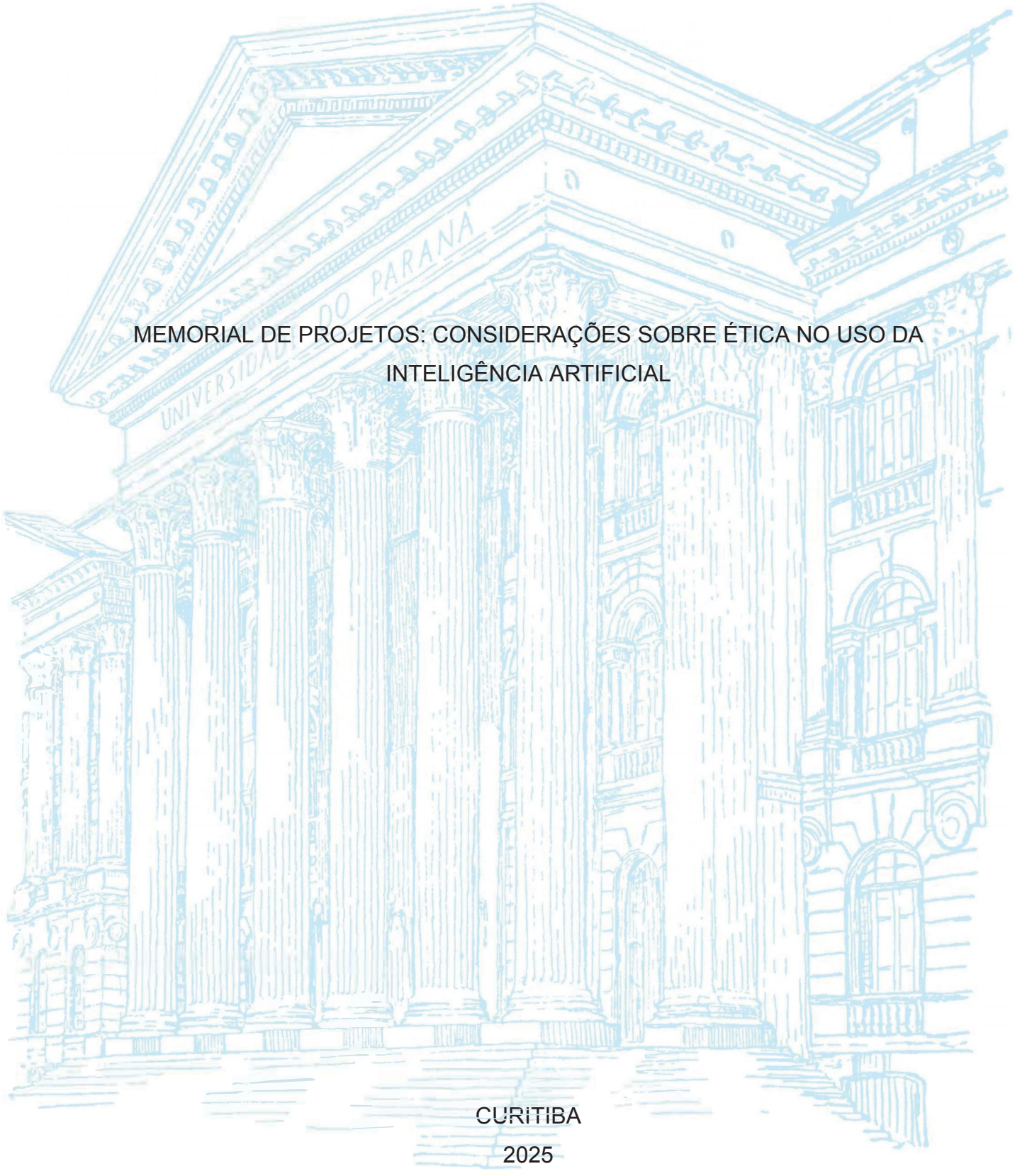
UNIVERSIDADE FEDERAL DO PARANÁ

GUILHERME DE MATOS KROGER



MEMORIAL DE PROJETOS: CONSIDERAÇÕES SOBRE ÉTICA NO USO DA INTELIGÊNCIA ARTIFICIAL

CURITIBA

2025

GUILHERME DE MATOS KROGER

MEMORIAL DE PROJETOS: CONSIDERAÇÕES SOBRE ÉTICA NO USO DA INTELIGÊNCIA ARTIFICIAL

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

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **GUILHERME DE MATOS KROGER**, intitulada: **MEMORIAL DE PROJETOS: CONSIDERAÇÕES SOBRE ÉTICA NO USO DA INTELIGÊNCIA ARTIFICIAL**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

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

Curitiba, 21 de Outubro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

O avanço acelerado da Inteligência Artificial (IA) traz consigo uma série de dilemas éticos e sociais que ultrapassam o campo técnico. Muito além do temor de uma IA Geral (AGI) hostil, os problemas mais urgentes já se manifestam nas tecnologias atuais, por meio de algoritmos que reforçam desigualdades, violam a privacidade e exploram o trabalho humano de forma precarizada. A chamada justiça algorítmica, baseada em modelos opacos e de difícil auditoria, ameaça a transparência e o direito à explicação. Além disso, a IA reproduz preconceitos de gênero, raça e classe, perpetuando exclusões históricas sob uma aparência de neutralidade técnica. A dependência de cadeias produtivas extrativistas e o consumo intensivo de energia contradizem o discurso de uma "tecnologia limpa". A consolidação do capitalismo de vigilância transforma dados pessoais em mercadoria e amplia o controle social. Diante disso, uma IA ética exige não apenas ajustes técnicos, mas a transformação estrutural das relações econômicas e políticas, com regulamentação efetiva e participação democrática na sua governança.

Palavras-chave: inteligência artificial; ética; automação; justiça algorítmica; capitalismo de vigilância.

ABSTRACT

The rapid advancement of Artificial Intelligence (AI) brings a wide range of ethical and social challenges that go far beyond technical concerns. More than the fear of a hostile Artificial General Intelligence (AGI), the most urgent issues are already visible in current technologies: algorithms that reproduce social inequalities, systems that violate privacy, and infrastructures that exploit precarious labor. Algorithmic justice often functions as an opaque “black box,” limiting transparency and the right to explanation. Furthermore, AI reflects and amplifies gender, racial, and class biases, reinforcing historical exclusions under the guise of objectivity. Its dependence on extractive production chains and high energy consumption contradicts the notion of a “clean technology.” At the same time, the rise of surveillance capitalism transforms personal data into a commodity, shaping behavior and undermining autonomy. Therefore, building ethical AI requires more than technical fixes; it demands a structural transformation of technological development, with effective regulation, social accountability, and democratic participation in governance.

Keywords: artificial intelligence; ethics; automation; algorithmic justice; surveillance capitalism.

SUMÁRIO

1 PARECER TÉCNICO

A despeito da existência de outros temas consideravelmente relevantes, como o *Deep Learning*, por exemplo, considerando-se que o uso catastrófico de tecnologias promissoras possui vários precedentes na história humana, conforme Rhodes (1986), torna-se imperativo que se trate das questões éticas antes da incorporação massiva de inovações com potencial disruptivo, como a Inteligência Artificial (IA).

A esse respeito, Brynjolfsson (2014) considera que a IA se tornou um símbolo do avanço tecnológico no século XXI, mais importante que qualquer coisa desde a revolução industrial. Contudo, o entusiasmo generalizado muitas vezes esconde as consequências sociais, éticas e ambientais que seu uso pode acarretar. A pergunta "a IA vai destruir o mundo?" traduz o medo da perda de controle, da desumanização e da dominação por máquinas. Mais realistas, porém, são os impactos cotidianos da IA já em uso — como a automação de decisões judiciais, um ponto criticado por Ribeiro (2024), o monitoramento invasivo de usuários, conforme detalhado por Zuboff (2019), e a substituição de trabalhadores por sistemas digitais, tema da matéria de McGowan (2025).

