727763	0.
↪719444			
1	VGG16 Data Augmentation	0.654987	0.
↪644444			
2	Modelo ResNet50 sem Data Augmentation	0.250674	0.
↪258333			

	Especificidade	F1-Score
0	0.929336	0.761894
3	0.907714	0.675051
1	0.883113	0.588581
2	0.752669	0.123172

Q2T5 5. Indique qual modelo dá o melhor o resultado e a métrica utilizada

RESULTADO Q2: O modelo que tem maior F1 score é o modelo ResNet50 com data augmentation, ele tem os melhores resultados de sensibilidade, ou seja, identifica corretamente 78,33% do total de casos verdadeiros de câncer de mama do período de teste; bem com é o modelo com melhor especificidade, ou seja, identifica corretamente 92,93% dos casos verdadeiros de pacientes sem a doença.

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

Parâmetros para elaboração do Trabalho:

1. Relevância Ética: O trabalho deve abordar questões éticas significativas relacionadas ao uso da inteligência artificial, especialmente no contexto do ChatGPT. Os alunos devem identificar dilemas éticos relevantes e explorar como esses dilemas afetam diferentes partes interessadas, como usuários, desenvolvedores e a sociedade em geral.

2. Análise Crítica: Os alunos devem realizar uma análise crítica das implicações éticas do uso do ChatGPT em estudos de caso específicos. Eles devem examinar como o algoritmo pode influenciar a disseminação de informações, a privacidade dos usuários e a tomada de decisões éticas. Além disso, devem considerar possíveis vieses algorítmicos, discriminação e questões de responsabilidade.

3. Soluções Responsáveis: Além de identificar os desafios éticos, os alunos devem propor soluções responsáveis e éticas para lidar com esses dilemas. Isso pode incluir sugestões para políticas, regulamentações ou práticas de design que promovam o uso responsável da inteligência artificial. Eles devem considerar como essas soluções podem equilibrar os interesses de diferentes partes interessadas e promover valores éticos fundamentais, como transparência, justiça e privacidade.

4. Colaboração e Discussão: O trabalho deve envolver discussões em grupo e colaboração entre os alunos. Eles devem compartilhar ideias, debater diferentes pontos de vista e chegar a conclusões informadas através do diálogo e da reflexão mútua. O estudo de caso do ChatGPT pode servir como um ponto de partida para essas discussões, incentivando os alunos a aplicar conceitos éticos e legais aprendidos ao analisar um caso concreto.

5. Limite de Palavras: O trabalho terá um limite de 6 a 10 páginas teria aproximadamente entre 1500 e 3000 palavras.

6. Estruturação Adequada: O trabalho siga uma estrutura adequada, incluindo introdução, desenvolvimento e conclusão. Cada seção deve ocupar uma parte proporcional do total de páginas, com a introdução e a conclusão ocupando menos espaço do que o desenvolvimento.

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

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

9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação.

Devese seguir o seguinte template de arquivo:

<https://bibliotecas.ufpr.br/wpcontent/uploads/2022/03/template-artigo-de-periodico.docx>

B - RESOLUÇÃO

Estudo de caso: Implicações Éticas do uso do ChatGPT

Nome dos autores:

Leandro Motta Marinho de Oliveira

Luan Silva Gomes

Paulo Silas Brasileiro Luiz

Priscila Lie Kawamura

Vinicius Natan Silva e Sou

RESUMO

Neste trabalho será discutido as implicações éticas do uso do modelo de inteligência artificial ChatGPT, uma ferramenta que emergiu como um fenômeno mundial, alcançando o público em geral com suas capacidades generativas capazes de criar textos de praticamente qualquer assunto e com grande potencial de disseminar opiniões preconceituosas, informações falsas entre outras utilizações para prejudicar a sociedade e indivíduo. Dentro deste contexto serão discutidos os desafios éticos emergentes relacionados ao seu desenvolvimento e aplicação, destacando questões cruciais como privacidade, disseminação de informações e tomada de decisões éticas. A análise crítica aborda também possíveis vieses algorítmicos, discriminação e responsabilidades associadas ao uso desta tecnologia. Este estudo visa fornecer uma compreensão aprofundada dos dilemas éticos contemporâneos na era da inteligência artificial.

Palavras-chave: ChatGPT. Questões éticas. Ferramenta. Inteligência Artificial.

ABSTRACT

In this paper, we will discuss the ethical implications of using the ChatGPT artificial intelligence model, a tool that has emerged as a global phenomenon, reaching

the general public with its generative capabilities capable of creating texts on virtually any subject and with significant potential to disseminate prejudiced opinions, false information, and other uses harmful to society and individuals. Within this context, we will examine the emerging ethical challenges related to its development and application, highlighting critical issues such as privacy, information dissemination, and ethical decision-making. The critical analysis will also address potential algorithmic biases, discrimination, and responsibilities associated with the use of this technology. This study aims to provide an in-depth understanding of contemporary ethical dilemmas in the era of artificial intelligence.

Keywords: ChatGPT. Ethical issues. Tool. Artificial Intelligence.

1 INTRODUÇÃO

Hoje em dia, com o avanço tecnológico, é possível usar a linguagem natural para conversar com uma máquina e obter respostas mais elaboradas. Com o uso da inteligência artificial, abriu-se diversas possibilidades de uso em diversas áreas que podem estar relacionadas a tecnologia ou não.

Para uso pessoal, profissional ou acadêmicos, muitos adotaram o famoso ChatGPT. As infinitas possibilidades de uso do ChatGPT para auxiliar-nos depende também de o usuário usar de forma clara e precisa para conseguir um resultado esperado e preciso.

O ChatGPT trouxe diversos benefícios para a sociedade, mas há questões que é necessário ter cautela ao utilizar qualquer ferramenta disponíveis no mercado. Não apenas ao utilizar, mas também ao desenvolver uma ferramenta para o público para violar questões ética. Por isso, neste trabalho iremos discutir essas questões éticas por trás desta ferramenta que se tornou famosa no mundo todo.

2 DESENVOLVIMENTO

2.1 CHATGPT

O ChatGPT é uma ferramenta que pode ser usado para diversos fins. Aqui temos um resumo que o próprio ChatGPT escreveu sobre a si mesmo:

“O ChatGPT é um modelo de linguagem desenvolvido pela OpenAI que utiliza inteligência artificial para gerar respostas em conversas com base no contexto fornecido. Ele é treinado em uma grande quantidade de dados textuais para entender e responder a uma ampla variedade de perguntas e comandos. O ChatGPT é capaz

de manter conversas fluentes e contextuais com os usuários, simulando uma interação humana em vários tópicos. Ele pode ser utilizado em uma variedade de aplicações, desde assistentes virtuais até suporte ao cliente e geração de conteúdo. O ChatGPT tem sido elogiado por sua capacidade de gerar respostas relevantes e coerentes, tornando-o uma ferramenta útil para comunicação e interação online.” (CHATGPT,2024)

Como mencionado no parágrafo anterior, o ChatGPT é capaz de realizar diversas ações como ter conversas fluentes sobre diversos tópicos, gerar conteúdo, gerar partes de código para desenvolvedores e para diversas outras aplicações devido a sua arquitetura chamada GPT (Generative Pretrained Transformer).

A Inteligência artificial trouxe para sociedade um avanço muito importante que impactou o dia a dia de diversas pessoas. E por envolver pessoas, esse progresso tecnológico trouxe diversas questões éticas e cuidados cruciais.

2.1.1 INTELIGÊNCIA ARTIFICIAL

Como visto no curso, há diversos tipos de inteligência artificial. No ChatGPT É usado a IA Generativa onde a resposta é formulada com base em informações de diversos sites disponibilizados na internet. Devido a isso, as informações que são aprendidas pelo ChatGPT têm como base a internet, e por isso é necessário cautela ao usar informações adquiridas dentro do ChatGPT para não correr o risco de plágio, respostas com viés preconceituoso ou a exposição de dados pessoais.

2.2 ÉTICA

A ética é um estudo necessários para o desenvolvimento de várias questões em relação ao que é considerado certo e errado, buscando entender e definir o que constitui de ações moralmente corretas e justas.

É essencial em diversas áreas incluindo a tecnologia para garantir que as ações e decisões sempre sejam justas e benéficas para a sociedade. No contexto da Inteligência artificial é algo crucial considerar as questões éticas relacionado a privacidade, segurança e equidade para nos guiar a agir de maneira responsável e justa para promover a convivência harmoniosa e equitativa na sociedade.

2.2.1 ÉTICA NA TECNOLOGIA E NA IA

Hoje em dia, mesmo antes do ChatGPT, já se existiam diversas leis que se aplicam as empresas e desenvolvedores no cuidado do uso de dados tanto públicos como pessoas. Os pontos principais são:

Privacidade de dados: Garantir que informações pessoais estejam protegidas e usadas de forma correta.

Responsabilidade Social: Desenvolver tecnologias que melhore a qualidade de vida das e pessoas e da sociedade considerando os impactos sociais e ambientais.

Transparência: O uso dos dados pessoas por empresas devem ser transparente. Para garantir essas práticas com a lei de proteção de dados LGPD (Lei Geral de Proteção de Dados) do Brasil

Inclusão e Diversidade: Promover a diversidade e a inclusão no desenvolvimento para garantir que as soluções não tenham viés causando a desigualdade.

Impacto Ambiental: Buscar soluções sustentáveis considerando os efeitos ambientais das tecnologias para minimizar os danos ao meio ambiente.

2.2.2 IA GENERATIVA, CHATGPT E ÉTICA

Durante o desenvolvimento e uso de ferramentas de IA Generativa igual o ChatGPT, como mencionado no tópico anterior, tem diversos fatores que é essencial ter devida a atenção para não violar questões éticas.

O uso de informações indevidas, podem trazer uma resposta com viés que acabe perpetuando estereótipos, preconceituoso e imoral durante uma pesquisa. É necessário o cuidado ao desenvolver este tipo de ferramenta para sempre ter como base, informações imparciais.

Uma empresa usar a IA Generativa para influenciar uma decisão a favor de outra empresa ou algo imoral mostra a falta de transparência e antiético da ferramenta desenvolvida sendo um dos pilares a transparência dos algoritmos

2.3 PRIVACIDADE

Os dados pessoais, a proteção dos dados e a segurança da privacidade são pontos importantes. Os sistemas de Inteligência Artificial devem ser projetados levando em conta a privacidade e segurança das informações do usuário, garantindo que

informações pessoais utilizadas durante o uso sejam tratadas com confidencialidade e não sejam utilizadas para fins não autorizados, respeitando a ética e as regulamentações existentes.