Nesse contexto, observa-se que a aplicação de algoritmos em decisões judiciais e administrativas tem despertado sérias preocupações com a transparência, a imparcialidade e o direito à defesa. Conforme analisa Ribeiro (2025), a substituição da interpretação judicial por estatísticas automatizadas aproxima o sistema de justiça de um cenário kafkiano, em que o cidadão é julgado sem compreender os critérios da decisão. A esse respeito, a filósofa Kate Vredenburg (2021), adverte sobre o risco de decisões opacas, sem possibilidade de contestação, fato este que compromete os direitos fundamentais dos indivíduos.

Esse tipo de opacidade se agrava quando os modelos de IA operam como *caixas-pretas*, dificultando uma auditoria compreensível para o público ou mesmo para os desenvolvedores, conforme Alves (2022). Embora o direito à explicação, previsto no Regulamento Geral sobre a Proteção de Dados (GDPR), represente um avanço legal importante, de acordo com Goodman (2017), ele se mostra insuficiente frente à complexidade técnica de muitos sistemas atuais.

Essa falta de transparência, consoante Guestrin (2016), não é o único desafio ético. A IA também reproduz as escolhas, crenças e preconceitos de seus

desenvolvedores e da sociedade em que está inserida. O caso do sistema de recrutamento da Amazon, que penalizava candidatas mulheres, é ilustrativo dos riscos de reprodução de desigualdades a partir de dados enviesados. Também são preocupantes as tecnologias de segurança baseadas em gênero binário, que causam constrangimentos a pessoas trans ao serem interpretadas como anomalias. Como alerta Keyes (2018), sistemas que categorizam identidades ignorando a auto identificação atentam contra direitos fundamentais e reforçam exclusões históricas.

Além disso, é importante reconhecer que a IA depende de uma infraestrutura material que inclui mineração de recursos, consumo de energia e cadeias globais de produção intensivas em mão de obra precarizada. Como observa Jones (2021), a automação não elimina o trabalho humano, mas o reconfigura sob novas formas fragmentadas, mal remuneradas e invisíveis. Trabalhadores alocados em países pobres, campos de refugiados ou regiões periféricas do globo são responsáveis por tarefas essenciais ao funcionamento da IA, como a rotulagem de dados, sem receberem o devido reconhecimento.

Esse cenário se agrava com a inserção da IA em um modelo econômico baseado na exploração massiva de dados pessoais: o capitalismo de vigilância. Conforme argumenta Zuboff (2019), plataformas digitais transformam interações cotidianas em dados exploráveis, moldando comportamentos e fragilizando a autonomia dos indivíduos. A personalização de conteúdo, longe de representar um benefício neutro, reforça bolhas informacionais, amplia desigualdades e compromete o debate público.

Diante de tais desafios, é imperioso discutir caminhos para uma IA ética e socialmente justa. Soluções meramente técnicas, como ajustes nos algoritmos ou a diversificação de dados, são insuficientes sem uma transformação estrutural do modelo de desenvolvimento tecnológico. Propostas como a democracia de dados, em que os indivíduos controlam o uso de suas informações, ou o socialismo digital, no qual empresas tecnológicas são geridas democraticamente, apontam alternativas mais sustentáveis. A aplicação efetiva de leis de direitos autorais, transparência algorítmica e proteção de dados também são medidas essenciais.

Em síntese, a ética da IA não se resume a tentar ensinar valores às máquinas, mas a reconfigurar os valores da sociedade que as desenvolve. Uma tecnologia verdadeiramente ética depende de condições materiais dignas para quem a produz,

participação popular e distribuição equitativa de benefícios. Sem essa base, qualquer tentativa de ética computacional estará condenada a tangenciar o problema.

REFERÊNCIAS

ALVES, M. A. S.; ANDRADE, O. M. Da “caixa-preta” à “caixa de vidro”: o uso da explainable artificial intelligence (XAI) para reduzir a opacidade e enfrentar o enviesamento em modelos algorítmicos. **Revista Direito Público**, v. 18, n. 100, 2022.

BRYNJOLFSSON, E.; MCAFEE, A. **The Second Machine Age: Work, Progress, and Prosperity in a Time of Brilliant Technologies**. New York: W. W. Norton & Company, 2014.

GOODMAN, B.; FLAXMAN, S. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". **AI Magazine**, v. 38, n. 3, 2017.

GUESTRIN, C.; RIBEIRO, M. T.; SINGH, S. **"Why Should I Trust You?": Explaining the Predictions of Any Classifier.** Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2016. Disponível em: https://ucinlp.github.io/ Acesso em: 19 jul. 2025.

JONES, P. **Work Without the Worker: Labour in the Age of Platform Capitalism**. Londres: Verso Books, 2021.

KEYES, O. The Misgendering Machines: Trans/HCI Implications of Automatic Gender Recognition. **Proceedings of the ACM on Human-Computer Interaction**, v. 2, n. 88, 2018.

McGOWAN, C. **The workers who lost their jobs to AI**. The Guardian, 2025. Disponível em: https://www.theguardian.com/technology/2025/may/31/the-workers-who-lost-their-jobs-to-ai-chatgpt Acesso em: 19 jul. 2025.

RHODES, R. **The making of the atomic bomb**. London: Simon and Schuster, 1986.

RIBEIRO, F. O. **Sobre o uso de IA em decisões judiciais.** Jornal GGN, 2024. Disponível em: https://jornalggn.com.br/artigos/sobre-o-uso-de-ia-em-decisoes-judiciais-por-fabio-de-o-ribeiro/ Acesso em: 19 jul. 2025.

RIBEIRO, F. O. **Abram alas para o Direito algorítmico kafkiano francês**. Jornal GGN, 2025. Disponível em: https://jornalggn.com.br/cidadania/o-direito-algoritmico-kafkiano-frances-por-fabio-de-oliveira-ribeiro/ Acesso em: 01 jul. 2025.

VREDENBURG, K. The Right to Explanation. **Journal of Political Philosophy**, v. 30, n. 2, 2021.

ZUBOFF, S. **The Age of Surveillance Capitalism**. Nova York: Public Affairs, 2019.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

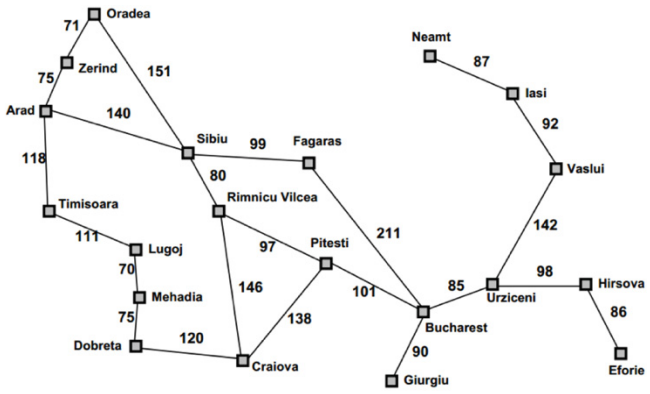
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 LÓGICA

Verificar se o argumento lógico é válido.

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

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

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

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

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

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

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

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

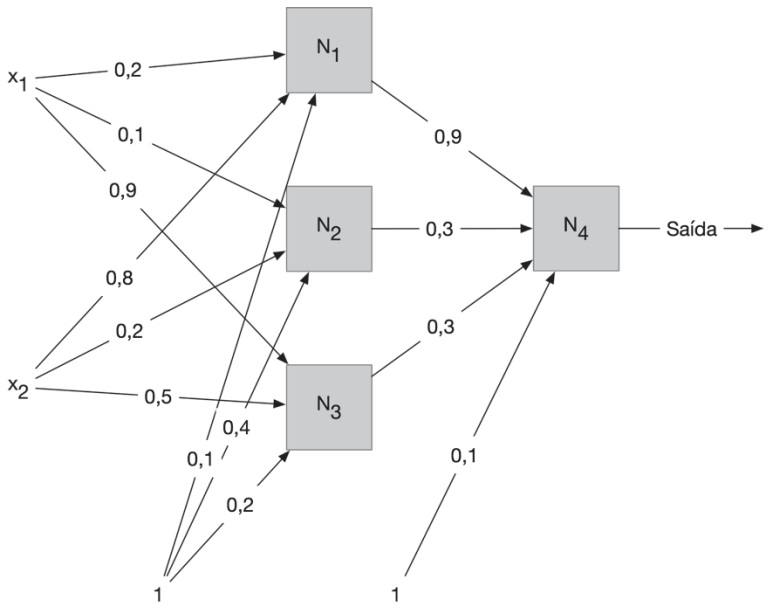
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

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

1 -

a) Inteligência artificial é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e algoritmos que podem realizar tarefas que normalmente exigiriam inteligência humana. Isso inclui aprendizado de máquina, processamento de linguagem natural, visão computacional e muito mais.

b) Classificação – Agir como humanos. De acordo com a resposta fornecida pelo chatGPT, a IA objetiva a realização de tarefas que exigiriam a inteligência humana, esta definição se enquadra no vetor “comportamento” (AGIR) e intersecciona o vetor “humano” (Inteligência humana / Fidelidade ao comportamento humano), resultando, assim, na classificação supracitada.

c) O Chat-GPT[1] (“Generative Pre-trained Transformer”), software lançado em novembro de 2022, é um LLM (“Large Language Model”) desenvolvido pela empresa Open AI, financiada principalmente pela Microsoft, ele possui a capacidade de responder a perguntas “abertas” sobre virtualmente qualquer tópico com dados disponíveis na web, sem a necessidade de recorrer a respostas codificadas manualmente. Utiliza-se de um modelo de rede neural treinada em dados de texto obtidos na internet, livros, wikipedia etc, baseado em aprendizado supervisionado e de reforço “Reinforcement Learning from Human Feedback” (RLHF)[2], nesta técnica, um humano classifica e ordena múltiplas respostas fornecidas pelo modelo, recompensando-o quando apresenta melhora, o processo é repetido várias vezes, no chamado “fine tunning”. A partir do treinamento, o modelo deve prever e gerar a sequência de palavras mais provável baseando-se nos padrões linguísticos dos dados (probabilidade condicional de palavras), observando o contexto fornecido na pergunta e resultando num estilo de escrita que mimetiza o humano. Mais detalhadamente, após o “prompt” do usuário, os dados de entrada são divididos em unidades menores ou “tokens” [3], que podem ser caracteres, palavras ou mesmo frases. A seguir, os “tokens” são convertidos em “embeddings”4 ou incorporações, representações de valores ou objetos - vetores com a finalidade de capturar dados significativos sobre cada objeto. Posteriormente, a sequência de embeddings de tokens é processada (cálculo da probabilidade das palavras seguintes) e

dela se extrai uma representação “interna“ do texto. Esta será processada e retornará como resposta ao “prompt” do usuário após efetuado o processo reverso - a decodificação e “destokenização”, conversão em texto. Em resumo, o chat-GPT, assim como os demais modelos baseados em linguagem natural, possuem a capacidade de “interagir” com as pessoas estabelecendo conversações, a partir do treinamento em grandes massas de dados de texto, aplicação de métodos estatísticos para extrair correlações que permitem predizer as próximas palavras a se utilizar nas frases, mimetizando os processos envolvidos na linguagem humana, sem, no entanto, nenhuma compreensão dos temas tratados em tais diálogos.

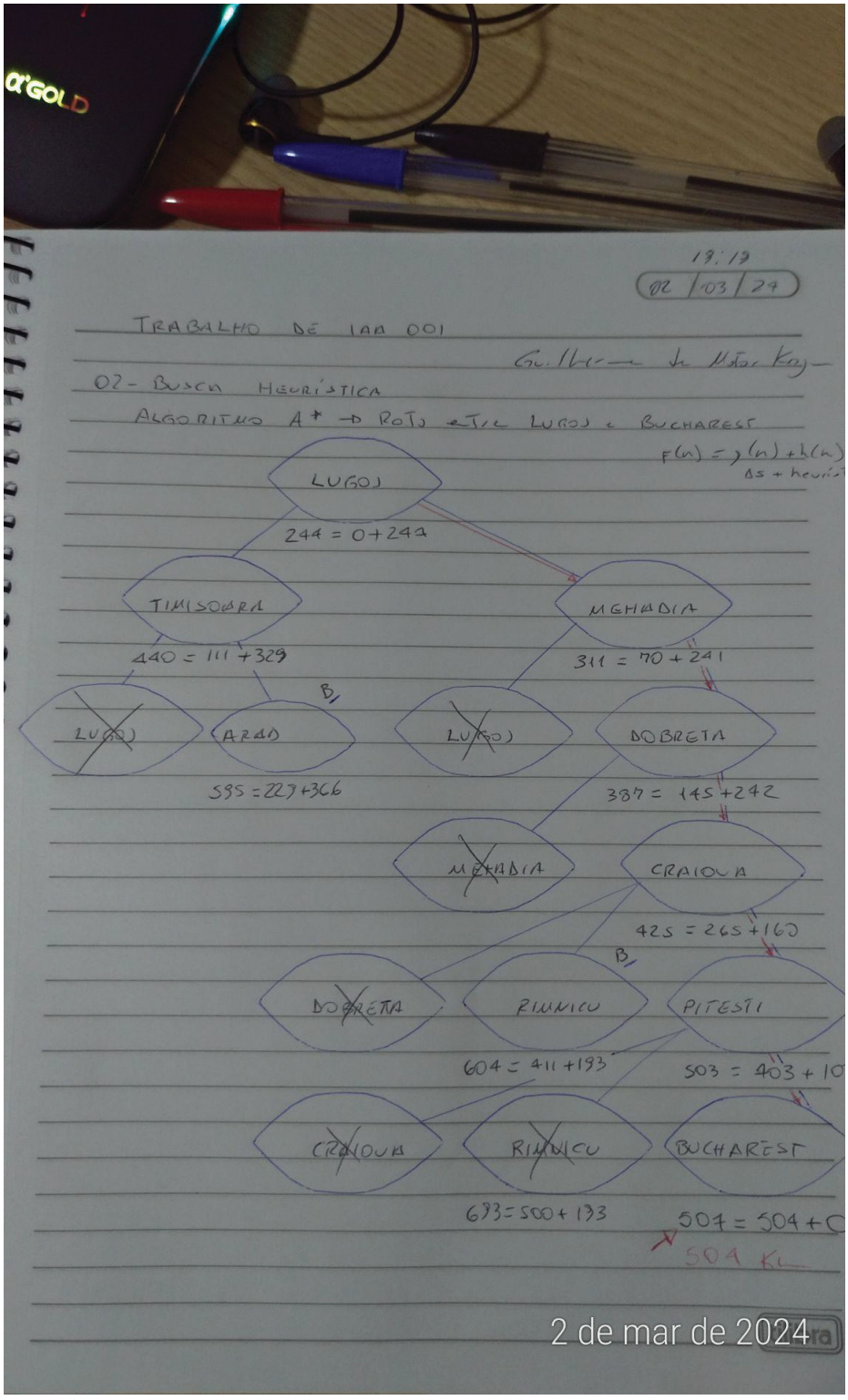
Fontes:
[1] https://pt.wikipedia.org/wiki/ChatGPT
[2] https://openai.com/blog/chatgpt
[3] https://encurtador.com.br/kIRW1
[4] https://www.cloudflare.com/pt-br/learning/ai/what-are-embeddings/