2.4 RESPONSABILIDADE

É importante que os desenvolvedores e usuários de IA sejam responsáveis por suas ações. Os desenvolvedores devem garantir que o GPT seja projetado e treinado de maneira ética, identificando e mitigando possíveis riscos como viés ou discriminação. Os usuários têm a responsabilidade de utilizar a ferramenta de maneira ética e responsável, não a utilizando para solicitações ilegais e criminosas.

2.5 TOMADA DE DECISÕES

IAs podem ser grandes aliadas na tomada de decisões, levando em conta a grande carga de informações que conseguem analisar e processar, tornando-se assim uma ferramenta valiosa para diversas áreas. Mas ainda devemos levar em conta que tudo isso é um grande modelo baseado em estatística, criado por humanos que podem conter vieses e preconceitos. Apesar de ser uma fonte valiosa de informações e uma grande ajuda na tomada de decisão, ainda devemos analisar as informações de maneira crítica, tornando-nos assim responsáveis por nossas decisões, levando em conta a ética e o bem-estar da sociedade como um todo e respeitando as regulamentações vigentes.

3 CONCLUSÃO

A IA generativa é um marco no progresso tecnológico, com isso é necessário combater os vieses desenvolvidos durante o treinamento desses dados. Como toda ferramenta, tem-se a necessidade de o desenvolvedor aprender, revisar, promover melhorias e se conscientizar das boas práticas da Inteligência e questões éticas. Com uso da própria IA usar ferramentas para ensinar a máquina detectar e corrigir os

problemas que podem infligir questões éticas. Na questão de privacidade, com o surgimento de novas ferramentas, serão necessárias o surgimento de novas leis como o exemplo da LGPD que é uma lei recente apesar do uso da tecnologia fazer parte do cotidiano da sociedade há alguns anos.

REFERÊNCIAS

DIDÁTICA TECH. ChatGPT e a ética da IA: discussão sobre os dilemas éticos. Disponível em: <https://didatica.tech/chatgpt-e-a-etica-da-ia-discussao-sobre-os-dilemas-eticos/>. Acesso em: 22 jun. 2024.

PEREIRA, Ana. Ética na tecnologia: saiba como funciona e como aplicá-la. Disponível em: <https://blog.casadodesenvolvedor.com.br/etica-na-tecnologia/>. Acesso em: 19 jun. 2024.

THOUGHTWORKS. Ética na tecnologia: do objetivo à prática. Disponível em: <https://www.thoughtworks.com/pt-br/perspectives/edition11-ethical-technology/article>. Acesso em: 19 jun. 2024.

MACKENZIE. Noções de ética no mundo digital. Disponível em: <https://blog.mackenzie.br/mercado-carreira/mercado-de-trabalho/nocoas-de-etica-no-mundo-digital-2/>. Acesso em: 17 jun. 2024.

OPENAI. ChatGPT: modelo de linguagem de grande escala. Disponível em: <https://chat.openai.com>. Acesso em: 17 jun. 2024.

De Souza, Fernanda. Ética e Inteligência Artificial (IA) para profissionais de tecnologia: navegando no mundo digital de forma responsável. Disponível em: <https://www.alura.com.br/artigos/etica-e-inteligencia-artificial>. Acesso em: 19 jun. 2024.

Neris, Stella. ChatGPT e Ética: conhecendo privacidade e responsabilidade. Disponível em: <https://www.anselmosantana.com.br/2023/06/05/chatgpt-e-etica-privacidade-e-responsabilidade/>. Acesso em: 19 jun. 2024.

Época NEGÓCIOS. ChatGPT pode influenciar pessoas a tomar decisões de vida ou morte, indica estudo. Disponível em: <https://epocanegocios.globo.com/tecnologia/noticia/2023/04/chatgpt-pode-influenciar-pessoas-a-tomar-decisoes-de-vida-ou-morte-segundo-estudo.ghtml>. Acesso em: 19 jun. 2024.

MORAIS, Ana Trigo. ChatGPT: um aliado na tomada de decisões? Disponível em: <https://www.imagensdemarca.pt/artigo/chatgpt-um-aliado-na-tomada-de-decisoes/>. Acesso em: 19 jun. 2024.

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Nome do artigo escolhido: Software engineering practices for machine learning — Adoption, effects, and team assessment
--

Qual o objetivo do estudo descrito pelo artigo?
--

O estudo, que utiliza componentes de Machine Learning, visa coletar e disseminar evidências empíricas sobre a importância das práticas de engenharia em diferentes domínios e organizações.

A publicação compreende um conjunto de dados abrangente que pode ser usado para ajudar as equipes a melhorar a qualidade de seus métodos de engenharia. Além disso, o estudo discute direções promissoras para o desenvolvimento de modelos de qualidade utilizando os dados publicados e apresenta um caso de uso exploratório em que os dados são usados para avaliar a habilidade das equipes de engenharia ao desenvolverem software com componentes de Machine Learning.

O estudo atual é baseado em dois estudos anteriores dos mesmos autores que investiga os métodos de engenharia para todo o ciclo de desenvolvimento e as práticas para o desenvolvimento de um machine learning confiável.

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

O estudo surgiu devido a necessidade de estabelecer métodos sólidos de engenharia de software no desenvolvimento de sistemas que fazem o uso de machine learning pelos seguintes fatores:

Aumento rápido do uso de ML em aplicações como sistemas críticos de segurança, tendo como necessidade um sistema com robustez, confiabilidade e a segurança do sistema algo crucial.

O campo de ML ainda está em desenvolvimento, fazendo com que a comunidade enfrente desafios para identificar e disseminar as melhores práticas para garantir a qualidade e a confiabilidade dos sistemas devido a carência de evidências sobre o quão efetivo são os métodos em diferentes contextos e organizações.

A engenharia de ML carece de modelos de qualidade e maturidade padronizados, dificultando a avaliação da qualidade dos sistemas e identificação

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

Foi realizado uma metodologia de métodos mistos dividida em três etapas:

Revisão de sistemática de literatura (RSL) que tinha como objetivo mapear as melhores práticas de engenharia existentes usando o google, google scholar e outras fontes de informação. Foi filtrado os documentos publicados entre 2015 e 2020 e com base na autoridade das fontes, objetividade e presença de duplicata.

Questionário estruturado com perguntas sobre o perfil das equipes, adoção dos métodos e efeitos percebidos pelos profissionais para validar e mensurar a adoção dos métodos de engenharia mapeadas na RSL. Foi utilizado a estratégia snowball para a distribuição de questionário e realizado a análise das respostas para determinar a taxa de adoção de cada método, agrupando e normalizando as porcentagens de acordo com a escala Likert.

Validação da entrevista para obter insights mais aprofundados sobre a prática do método de engenharia e compreender o não uso de certos métodos. As entrevistas foram transcritas e analisadas usando técnica de análise temática.

Quais os **principais resultados** obtidos pelo estudo?

O estudo resultou na criação de um catálogo com 45 métodos de engenharia de software para ML.

A adoção dos métodos varia conforme a região, o tipo de organização, o tamanho e a experiência da equipe, aumentando com o crescimento e a experiência das equipes. Profissionais tendem a priorizar métodos específicos para ML, em detrimento das práticas tradicionais de engenharia de software.

Métodos voltados para a mitigação de vieses e garantia de segurança têm baixa adoção. Alguns métodos frequentemente utilizados têm baixo impacto na qualidade do software e na rastreabilidade, enquanto os menos populares demonstram grande impacto. Há desafios relatados na implementação dos métodos de testes e monitoramento dos modelos. Há necessidade de diretrizes e políticas claras para garantir a adoção de métodos confiáveis, especialmente para a proteção de dados sensíveis e mitigação de vieses.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

```

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

```

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

```
[ ]: # TensorFlow e tf.keras
import tensorflow as tf
from tensorflow import keras

# Bibliotecas Auxiliares
import numpy as np
import matplotlib.pyplot as plt

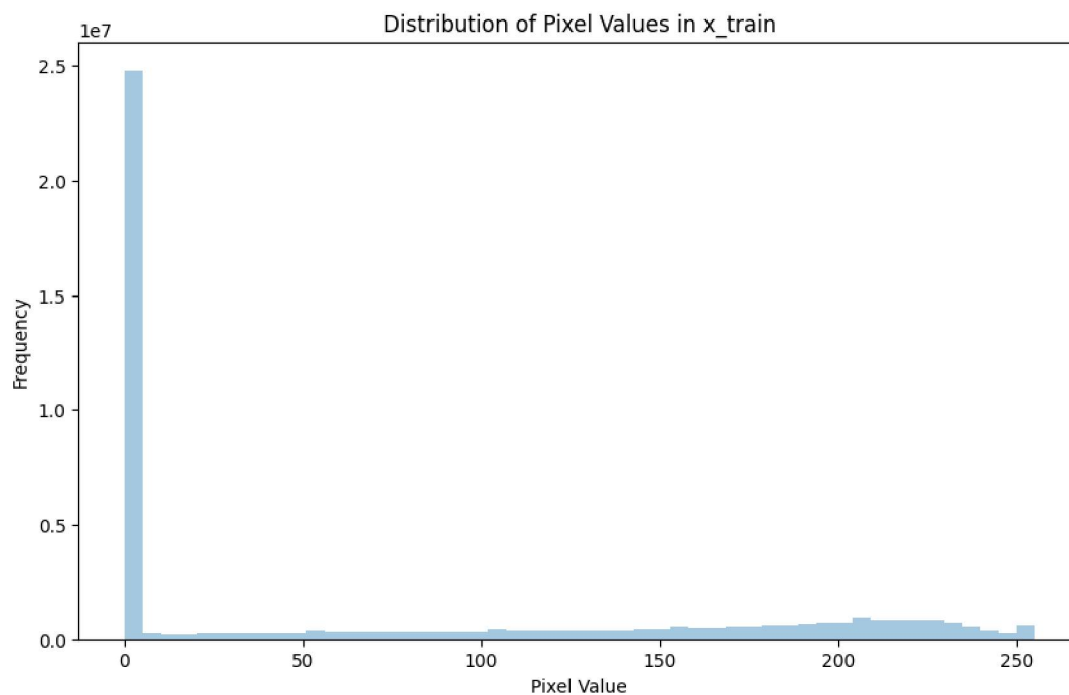
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

# Importando Fashion Mnist Dataset
data = tf.keras.datasets.fashion_mnist

# Treino e Teste
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.
    ↪load_data()

import seaborn as sns

plt.figure(figsize=(10, 6))
sns.distplot(x_train.flatten(), kde=False)
plt.title('Distribution of Pixel Values in x_train')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()
```