d) Analisando-se as abordagens da inteligência artificial formuladas por Norvig e Russell, observa-se que o chat-GPT apresenta características que poderiam ser atribuídas a mais de uma delas, a saber: Na categoria “Pensar como os Humanos” não há enquadramento, aqui seria necessário desvendar o próprio funcionamento da mente humana para depois replicá-lo em máquinas. Exige Introspecção do próprio sistema, o que não se aplica ao GPT. Por sua vez, “Pensar Racionalmente” e sua busca de racionalidade perfeita, apesar do modelo ser capaz de raciocinar logicamente, tomar decisões com base nas informações que recebe e resolver problemas com eficiência; esbarra nas dificuldades trazidas pela incompletude e incerteza do mundo real. Algumas coincidências ocorrem na abordagem “Agir como os Humanos”, pois o sistema é capaz de gerar texto em estilo humano, traduzir idiomas e responder a questionamentos de maneira natural e fluida, atendendo assim a alguns requisitos do Teste de Turing (processamento de linguagem natural, representação do conhecimento, raciocínio e aprendizagem). Porém, a abordagem que mais se aproxima é a da “Ação Racional”, nela são implementados os agentes racionais, que percebem o ambiente, respondem a situações e buscam o melhor resultado possível, tomam decisões com base nos objetivos que lhe são dados, fazendo inferências e também são capazes de planejar ações e executá-las de forma eficiente. Dessa forma, o agente racional age “para alcançar o melhor resultado ou, quando há incerteza, o melhor resultado esperado.”[1] É exatamente isso que se observa no chatGPT ao responder perguntas, construir textos diversos e reformular as respostas quando contradito

[1] Norvig & Russell – Inteligência Artificial – Uma Abordagem Moderna

2 -

a)


19:13
02/03/24
TRABALHO DE IAA 001
02- BUSCA HEURÍSTICA
ALGORITMO A* → ROTA LUGOJ - BUCHAREST
LUGOJ
244 = 0+244
TIMISOARA
MEHADIA
440 = 111+329
311 = 70+241
ARAD
DOBRETA
595 = 223+366
387 = 145+242
CRAIOVA
425 = 265+160
RIMNICU
PITESTI
604 = 411+193
503 = 403+10
BUCHARÉST
693 = 500+193
504 KL
2 de mar de 2024

3 -

Objetivo - q ↔ p

Prova:

R1: p → q
R2: q → r
R3: ¬r ∨ p

R4: p → r SILOGISMO HIPOTÉTICO R1, R2
R5: r → p EQUIVALÊNCIA IMPLICAÇÃO R3
R6: (p → r) ∧ (r → p) CONJUNÇÃO R4, R5
R7: p ↔ r EQUIVALÊNCIA BICONDICIONAL R6
R8: q → p SILOGISMO HIPOTÉTICO R2, R5
R9: (q → p) ∧ (p → q) CONJUNÇÃO R8, R1
R10: q ↔ p EQUIVALÊNCIA BICONDICIONAL R9

4 -

a) Valor de saída do neurônio N1 = 0,3 u=xi∗wi+b u=(−3∗0,2)+(1∗0,8)+(1∗0,1) u=−0,6+0,8+0,1 u=0,3 fa(u)=u fa(0,3)=0,3

b) Valor de saída do neurônio N2 = 0,3 u=xi∗wi+b u=(−3∗0,1)+(1∗0,2)+(1∗0,4) u=−0,3+0,2+0,4 u=0,3 fa(u)=u fa(0,3)=0,3

c) Valor de saída do neurônio N3 = -2 u=xi∗wi+b u=(−3∗0,9)+(1∗0,5)+(1∗0,2) u=−2,7+0,5+0,2 u=−2 fa(u)=u fa(−2)=−2

d) Valor de saída da rede como um todo N4 = -0,139092448 u=xi∗wi+b u=(0,3∗0,9)+(0,3∗0,3)+(−2∗0,3)+(1∗0,1) u=0,27+0,09+(−0,6)+0,1 u=−0,14 fa(u)=eu−e−u/(eu+e−u) fa(−0,14)=−0,139092448

Função Linear

$$f_a(u) = u$$

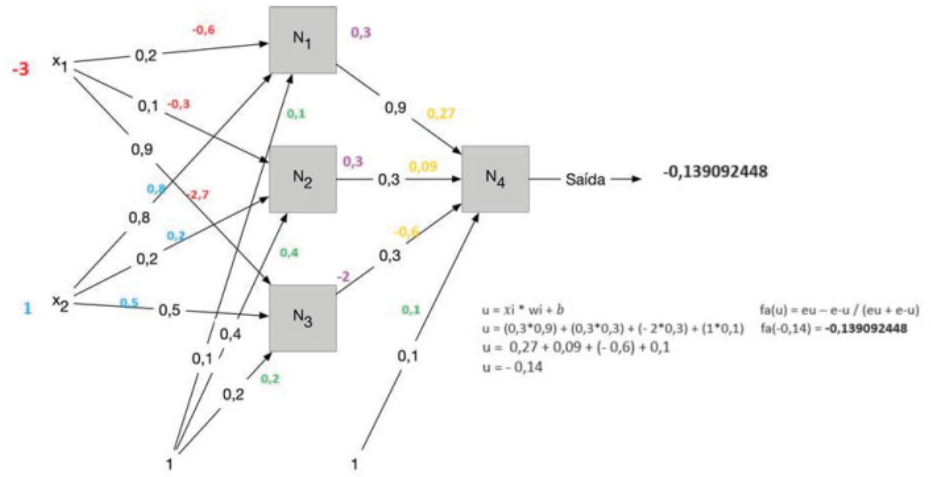
Função Tangente-Hiperbólica

$$f_a(u) = \frac{e^u - e^{-u}}{(e^u + e^{-u})}$$

$$u = \sum_{i=1}^{n}(x_i \times w_i)$$

$$Saída = f_a(u)$$

$$Saída = \begin{cases} 0, & se \sum_{i=1}^{n}(x_i w_i) + b < 0 \\ 1, & se \sum_{i=1}^{n}(x_i w_i) + b \geq 0 \end{cases}$$



APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

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

B - RESOLUÇÃO

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

1 -

a)

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

# machine learning
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import LabelEncoder

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

```
dados = pd.read_csv('precos_carros_brasil.csv', low_memory=False)
# Columns (1,2,3,4,5,6,7,8) have mixed types. Specify dtype option on import or
set low_memory=False.
```

```
dados.columns
Index(['year_of_reference', 'month_of_reference', 'fipe_code',
       'authentication', 'brand', 'model', 'fuel', 'gear', 'engine_size',
       'year_model', 'avg_price_brl'],
      dtype='object')
```

```
dados.head()

year_of_reference    month_of_reference    fipe_code      authentication brand
0     2021.0  January 004001-0       cfzlctzfwrcp   GM - Chevrolet
1     2021.0  January 004001-0       cdqwxwpw3y2p   GM - Chevrolet
```

```
        2     2021.0  January 004001-0       cb1t3xwwj1xp   GM - Chevrolet
        3     2021.0  January 004001-0       cb9gct6j65r0   GM - Chevrolet
        4     2021.0  January 004003-7       g15wg0gbz1fx   GM - Chevrolet

        model fuel   gear   engine_size   year_model     avg_price_brl
        0 Corsa Wind 1.0 MPFI / EFI 2p     Gasoline       manual 1      2002.0 9162.0
        1 Corsa Wind 1.0 MPFI / EFI 2p     Gasoline       manual 1      2001.0 8832.0
        2 Corsa Wind 1.0 MPFI / EFI 2p     Gasoline       manual 1      2000.0 8388.0
        3 Corsa Wind 1.0 MPFI / EFI 2p     Alcohol manual 1      2000.0 8453.0
        4 Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI  Gasoline manual 1,6   2001.0 12525.0
```

b)

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

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

```
dados.isna().sum()

year_of_reference     65245
month_of_reference    65245
fipe_code             65245
authentication        65245
brand                 65245
model                 65245
fuel                  65245
gear                  65245
engine_size           65245
year_model            65245
avg_price_brl         65245
dtype: int64
```

```
#Removendo linhas com todos os valores em branco:
dados.dropna(how='all', inplace=True)
```

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

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

```
dados.isna().sum()

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

c)

```
#dados.duplicated() .sum()
# Mostrando as linhas duplicadas com suas contagens
dados.groupby(list(dados.columns)).size().to_frame('Contagem').reset_index().query('Contagem
> 1')

        year_of_reference    month_of_reference    fipe_code      authenticationbrand
55840   2021-01-01     June    025232-8       5rtdwkpkpq5h   Renault
114950  2022-01-01     December       003296-4       3r6c277cnqcb   Ford

model   fuel    gear    engine_size    year_model     avg_price_brl Contagem
DUSTER OROCH Dyna. 2.0 Flex 16V Mec. Gasoline       manual 2       2018.0 69893.02
Ranger Limited 3.0 PSE 4x4 CD TB Diesel     Diesel manual 3       2007.0 64638.02

dados.drop_duplicates( inplace = True)
```

d)

```
#Criar categorias para separar dados numéricos de categóricos

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

```
#dados = dados.loc[dados['year_of_reference'] == 2021]

dados[numericas_cols].describe().round(0)
```

	year_of_reference	year_model	avg_price_brl
count	202295	202295.0	202295.0
mean	2021-07-26 02:43:45.413381632	2011.0	52757.0
min	2021-01-01 00:00:00	2000.0	6647.0
25%	2021-01-01 00:00:00	2006.0	22855.0
50%	2022-01-01 00:00:00	2012.0	38027.0
75%	2022-01-01 00:00:00	2016.0	64064.0
max	2023-01-01 00:00:00	2023.0	979358.0
std	NaN	6.0	51629.0

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

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

e)

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

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
```

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

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

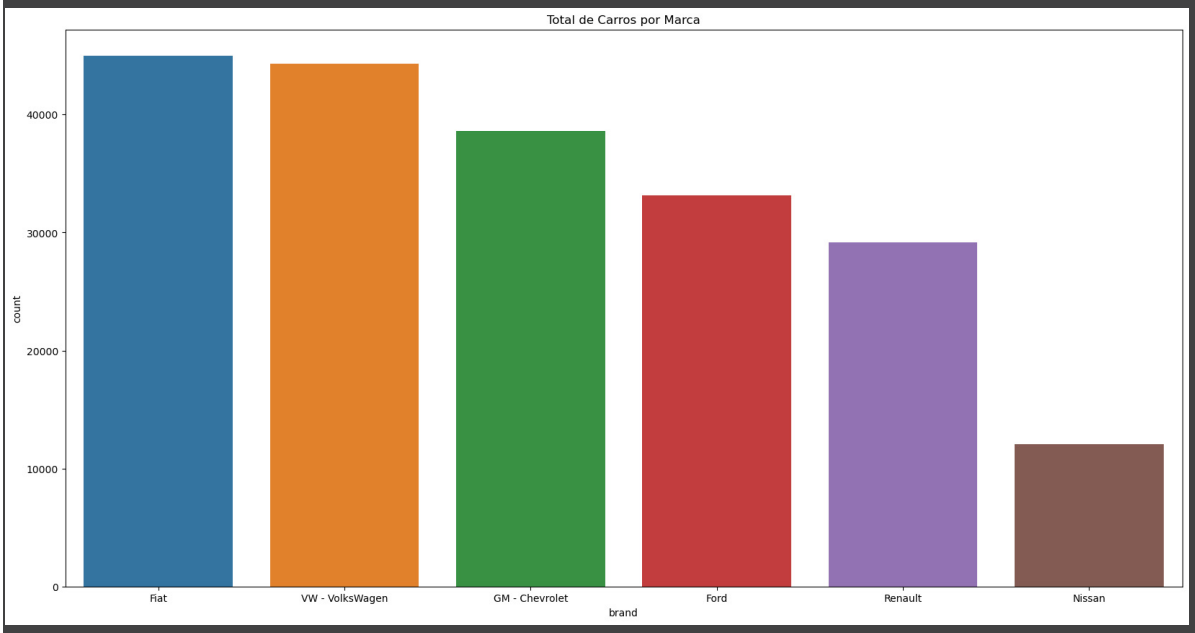
f)

```
Total de itens válidos 202.295, Menor valor médio encontrado 6647.0, valor
médio 52757.0, maior valor 979358.0 6 Fabricantes listados, 2.112 modelos, 29
tipos de motorização, 3 tipos de combustível, faixa de modelos de 2000 a 2023
 Preços médios por Quartis ( 25% 22855.0 - 50% 38027.0 - 75% 64064 o)</spa))
```

2 -

a)

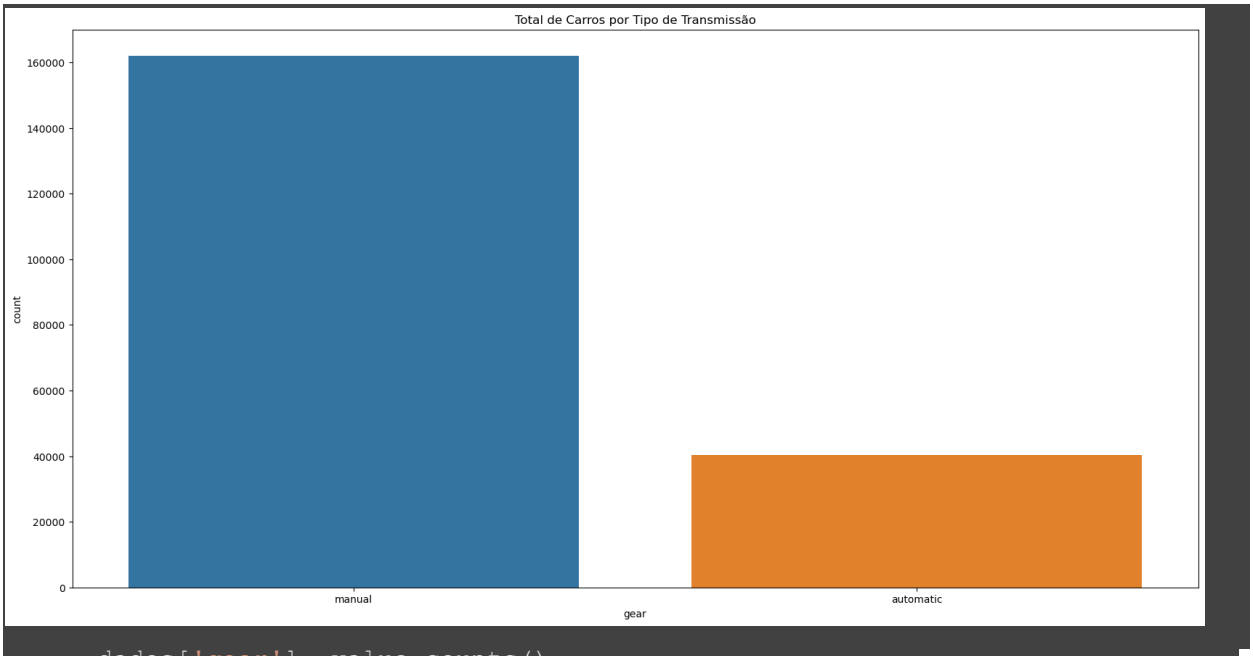
```
        # sns.countplot para criar gráfico de barras - x contagem, y ordenação
        # index para ordenar os valores

        plt.figure(figsize=(20,10))
        sns.countplot(x="brand", data=dados,
order=dados['brand'].value_counts().index).set_title('Total de Carros por Marca')
        ;
```