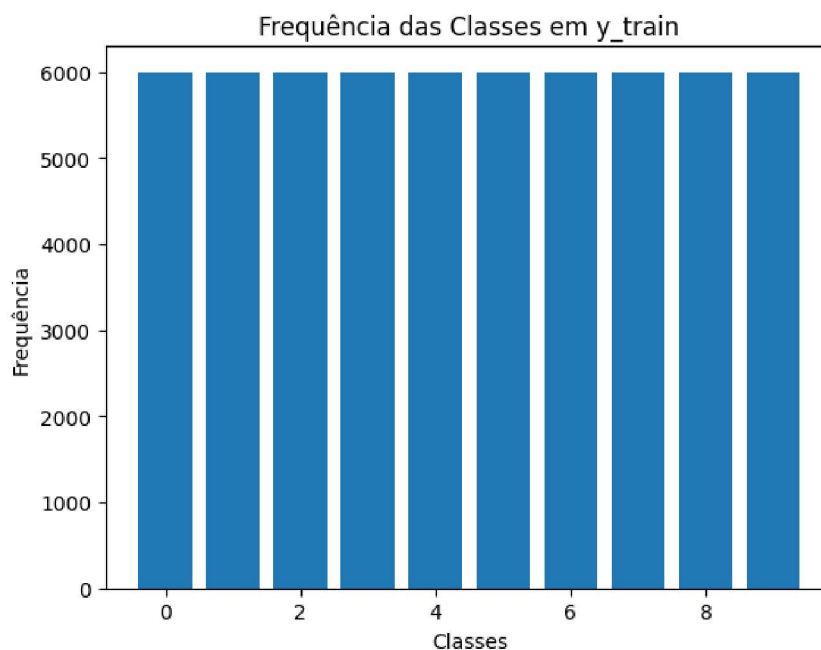


```
[ ]: # Calcular a frequência de cada classe em y_train
unique_classes, class_counts = np.unique(y_train, return_counts=True)

# Criar o gráfico de barras
plt.bar(unique_classes, class_counts)

# Adicionar rótulos aos eixos
plt.xlabel('Classes')
plt.ylabel('Frequência')
plt.title('Frequência das Classes em y_train')

# Mostrar o gráfico
plt.show()
```



```
[ ]: y_train
      # array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
      class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
                     'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

      x_train.shape
      # (60000, 28, 28)
      len(y_train)
      # 60000
      x_test.shape
      # (10000, 28, 28)
      len(y_test)
      # 10000

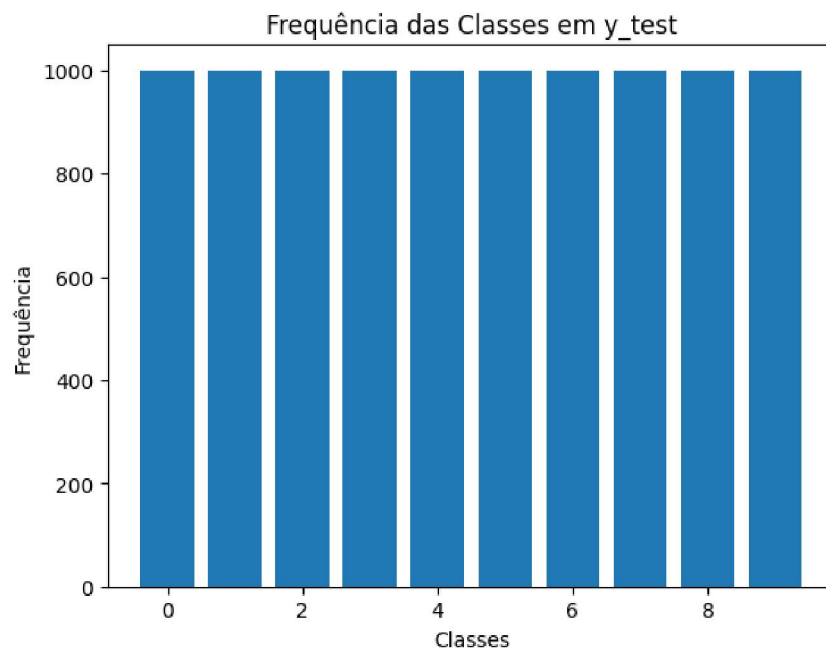
      # Calcular a frequência de cada classe em y_test
      unique_classes, class_counts = np.unique(y_test, return_counts=True)

      # Criar o gráfico de barras
      plt.bar(unique_classes, class_counts)

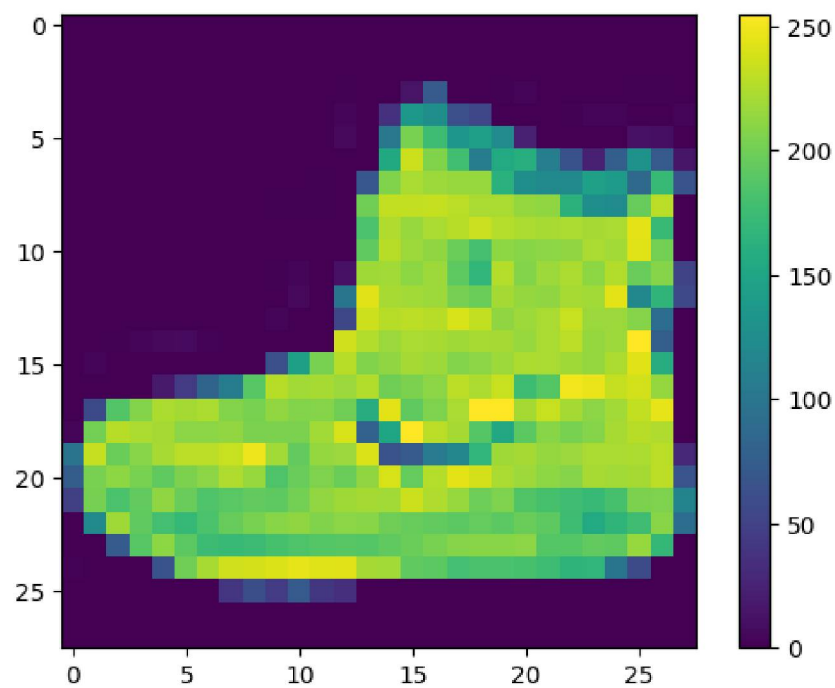
      # Adicionar rótulos aos eixos
      plt.xlabel('Classes')
      plt.ylabel('Frequência')
      plt.title('Frequência das Classes em y_test')
```



```
# Mostrar o gráfico  
plt.show()
```



Pré Processamento



```
[ ]: #Padronização
x_train = x_train / 255.0
x_test = x_test / 255.0

[ ]: plt.figure(figsize=(10,10))
for i in range(36):
    plt.subplot(6,6,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_train[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```



Treinamento do Modelo

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

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

```
[ ]: model.compile(optimizer='adam',
                  loss='sparse_categorical_crossentropy',
```

```
metrics=['accuracy'])
```

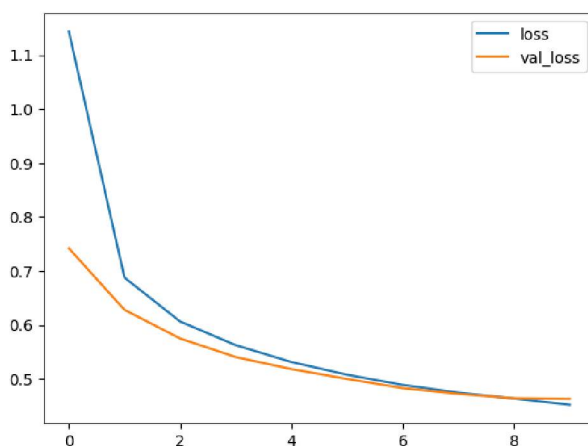
```
[ ]: r = model.fit(x_train, y_train, validation_data = (x_test, y_test), epochs=10)
```

Avaliação do Modelo

```
[ ]: print("Train score: ", model.evaluate(x_train, y_train))
     print("Test score: ", model.evaluate(x_test, y_test))
```