```
dados['brand']. value_counts()
brand
Fiat                44962
VW - VolksWagen     44312
GM - Chevrolet      38590
Ford                33150
Renault             29191
Nissan              12090
Name: count, dtype: int64
```

b)

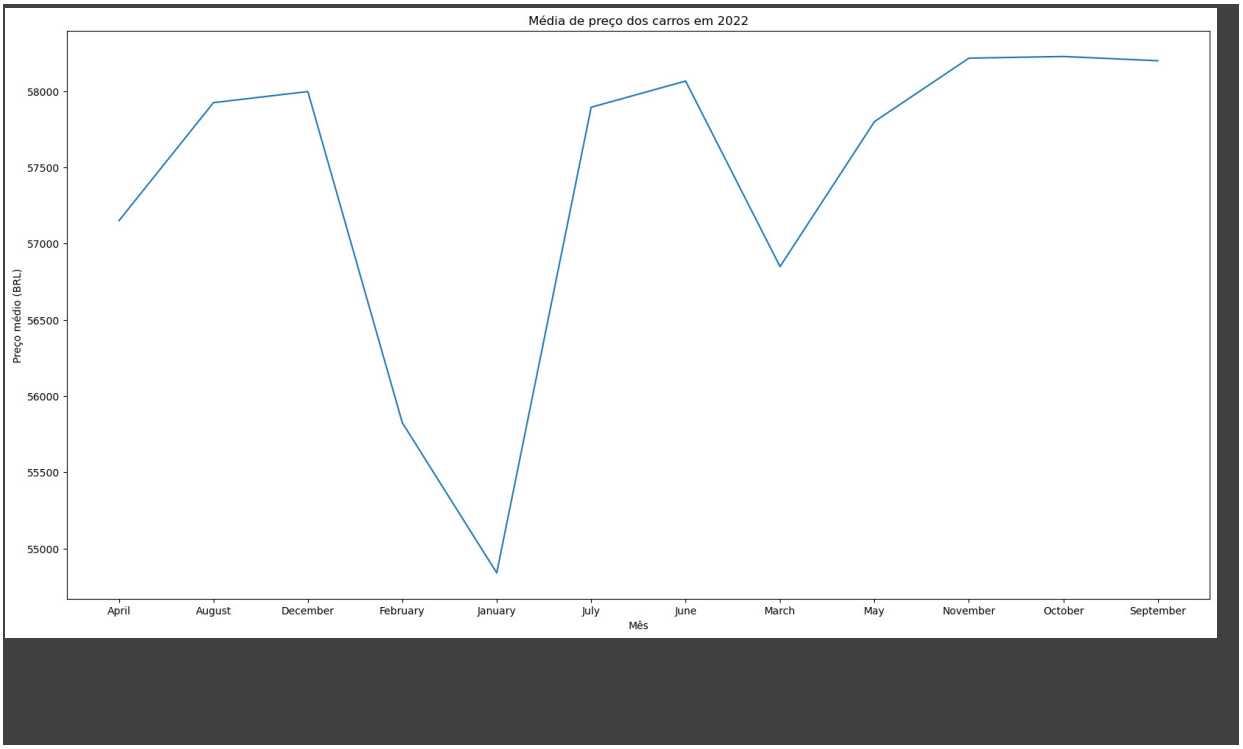
```
        # sns.countplot para criar gráfico de barras - x contagem, y ordenação
        # index para ordenar os valores

        plt.figure(figsize=(20,10))
        sns.countplot(x="gear", data=dados,
order=dados['gear'].value_counts().index).set_title('Total de Carros por Tipo de
Transmissão')
        ;
```




```
dados['gear']. value_counts()
gear
manual       161883
automatic     40412
Name: count, dtype: int64
```

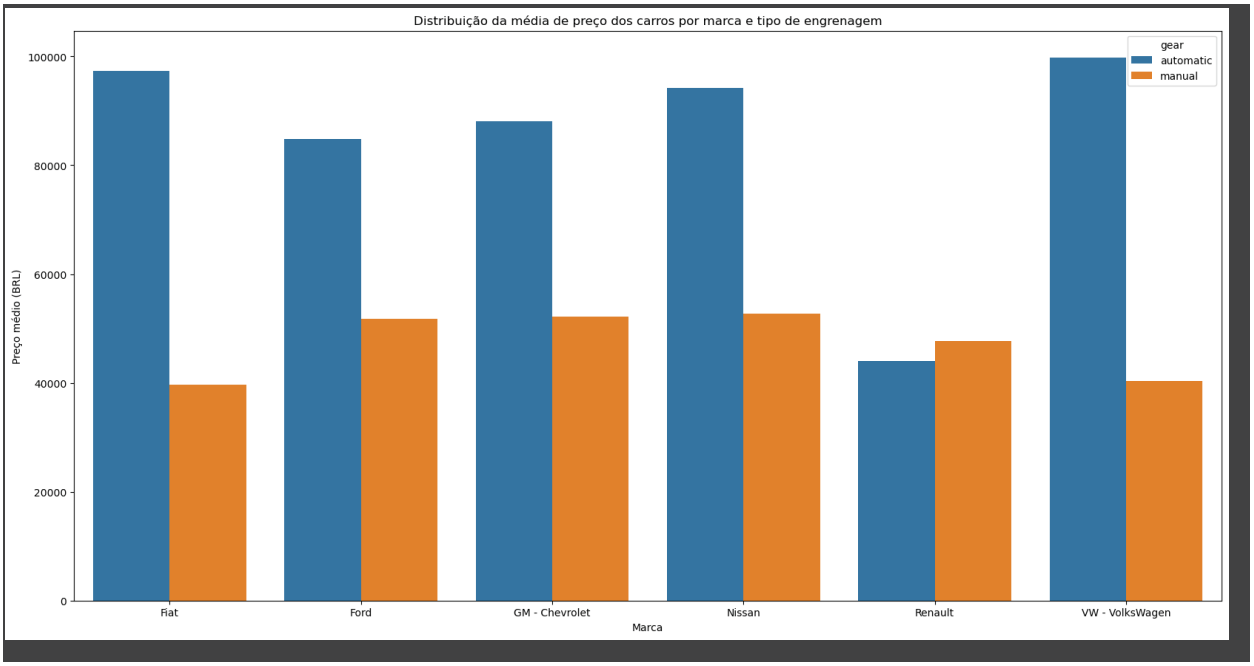
c)

```
dados_mensal = dados.loc[dados['year_of_reference'] == '2022']
dados_mensal = dados_mensal.groupby('month_of_reference')['avg_price_brl'].mean().round()
plt.figure(figsize=(20,10))
plt.plot(dados_mensal.index, dados_mensal.values)
plt.xlabel('Mês')
plt.ylabel('Preço médio (BRL)')
plt.title('Média de preço dos carros em 2022')
plt.show()
```



d)

```
dados_marcas_cambio = dados.groupby(['brand', 'gear'])['avg_price_brl'].mean().reset_index()
plt.figure(figsize=(20,10))
sns.barplot(x=dados_marcas_cambio['brand'], y=dados_marcas_cambio['avg_price_brl'],
hue=dados_marcas_cambio['gear'])
plt.xlabel('Marca')
plt.ylabel('Preço médio (BRL)')
plt.title('Distribuição da média de preço dos carros por marca e tipo de engrenagem')
plt.show()
```




```
dados[['brand', 'gear']].value_counts()
brand            gear
Fiat             manual       42069
VW - VolksWagen  manual       38644
GM - Chevrolet   manual       27828
Ford             manual       24876
Renault          manual       21346
GM - Chevrolet   automatic    10762
Ford             automatic     8274
Renault          automatic     7845
Nissan           manual        7120
VW - VolksWagen  automatic     5668
Nissan           automatic     4970
Fiat             automatic     2893
Name: count, dtype: int6
```

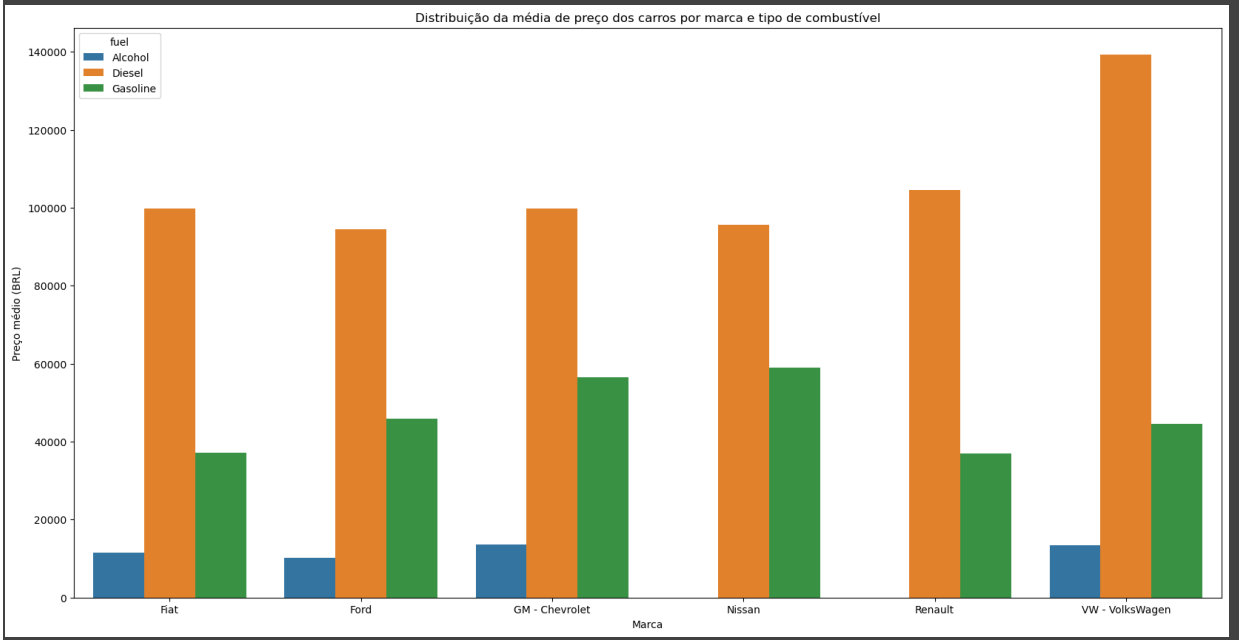
e)

```
Nota-se uma diferença acentuada ( para mais ) na média de preços dos carros equipados com câmbio automático, com exceção da fabricante Renault, o que pode indicar excessiva desvalorização dos veículos específicos desta marca Para as demais, a razão entre os preços é visivelmente proporcional..
```

f)

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

```
plt.figure(figsize=(20,10))
sns.barplot(x=dados_marcas_combustivel['brand'], y=dados_marcas_combustivel['avg_price_brl'],
hue=dados_marcas_combustivel['fuel'])
plt.xlabel('Marca')
plt.ylabel('Preço médio (BRL)')
plt.title('Distribuição da média de preço dos carros por marca e tipo de combustível')
plt.show()
```



g)

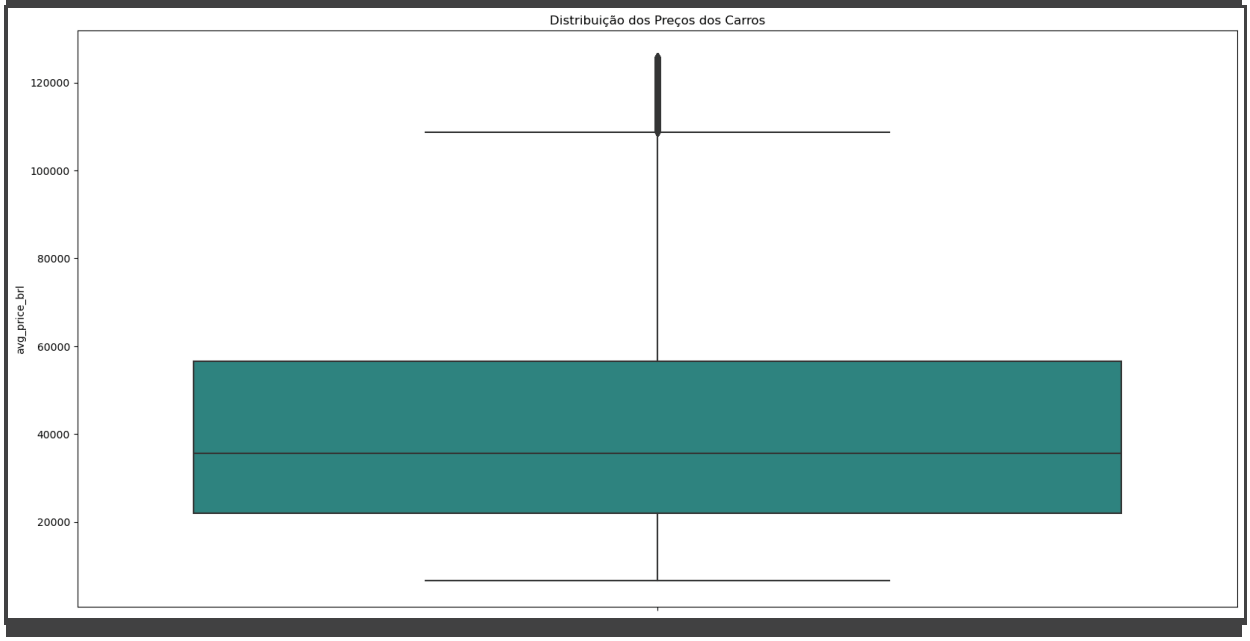
```
Veículos a diesel apresentam uma média de preço bem mais elevada que os demais, o que se
justifica pelo tipo de veículo normalmente movido a este combustível ( utilitários esportivos
e pick-ups ). Menores preços dos veículos a álcool se justificam pela fabricação mais antiga
e pela predominância dos modelos "populares" ( Palio, Gol, Corsa . Veículos a gasolina se
posicionam entre os extremos.a.
```

3 -

Removendo Outliers

```
# Seleciona a coluna de preço médio
coluna_preco = dados['avg_price_brl']