```
1875/1875   3s 2ms/step -
accuracy: 0.8482 - loss: 0.4254
Train score: [0.42815962433815, 0.8467833399772644]
313/313    1s 2ms/step -
accuracy: 0.8392 - loss: 0.4582
Test score: [0.46324795484542847, 0.8338000178337097]
```

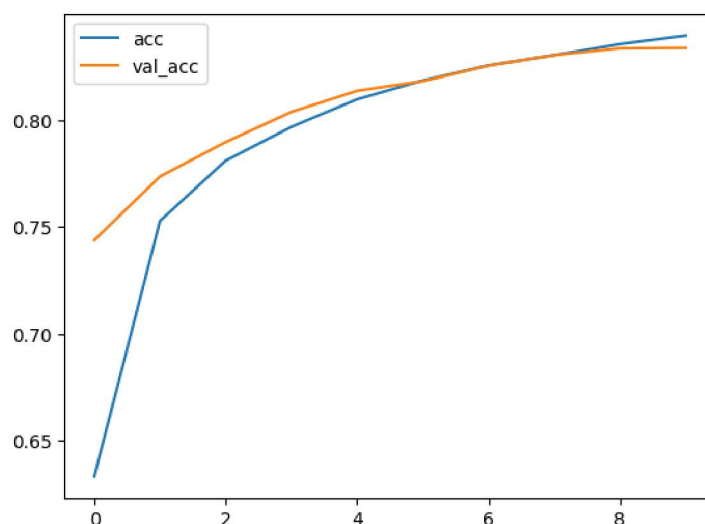
Gráfico de Perda



No plot acima temos a evolução da perda nos dados de treinamento (azul) e dos dados de validação (laranja). O modelo começa com uma perda alta, mas melhora rapidamente nas próximas épocas. Vemos que está convergindo adequadamente, pois ambas as curvas apresentam tendência decrescente e se estabilizam nas épocas finais. Também não é possível ver overfitting significativo, já que na maioria do histórico as curvas se mantêm distantes. Por volta da época 6 ambas as curvas começam a estabilizar a perda em torno de 0.50 indicando que o modelo atingiu um ponto de equilíbrio na aprendizagem.

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

Gráfico de Acurácia



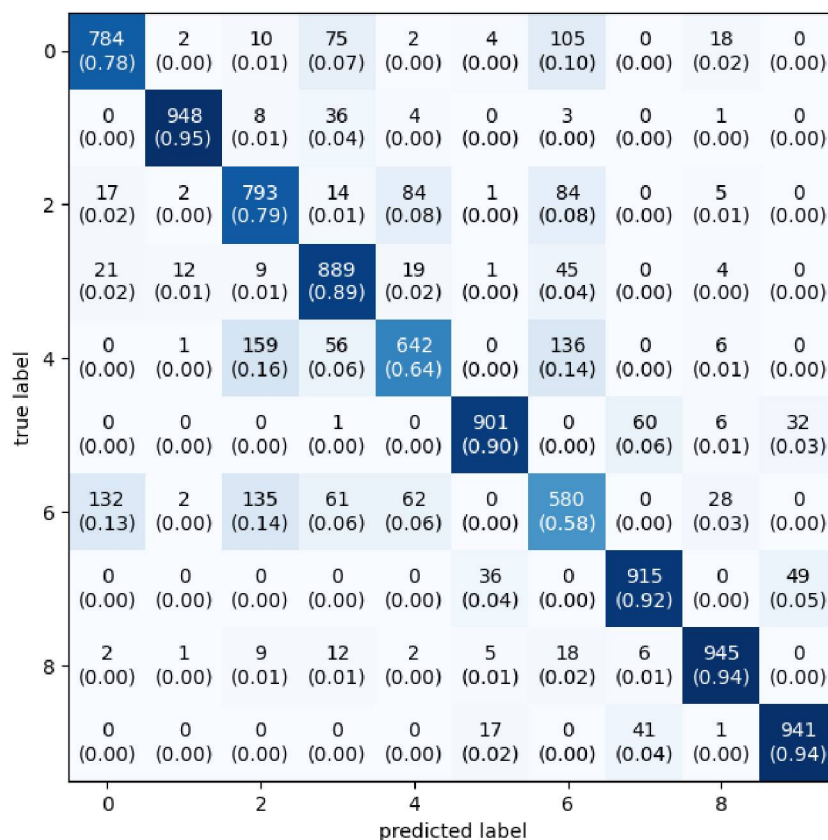
No gráfico acima é possível observar a evolução da acurácia (eixo y) com o passar das épocas (eixo x), em azul estão os dados de treinamento e em laranja os dados de validação. Na primeira época a acurácia dos dados de treinamento é de 53%, ou seja, do total de observações de treinamento, o modelo conseguiu classificar corretamente pouco mais da metade, apesar disso já na primeira época a acurácia dos dados separados para validação é superior ao treinamento, cerca de 74%, o que mostra bom potencial do modelo em generalizar quando apresentado a novos dados. Na segunda época a acurácia dos dados de treinamento aumenta cerca de 21 pontos percentuais, chegando a 74%, enquanto a acurácia dos dados de validação, apenas 3 pp, chegando a 77%, o que demonstra melhora do aprendizado do modelo. A distância das acurácias entre amostra de treino e validação diminui à medida que as épocas evoluem. São praticamente equivalentes entre as épocas 5 a 8, quando então a acurácia dos dados de treinamento supera ligeiramente a dos dados de validação. Um bom ponto de parada seria a época 7, para evitar overfitting.

Predições

```
[ ]: y_pred = model.predict(x_test).argmax(axis=1)
      print(y_pred)
```

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

Acurácia



```
[ ]: test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)

print('\nTest accuracy:', test_acc)
```

313/313 - 0s - 2ms/step - accuracy: 0.8878 - loss: 1.3274

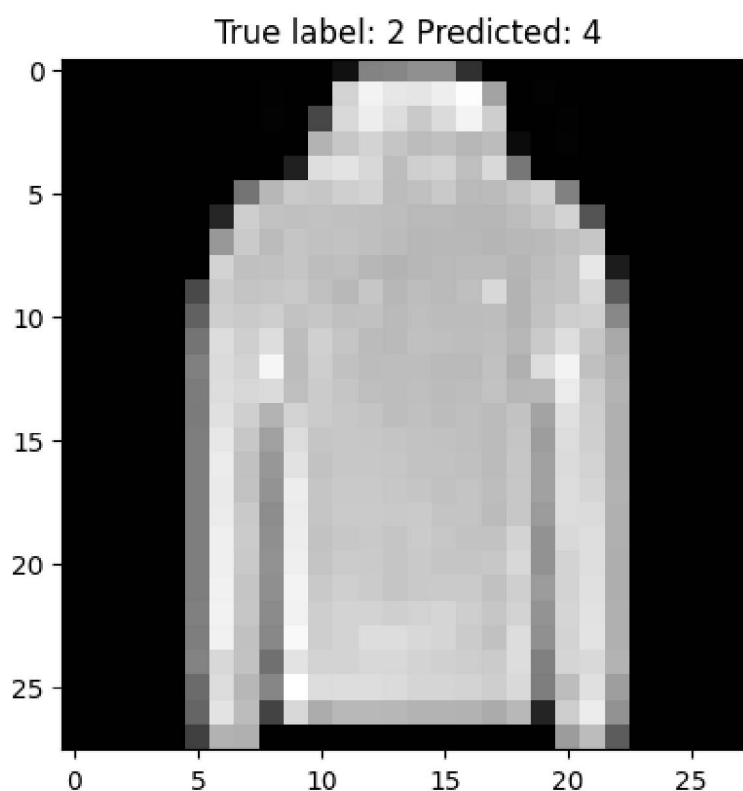
Test accuracy: 0.8877999782562256

O modelo identificou corretamente 88,78% dos dados de teste.

Mostrar algumas Predições Erradas

```
[ ]: misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)
plt.imshow(x_test[i].reshape(28, 28), cmap='gray')
plt.title('True label: %s Predicted: %s' % (y_test[i], y_pred[i]))
```



Acima está um exemplo predição incorreta que o modelo acabou classificando como “Coat”, categoria 4, mas na verdade era um “Pullover”, categoria “2”.

2 - Regressão (RNA)

```
[ ]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

import tensorflow as tf

from tensorflow.python.keras import backend
from sklearn.preprocessing import StandardScaler
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'
```



```

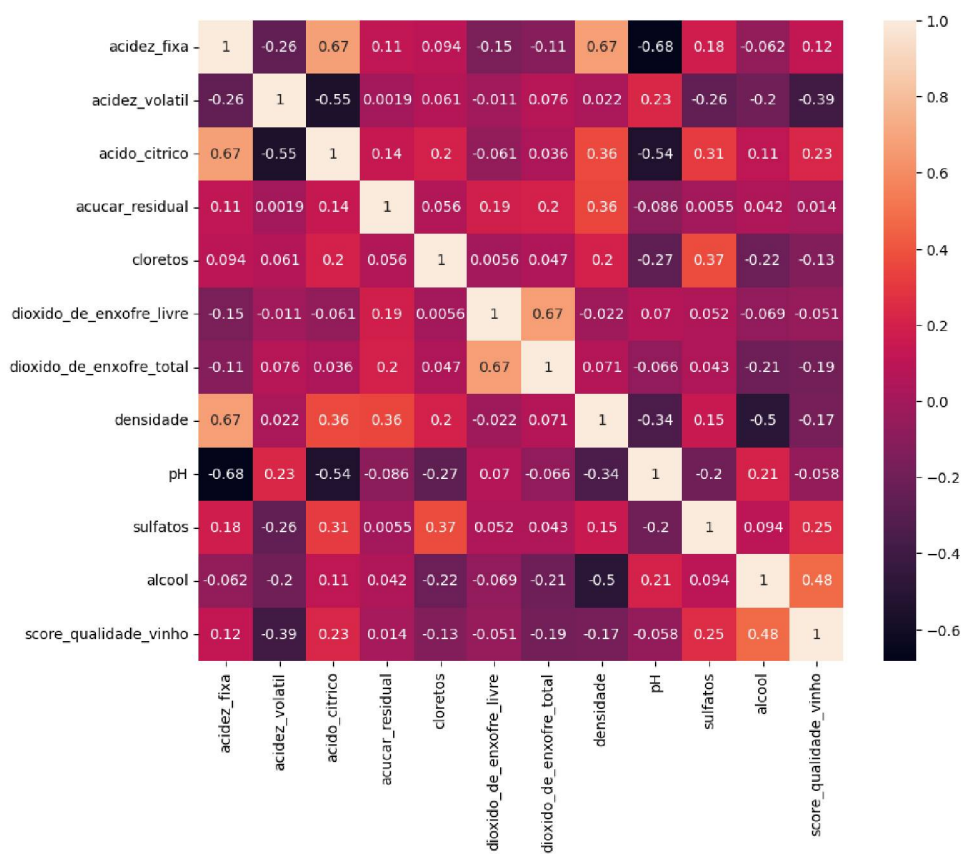
data = pd.read_csv(url, delimiter=';', decimal='.')

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
]

fig, ax = plt.subplots(figsize=(10,8))
sb.heatmap(data.corr(), annot=True, ax=ax)
plt.show()

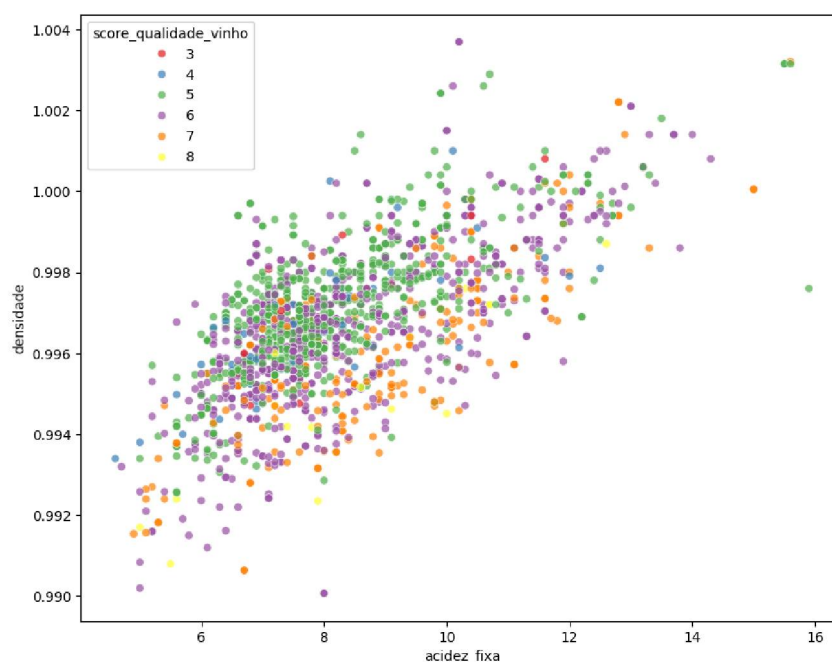
```

Matriz de correlação



```
[ ]: plt.figure(figsize=(10, 8))
      sb.scatterplot(
          x='acidez_fixa',
          y='densidade',
          hue='score_qualidade_vinho',
          data=data,
          alpha=0.7,
          palette='Set1')
```

Matriz de dispersão



```
[ ]: # Separação em dados de treino e de teste
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

      # Criação do Modelo
      scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)

      i = tf.keras.layers.Input(shape=(11,))
      x = tf.keras.layers.Dense(16, activation='relu')(i)
      x = tf.keras.layers.Dense(8)(x)
```

```

x = tf.keras.layers.Dense(6)(x)
x = tf.keras.layers.Dense(1)(x)

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

# Compilação e treinamento do modelo
# Função RMSE
def rmse(y_true, y_pred):
    return backend.sqrt(backend.mean( backend.square(y_pred - y_true), axis=-1) )

# Função R2
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)

# Compilando
optimizer=tf.keras.optimizers.Adam(learning_rate=0.02)
# 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])

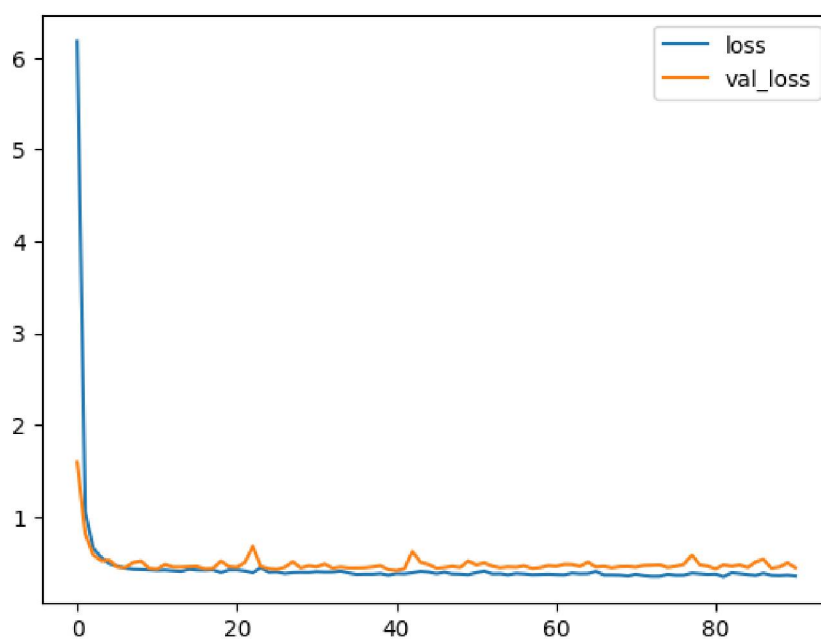
# Condição de parada antecipada
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=50,
    restore_best_weights=True)

r = model.fit(
    X_train, y_train,
    epochs=1500,
    batch_size=32,
    validation_data=(X_test, y_test),
    callbacks=[early_stop])

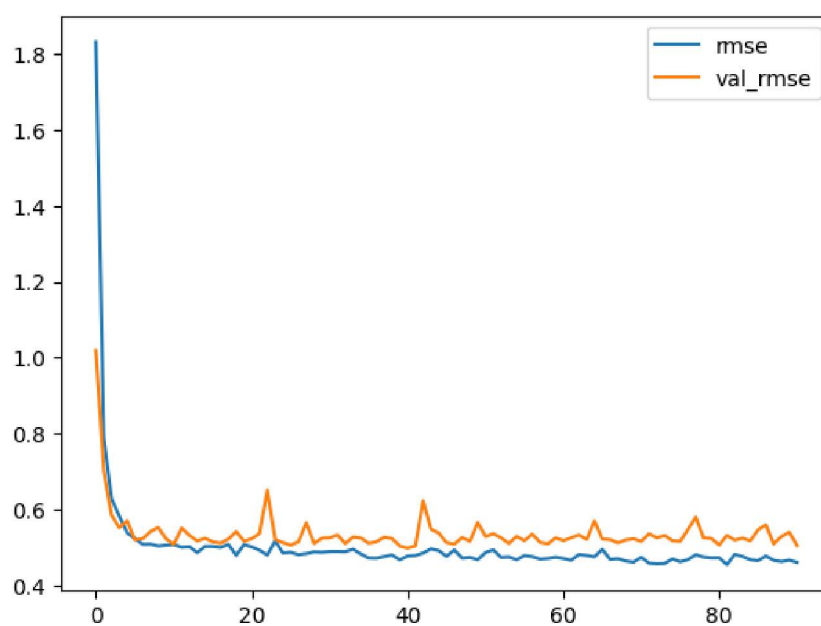
# Avaliação do modelo
plt.plot( r.history['loss'] , label='loss' )
plt.plot( r.history['val_loss'] , label='val_loss' )
plt.legend()

```

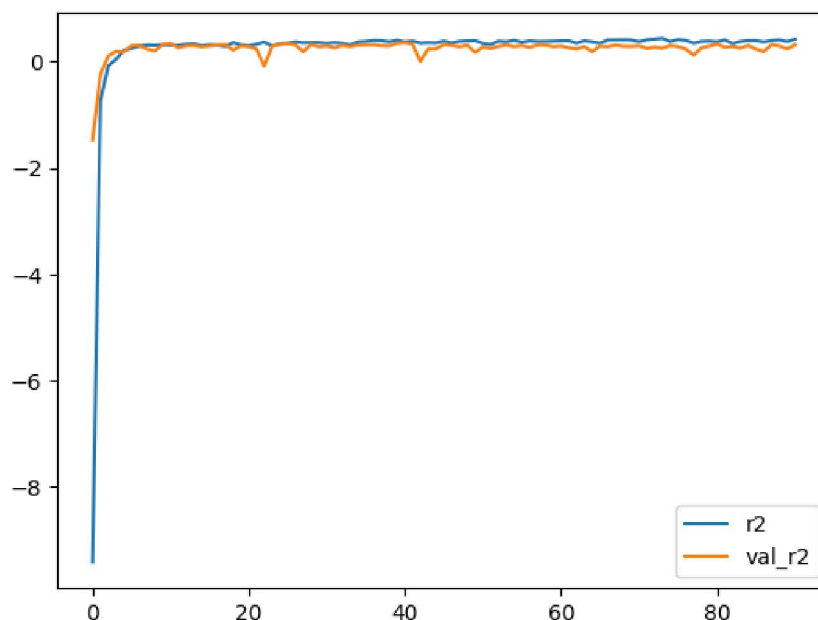
Avaliação do modelo



```
[ ]: plt.plot( r.history['rmse'] , label='rmse' )  
plt.plot( r.history['val_rmse'] , label='val_rmse' )  
plt.legend()
```



```
[ ]: plt.plot( r.history['r2'] , label='r2' )
      plt.plot( r.history['val_r2'] , label='val_r2' )
      plt.legend()
```



#####Foi utilizado o parâmetro epochs (épocas) setado para 1500, visando um maior desenvolvimento do modelo. No entanto, com intuito de não exaurir todo processamento, foi colocada a condição de parada com o parâmetro patience em 50, monitorando a variável val_loss. Conforme se verifica nos gráficos, o processamento alcança uma estabilização relativamente cedo. Uma das causas pode ser o tamanho pequeno da amostra.

Predições

```
[ ]: y_pred = model.predict(X_test)
      mse = mean_squared_error(y_test, y_pred)
      rmse = sqrt(mse)
      r2 = r2_score(y_test, y_pred)

      print('mse      = ', mse)
      print('rmse     = ', rmse)
      print('r2       = ', r2)
```

```
mse      = 0.4122875117521187
rmse     = 0.6420961857479911
r2       = 0.38747960329055786
```

R: Depois de executado o modelo, alcançou-se o valor de cerca de 0.41228 para o Mean Squared Error (MSE), que é a métrica utilizada para descrever o quão perto a reta de regressão está em relação ao conjunto de dados. Ainda, pode-se citar o Coeficiente de Determinação, que teve um valor aproximado de 0.38747. Esse valor indica que que, no conjunto de dados observado, há uma correlação leve em que as variáveis movem-se na mesma direção crescente.

3 - Sistemas de Recomendação

```
[ ]: 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

df = pd.read_csv('Base_livros_in_6.csv', on_bad_lines='skip', sep=',')
print(df.columns.values)
# ['ISBN' 'Titulo' 'Autor' 'Ano' 'Editora' 'ID_usuario' 'Notas']
df.shape
# (128895, 7)
df.ID_usuario = pd.Categorical(df.ID_usuario)
df.ISBN = pd.Categorical(df.ISBN)
df.Titulo = pd.Categorical(df.Titulo)
df.Autor = pd.Categorical(df.Autor)
df.Editora = pd.Categorical(df.Editora)
df.Ano = pd.Categorical(df.Ano)
df.Notas = pd.Categorical(df.Notas)
df['new_ID_usuario'] = df.ID_usuario.cat.codes
df['new_ISBN'] = df.ISBN.cat.codes
df['new_notas'] = df.Notas.cat.codes
# Foi necessário ajustar o parametro K (dimensão dos embeddings)
N = len(set(df.new_ID_usuario))
M = len(set(df.new_ISBN))
K = 100
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u)
u_emb = Flatten()(u_emb)

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

```

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

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

model = Model(inputs=[u, m], outputs=x)
model.compile(
    loss='mse',
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)

user_ids, isbns, ratings = shuffle(df.new_ID_usuario, df.new_ISBN, df.new_notas)

Ntrain = int(0.7 * len(ratings))

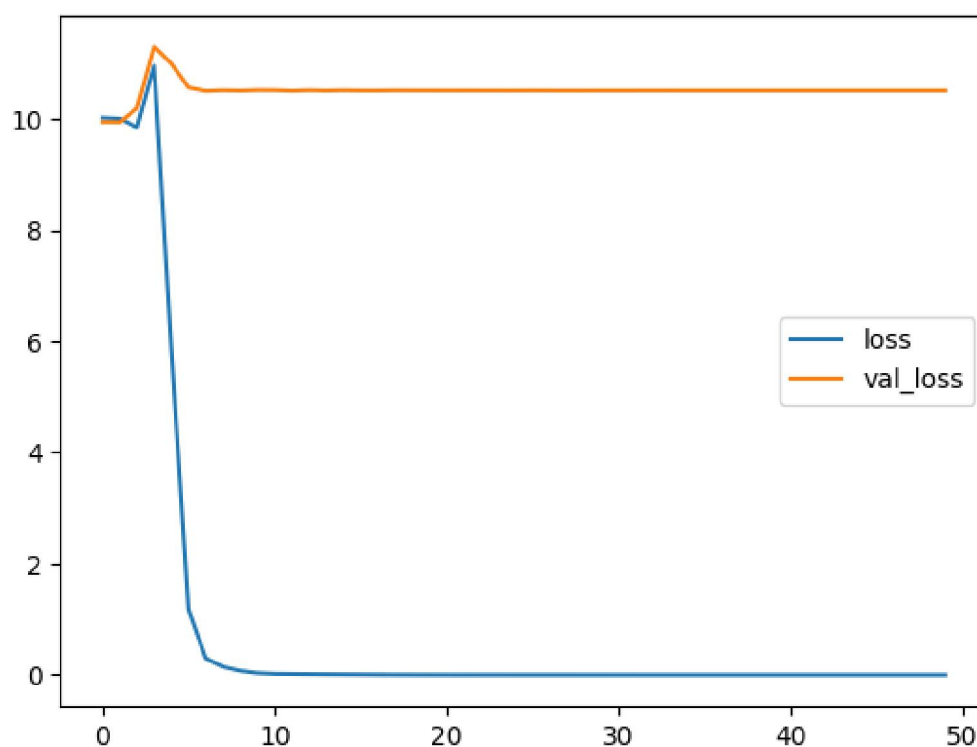
train_users = user_ids:Ntrain
train_books = isbns:Ntrain
train_ratings = ratings:Ntrain
test_users = user_ids[Ntrain:]
test_books = isbns[Ntrain:]
test_ratings = ratings[Ntrain:]

# centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating
epochs = 50
r = model.fit(
    x=[train_users, train_books],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2,
    validation_data=([test_users, test_books], test_ratings)
)

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

```


Gráfico da função de perda



Foi utilizado o número de 50 para epochs, depois de algumas epochs o gráfico se estabiliza, a diferença permanece grande devido ao número de dados ser pequeno. (por volta de 128 mil em comparação aos 20 milhões do exemplo apresentado em aula).

Geração da recomendação

```
[ ]: input_usuario = np.repeat(a=11089, repeats=M)
      book = np.array(list(set(isbns)))

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

      rat = preds.flatten() + avg_rating

      idx = np.argmax(rat)

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

4028/4028 8s 2ms/step

Recomendação: Livro - 88700 / 10.694294 *

```
[ ]: print(df.loc[[book[idx]]])
```

	ISBN	Titulo \
88700	451526201	Plunkitt of Tammany Hall: A Series of Very Pla...

	Autor	Ano	Editora	ID_usuario	Notas \
88700	William L. Riordon	1995	Signet Classics	19609	9

	new_ID_usuario	new_ISBN	new_notas
88700	7141	72537	9

```
[ ]: input_usuario = np.repeat(a=50, repeats=M)
book = np.array(list(set(isbns)))

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

rat = preds.flatten() + avg_rating

idx = np.argmax(rat)

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

4028/4028 8s 2ms/step

Recomendação: Livro - 113079 / 10.522945 *

```
[ ]: print(df.loc[[book[idx]]])
```

	ISBN	Titulo \
113079	1560441771	Where Dinosaurs Still Rule: A Guide to Dinosau...

	Autor	Ano	Editora	ID_usuario	Notas	new_ID_usuario \
113079	Debbie Tewell	1993	Globe Pequot Pr	26216	3	9689

	new_ISBN	new_notas
113079	15779	3

R: Executando o sistema de recomendação com dois usuarios diferentes 11089 e 50 e exibindo duas recomendações diferentes de livros. Para o sistema de recomendações funcionar foi necessario alterar o parametro de embeddings.

4 - Deepdream

Importando a imagem

```
[ ]: 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'

def download(url, max_dim=None):
    name = url.split('/')[-1]
    image_path = tf.keras.utils.get_file(name, origin=url)
    img = PIL.Image.open(image_path)
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)

def deprocess(img):
    img = 255*(img + 1.0)/2.0
    return tf.cast(img, tf.uint8)

def show(img):
    display.display(PIL.Image.fromarray(np.array(img)))

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>'))
```



```
[ ]: # Preparação do modelo para extração
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)

# Calculo da perda
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)

# Calculo do Gradiente em cima da Imagem Original
class DeepDream(tf.Module):
    def __init__(self, model):
        self.model = model

n+ndtf.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)

# Main Loop
def run_deep_dream_simple(img, steps=100, step_size=0.01):

    img = tf.keras.applications.inception_v3.preprocess_input(img)
    img = tf.convert_to_tensor(img)
    step_size = tf.convert_to_tensor(step_size)
    steps_remaining = steps
    step = 0
    while steps_remaining:
        if steps_remaining > 100:
            run_steps = tf.constant(100)
        else:
            run_steps = tf.constant(steps_remaining)
        steps_remaining = steps_remaining - run_steps
        step = step + run_steps

        loss, img = deepdream(img, run_steps, tf.constant(step_size))

        display.clear_output(wait=True)
        show(deprocess(img))
        print('Step , loss '.format(step, loss))

    result = deprocess(img)
    display.clear_output(wait=True)
    show(result)

    return result

dream_img = run_deep_dream_simple(img=original_img,

```

```
steps=100, step_size=0.01)
```

Resultado Main Loop



```
[ ]: import time
start = time.time()

OCTAVE_SCALE = 1.60

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[: -1]
float_base_shape = tf.cast(base_shape, tf.float32)

for n in range(-2, 3):
    new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

    img = tf.image.resize(img, new_shape).numpy()

    img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```


Modelo até um oitava



Imagem onírica obtida por Main Loop

R: No Main Loop é feito um pré processamento feito uma conversão de acordo com as iterações definidas. Depois é feito um aumento de gradiente. Depois a imagem é desprocessada que vai converter a imagem com o valor original dando para ver a imagem gerada com o a função `dream_img`. A imagem resultante é uma imagem ruidosa com baixa resolução e ao se observar a imagem onírica, não parece ter muito padrão comparado ao mostrado em aula que tem as bordas da imagem bem definidas dando para observar os contornos da figura que está na imagem.

Imagem onírica obtida ao levar o modelo até uma oitava;

R: Neste é feito diversos redimensionamento para ajustar a granularidade e o problema de baixa resolução do modelo, na imagem obtida, é possível verificar os contornos mais salientes resultando em uma imagem onírica onde é possível identificar uma diferença entre o fundo e o animal.

Diferenças entre imagens oníricas obtidas com Main Loop e levando o modelo até a oitava.

R: A diferença que no main loop produz uma imagem onírica com padrões aprimorados na escala original da imagem. Já no modelo levado até uma oitava é possível ver uma granularidade maior gerando uma imagem mais onírica onde a diferença de gradiente é nitidamente visível gerando uma imagem mais complexa e detalhada

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

- O conjunto de dados brutos (ou uma visualização de dados que você acredite que possa ser **melhorada**);
- Explicação do **contexto e o público-alvo** da visualização de dados e do storytelling que será desenvolvido;
- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada; (50 pontos)**

B – RESOLUÇÃO

1. Base de dados bruto/Visualização de dado

Link para a base de dados: <https://basedosdados.org/dataset/782c5607-9f69-4e12-b0d5-aa0fla7a94e2?table=28d16282-d100-4ea8-9dde-36c05c8f1ca2>

Foi necessário selecionar os dados pois a base possui mais de 1 GB. Nele foi filtrado e coletado apenas os dados que são interessantes para esse storytelling. Os dados foram filtrados pelo id_estacao da cidade de São Paulo, pegando a média da precipitação por mês de todos os anos (2000-2024). As informações do id_estacao de cada município estão disponíveis no mesmo link fornecido.

Média de precipitações	
mes	media_precipitacao (mm)
1	10.091
2	11.642
3	10.496
4	7.126
5	4.686
6	5.972
7	4.843
8	3.889
9	6.002
10	7.121
11	7.694
12	10.498

Fonte do mapa fornecido no storytelling:

<https://quilomboperiferico.com.br/registro-de-enchentes/>

2. Contexto e Público-Alvo

Os dados brutos apresentado é a média de precipitações de 2000 a 2024 por meses na cidade de São Paulo. Com esses dados está sendo feito a relação das épocas de maior número de chuvas em São Paulo que possui mais enchentes e usando uma história fictícia, será passado os dados adquiridos.

Com esta visualização e storytelling tem como principais públicos alvo: Governo Municipal para investir na infraestrutura urbana para melhorar os sistemas de drenagem da cidade e alertas de possíveis enchentes. Para a população local se conscientizar e compadecer com situação das vítimas de enchentes seja por meio de trabalho voluntário, doações ou apoio emocional.

3. Visualização de Dados:

Como mencionado no primeiro tópico, o dado já foi tratado via big query devido ao grande volume de dados apresentados na base bruta. O dado filtrado e coletado era pequeno, então pela praticidade do Excel foi utilizado o mesmo para gerar o gráfico de precipitações usado no storytelling mostrando a média de precipitações de janeiro a dezembro dos anos 2000 a 2024. Foi utilizado esse tipo de visualização do gráfico, pois é facilita a visualização dos meses que houve a maior média de precipitações ocorridas por meses. Com esses dados, podemos entender como a quantidade de chuvas que ocorre no mês afeta diretamente a quantidade de enchentes que ocorre durante o ano.

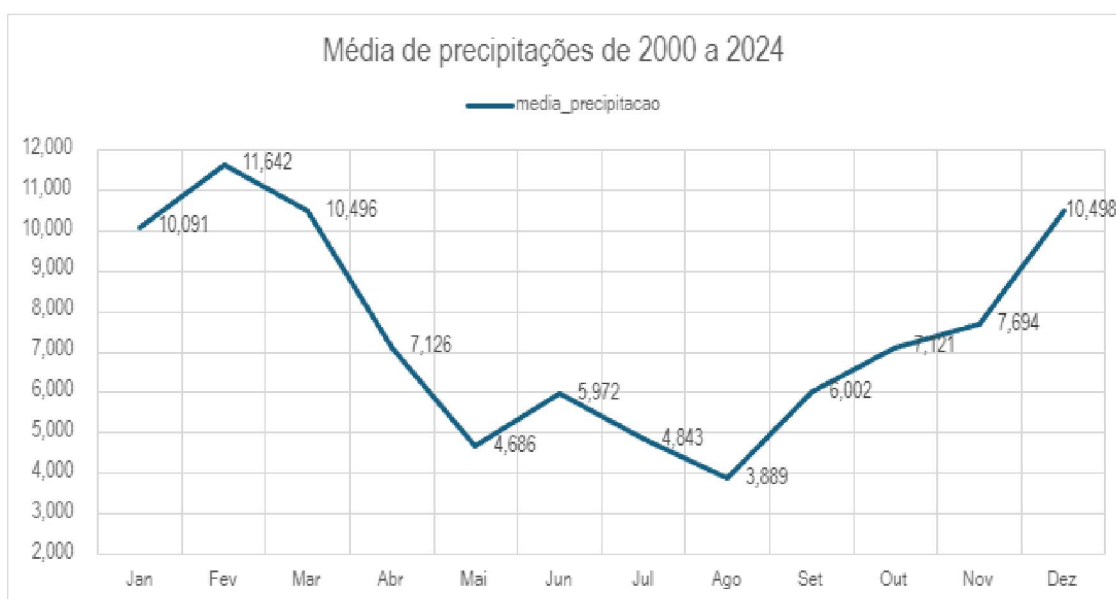
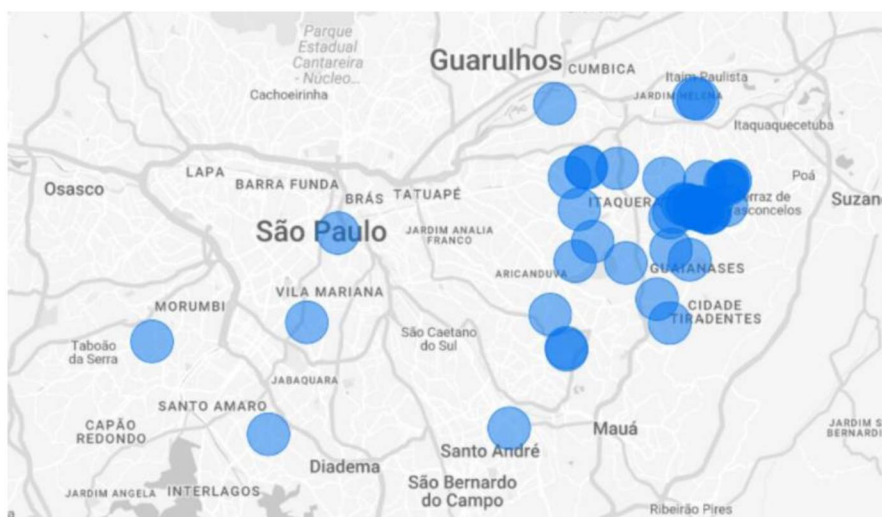
A imagem fornecida sobre as regiões mais afetadas pelas enchentes em São Paulo foi escolhida para enfatizar a história relatada durante o storytelling. Não foi encontrado uma base bruta com os dados dela tabela.

4. Narrativa/Storytelling:

São Paulo, a maior cidade do Brasil, é também uma das mais castigadas pelas chuvas de verão. Todos os anos, entre dezembro e março, as ruas se transformam em rios turbulentos, e as casas em ilhas de desespero. O que deveria ser uma bênção da natureza.

A chuva que irriga os campos e enche os reservatórios torna-se uma maldição para milhares de famílias. Esta é a história de como as águas que deveriam trazer vida acabam levando sonhos, esperanças e, muitas vezes, tudo o que as pessoas têm. É a história de uma cidade que luta contra as enchentes, mas também de uma população que resiste mesmo quando as águas sobem e o futuro parece incerto.

Enchentes na cidade de São Paulo



Janeiro: Início do Caos

Dezembro a março marcam o período das chuvas mais intensas em São Paulo, e janeiro é quando a natureza começa a mostrar sua força implacável. Com uma média de **10.091 mm** de precipitação, as ruas já começam a alagar, transformando avenidas em rios e bairros em cenários de caos. Para muitos, como a família Souza, que mora no Itaim Paulista, na Zona Leste, é o sinal de que o verão será longo e difícil. Dona Marta, mãe de dois filhos, lembra com angústia do dia em que a água invadiu sua casa. "A chuva não parava, e a água começou a entrar pela porta. Tivemos que sair às pressas, carregando o que dava" relata enquanto segura uma foto da família, já desbotada pela umidade. "Perdemos quase tudo. Até os brinquedos das crianças foram levados pela enxurrada."

A história das enchentes em São Paulo não é nova. Em janeiro de 1929, a cidade viveu uma de suas maiores tragédias: **125,4 mm** de chuva caíram em um único dia, sendo **82 milímetros** em apenas **uma hora**. O rio Tietê transbordou, subindo cerca de dois metros e arrastando tudo em seu caminho. Muros desabaram, animais domésticos morreram afogados e bairros como Vila Elisa, Vila Guilherme e Mooca ficaram debaixo d'água. Quase um século depois, a história se repete. Em 2025, mal começou o ano, e um idoso de 73 anos perdeu a vida após ser arrastado por uma enxurrada em Pinheiros. Será que essa tragédia poderia ter sido evitada?

Fevereiro: O Pico das Águas

Fevereiro é o mês mais crítico, com uma média de **11,64 mm** de chuva. Foi nesse mês que o pequeno Lucas, filho de Dona Marta, de 7 anos, viu sua casa ser invadida pela água. "Não pude sair para brincar por dias" conta ele, com os olhos cheios de lágrimas, enquanto segura o carrinho de brinquedo que conseguiu salvar. Para os pais, o desafio é ainda maior: como garantir que os filhos tenham um futuro melhor se a educação é interrompida pelas chuvas? "Ele perdeu uma semana de aula porque a escola ficou alagada", lamenta Dona Marta, enquanto tenta secar os livros do filho ainda encharcados.

No dia 24 de fevereiro deste ano, os moradores da cidade de São Paulo foram surpreendidos pela primeira vez com um alerta de chuvas severas da Defesa Civil no celular. Nesse dia, devido ao calor intenso e à umidade excessiva, uma tempestade histórica despejou **144 mm** de chuva em apenas **três horas**, o segundo maior volume desde 1961. As ruas viraram rios, carros foram arrastados e famílias inteiras perderam tudo o que tinham. "Foi como se o céu tivesse caído sobre a cidade", descreve um morador. Esse evento não só mostrou a força da natureza, mas também a urgência de uma melhoria no planejamento urbano para a cidade de São Paulo. Enquanto as águas baixavam, uma pergunta ecoava na mente de todos: até quando vamos conviver com essa realidade?

Março: A Resistência Continua

No quarto mês consecutivo, março ainda traz chuvas intensas, com uma média de **10,49 mm**. Dona Marta e sua família já estão mais preparadas, mas o medo não desaparece. "A cada chuva forte, a gente fica de olho no rio, torcendo para não transbordar", diz ela, enquanto observa as águas barrentas subirem perigosamente perto de sua casa. As famílias aprenderam a conviver com as enchentes, mas o custo emocional é alto. "É como se a gente vivesse em um estado de alerta constante", desabafa Dona Marta, segurando o pequeno Lucas, que ainda tem pesadelos com o dia em que a água invadiu seu quarto. Apesar da diminuição das chuvas, ainda não é o fim das enchentes. Para muitos, é apenas uma trégua temporária antes do próximo dilúvio.

Abril a setembro: Um Respiro

A partir de abril, as chuvas começam a diminuir com uma média de **7,12 mm**, e chegando a apenas **3,88 mm** em agosto. Para muitas famílias, como a de Dona Marta, é um período de reconstrução. Ela aproveita para reformar a casa, substituindo os móveis destruídos pela água e repintando as paredes que ainda carregam marcas do nível das enchentes. "É como se a gente estivesse sempre se preparando para a próxima batalha", diz ela enquanto arruma os poucos pertences que conseguiu salvar.

O que será das famílias de São Paulo?

Outubro marca o início do aumento das chuvas com uma média de **7,12 mm**, que sobe para **10,49 mm** em dezembro. Para a família Souza, é um período de ansiedade. "Não posso viajar com medo de encontrar a casa alagada", diz Dona Marta. A família continua a vida mas a sensação de incerteza e impotência é grande, postergando planos e sonhos.

A história da família Souza é um reflexo da realidade de milhares de paulistanos que convivem com as enchentes todos os anos. As chuvas, que deveriam ser uma bênção, se transformam em uma maldição para muitos, mas a mudança é possível. Com a construção de piscinões, a revitalização dos rios e a conscientização da população sobre o descarte

correto do lixo, podemos transformar São Paulo em um lugar mais seguro e resiliente. E você, o que pode fazer para ajudar? Comece hoje mesmo, descartando o lixo corretamente e cobrando ações do governo. Juntos, podemos garantir que as águas que hoje trazem destruição, no futuro, tragam vida e esperança.

APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

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

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva. O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA. O PDF deverá conter o código-fonte e as imagens obtidas.

B - RESOLUÇÃO

Questão 1

```
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import permutations

# Definição de parâmetros
NUM_CIDADES = 100
DIMENSAO = 100
POPULACAO_SIZE = 100
GERACOES = 1000
TAXA_MUTACAO = 0.01

# Gerar cidades com coordenadas aleatórias
cidades = np.random.rand(NUM_CIDADES, 2) * DIMENSAO

# Função para calcular a distância euclidiana
def distancia_total(caminho, cidades):
    dist = 0
    for i in range(len(caminho) - 1):
        dist += np.linalg.norm(cidades[caminho[i]] - cidades[caminho[i +
        ↪ 1]])
    dist += np.linalg.norm(cidades[caminho[-1]] - cidades[caminho[0]]) #
    ↪ Retorno à cidade inicial
    return dist
```

Inicialização da população

```
def inicializar_populacao(size, num_cidades):
    populacao = []
    for _ in range(size):
        individuo = list(np.random.permutation(num_cidades))
        populacao.append(individuo)
    return populacao
```

Função de seleção por torneio

```
def selecao_torneio(populacao, cidades, k=5):
    selecionados = random.sample(populacao, k)
    selecionados.sort(key=lambda ind: distancia_total(ind, cidades))
    return selecionados[0]
```

Crossover OX (Order Crossover)

```
def crossover_ox(parent1, parent2):
    size = len(parent1)
    p1, p2 = sorted(random.sample(range(size), 2))
    child = [None] * size
    child[p1:p2] = parent1[p1:p2]
    fill_pos = [i for i in parent2 if i not in child]
    idx = 0
    for i in range(size):
        if child[i] is None:
            child[i] = fill_pos[idx]
            idx += 1
    return child
```

Mutação por swap

```
def mutacao(individuo):
    if random.random() < TAXA_MUTACAO:
```

```

        i, j = random.sample(range(len(individuo)), 2)
        individuo[i], individuo[j] = individuo[j], individuo[i]
    return individuo

# Algoritmo Genético
def algoritmo_genetico():
    populacao = inicializar_populacao(POPULACAO_SIZE, NUM_CIDADES)
    melhor_solucao = min(populacao, key=lambda ind: distancia_total(ind,
↪ cidades))

    for geracao in range(GERACOES):
        nova_populacao = []
        while len(nova_populacao) < POPULACAO_SIZE:
            parent1 = selecao_torneio(populacao, cidades)
            parent2 = selecao_torneio(populacao, cidades)
            child = crossover_ox(parent1, parent2)
            child = mutacao(child)
            nova_populacao.append(child)

        populacao = sorted(nova_populacao, key=lambda ind:
↪ distancia_total(ind, cidades))
        if distancia_total(populacao[0], cidades) <
↪ distancia_total(melhor_solucao, cidades):
            melhor_solucao = populacao[0]

    return melhor_solucao

# Executar o algoritmo e obter melhor caminho
melhor_caminho = algoritmo_genetico()

# Plotar cidades e caminho
plt.figure(figsize=(8, 8))

```

```

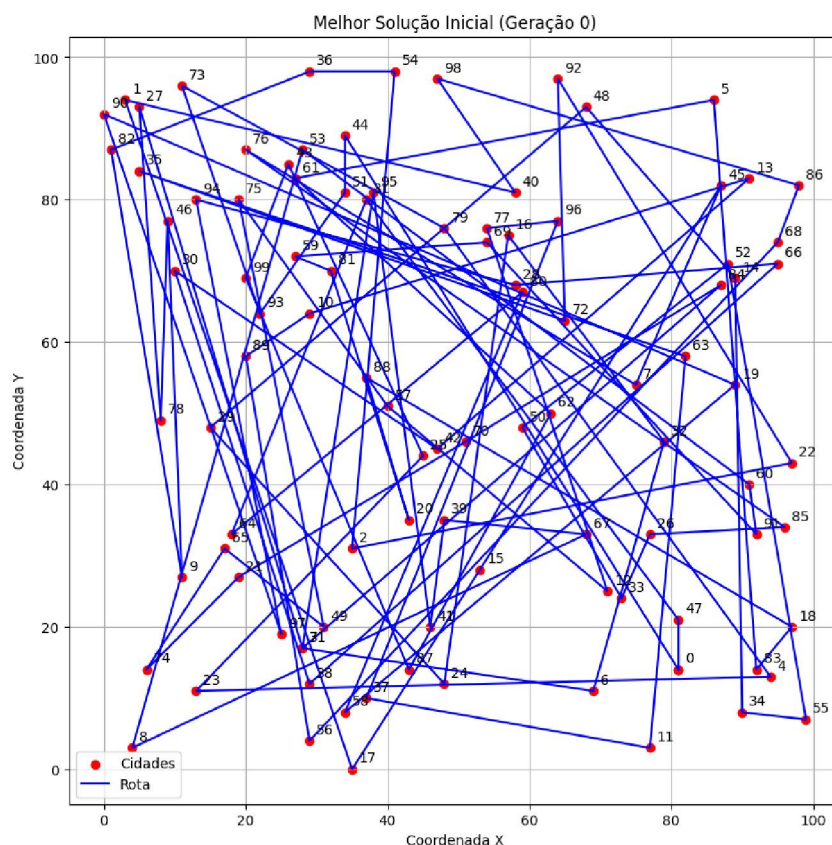
caminho_x = [cidades[i][0] for i in melhor_caminho] +
↳ [cidades[melhor_caminho[0]][0]]
caminho_y = [cidades[i][1] for i in melhor_caminho] +
↳ [cidades[melhor_caminho[0]][1]]
plt.plot(caminho_x, caminho_y, marker='o', linestyle='--')
plt.scatter(cidades[:, 0], cidades[:, 1], c='red')
plt.title("Melhor Caminho Encontrado")
plt.show()

```

Mostrando os resultados

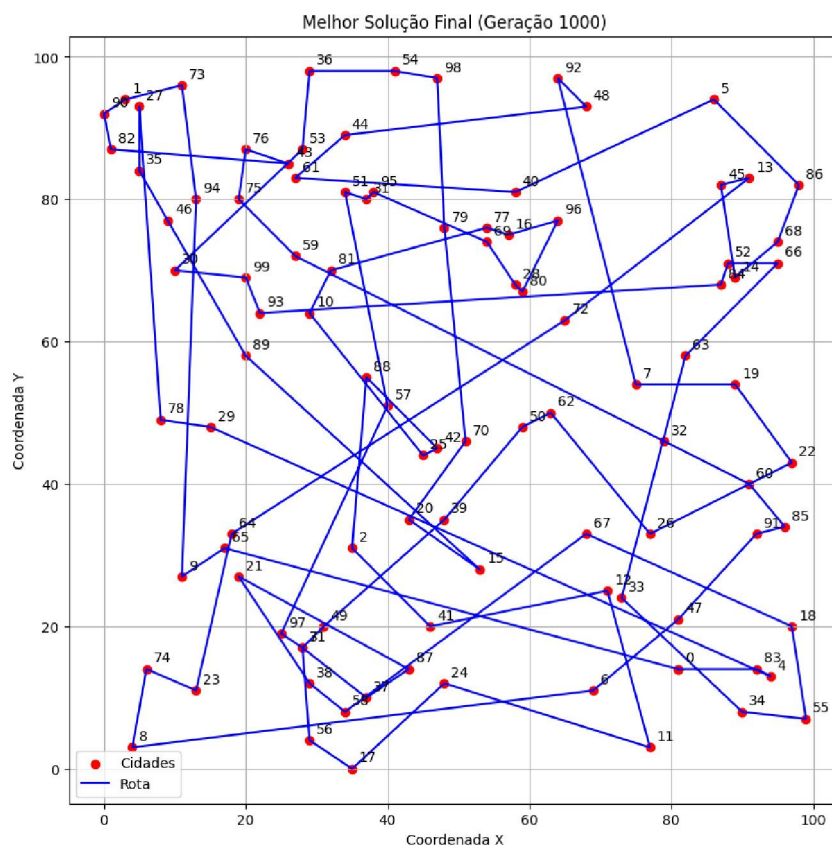
Melhor solução inicial (custo: 4836.21):

[0,44,51,93,73,79,4,23,25,43,99,87,62,61,5,83,18,
88,20,81,59,69,91,60,53,8,67,39,41,95,3,27,78,46,9,82,36,54,2,22,92,72,35,63,11,37,96,
77,12,57,75,49,65,74,21,70,84,71,6,32,16,24,29,48,14,34,55,52,50,33,26,85,76,7,45,15,
17,1,40,98,86,68,56,89,10,13,42,30,38,66,28, 94,97,90,19,58,80,64,31,47]



Melhor solução final encontrada (custo: 1895.59): [0,83,4,29,78,27,35,46,

89,15,20,70,79,98,54,36,53,30,99,93,84,52,66,63,33,34,55,18,67,58,38,21,
 87,37,71,97,57,51,31,95,69,28,80,96,16,77,81,10,25,42,88,2,41,12,11,24,17
 ,56,3,49,39,50,62,26,22,19,7,92,48,44,61,40,5,86,68,14,45,13,72,64,23,74,
 8,6,47,91,85,60,32,59,75,76,43,82,90,1,73,94,9,65]



Questão 2

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer

# Definição de textos
textos = [
    "O cachorro correu pelo parque.",
    "O cão brincou com a bola no parque.",
    "A criança jogou a bola para o cachorro.",
```



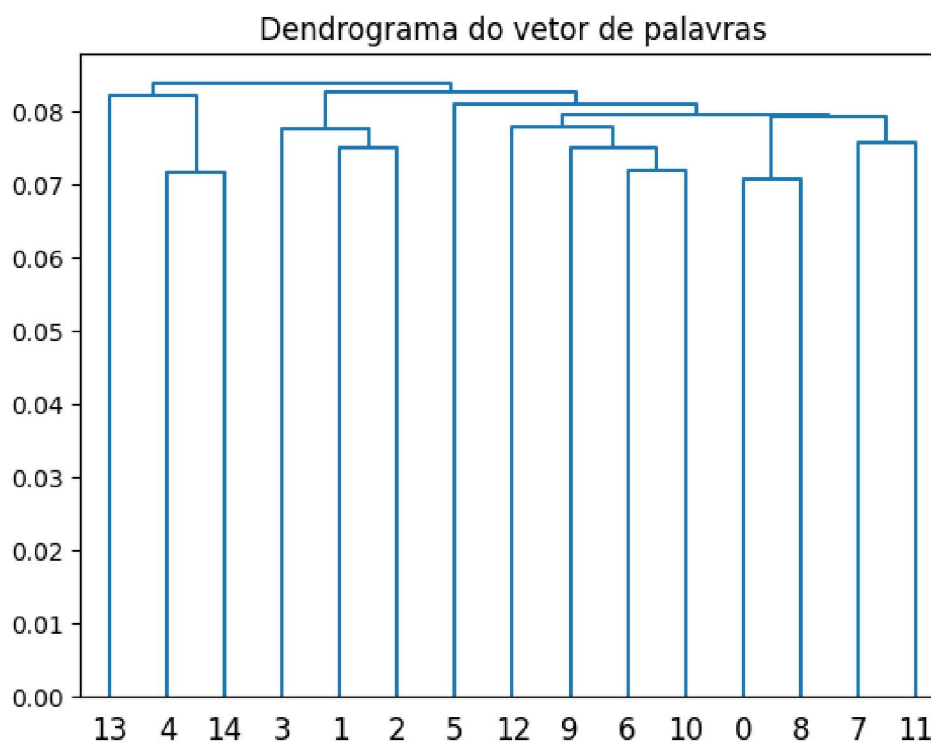
```
"O gato dormiu o dia todo.",
"O felino descansava tranquilamente.",
"A bola rolou para longe do parque."
]

# Vetorização usando TF-IDF
vectorizer = TfidfVectorizer()
vetores = vectorizer.fit_transform(textos).toarray()

# Redução de dimensionalidade com PCA
pca = PCA(n_components=2)
vetores_pca = pca.fit_transform(vetores)

# Plotando os vetores
plt.figure(figsize=(8, 6))
for i, txt in enumerate(textos):
    plt.scatter(vetores_pca[i, 0], vetores_pca[i, 1])
    plt.annotate(f"Texto {i+1}", (vetores_pca[i, 0], vetores_pca[i, 1]))

plt.title("Projeção dos Textos via PCA")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.grid()
plt.show()
```



Aplicando o PCA - Plotando o Resultado