# Calcula o quartil 1 (Q1) e o quartil 3 (Q3)
Q1 = coluna_preco.quantile(0.25)
Q3 = coluna_preco.quantile(0.75)

# Calcula o Intervalo Interquartil (IQR)
```

```
IQR = Q3 - Q1

# Define os limites inferior e superior para outliers
limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

# Filtra o DataFrame
dados = dados[(dados['avg_price_brl'] >= limite_inferior) & (dados['avg_price_brl'] <=
limite_superior)]

# Imprime o DataFrame filtrado
dados.shape

(188528, 11)

#sns.boxplot(dados['avg_price_brl']).set_title("Distribuição dos Preços dos Carros");

plt.figure(figsize=(20,10))
sns.boxplot(y=dados['avg_price_brl'], palette = 'viridis')
plt.title("Distribuição dos Preços dos Carros")
plt.show()
```



a)

Variáveis Independentes Escolhidas - 'fipe_code','fuel','gear','engine_size'

```
dados.head()
```

year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine	year_model	avg_price_brl

									_size		
0	2021-01-01	January	004001-0	cfzlctzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2002.0	9162.0
1	2021-01-01	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2001.0	8832.0
2	2021-01-01	January	004001-0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000.0	8388.0
3	2021-01-01	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000.0	8453.0
4	2021-01-01	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline	manual	1,6	2001.0	12525.0

```
# "engine_size" - Converte as colunas para o tipo numérico - Substituir vífgulas por pontos
# Replace commas with periods and then convert to numeric
dados["engine_size"] = pd.to_numeric(dados["engine_size"].str.replace(',', '.'),
errors='coerce')
# fipe_code - Converte as colunas para o tipo numérico - Substituir traço por ponto
# Replace commas with periods and then convert to numeric
dados['fipe_code'] = pd.to_numeric(dados['fipe_code'].str.replace('-', '.'), errors='coerce')

#Utilizando LabelEncoder e fit_transform
#dados['fipe_code'] = LabelEncoder().fit_transform(dados['fipe_code'])

# gear - Utilizando LabelEncoder e fit_transform
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])

#fuel - Utilizando LabelEncoder e fit_transform
dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])
```

```
dados.head()
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
0	2021-01-01	January	4001.0	cfzlctzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1.0	2002.0	9162.0
1	2021-01-01	January	4001.0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1.0	2001.0	8832.0
2	2021-01-01	January	4001.0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	2	1	1.0	2000.0	8388.0
3	2021-01-01	January	4001.0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	0	1	1.0	2000.0	8453.0
4	2021-01-01	January	4003.7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	2	1	1.6	2001.0	12525.0

```
dados.dtypes
```

```
year_of_reference     datetime64[ns]
month_of_reference            object
fipe_code                    float64
authentication                object
brand                         object
model                         object
fuel                           int32
gear                           int32
engine_size                  float64
year_model                   float64
avg_price_brl                float64
dtype: object
```

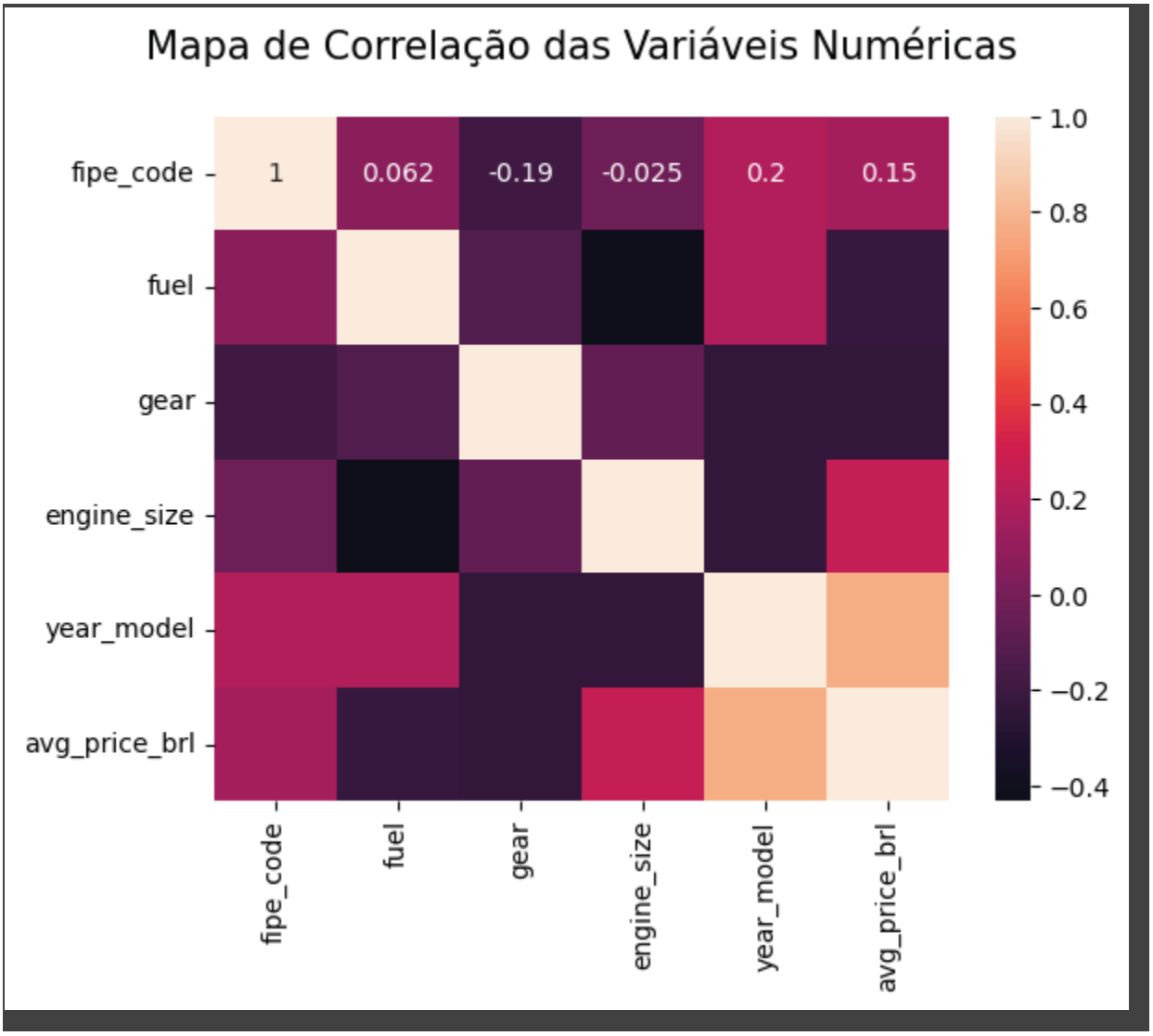
Mantendo apenas as variáveis numéricas

```
# Variável dados_num contém apenas variáveis numéricas de interesse (exclui o restante)
dados_num = dados.drop(['year_of_reference', 'month_of_reference',
      'authentication', 'brand', 'model'],axis = 1)
dados_num.head()
```

	fipe_code	fuel	gear	engine_size	year_model	avg_price_brl
0	4001.0	2	1	1.0	2002.0	9162.0
1	4001.0	2	1	1.0	2001.0	8832.0
2	4001.0	2	1	1.0	2000.0	8388.0
3	4001.0	0	1	1.0	2000.0	8453.0
4	4003.7	2	1	1.6	2001.0	12525.0

b)

```
# Mapa de correlação das variáveis numéricas com variável Target
sns.heatmap(dados_num.corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```




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

	fipe_code	fuel	gear	engine_size	year_model
0	4001.0	2	1	1.0	2002.0
1	4001.0	2	1	1.0	2001.0
2	4001.0	2	1	1.0	2000.0
3	4001.0	0	1	1.0	2000.0
4	4003.7	2	1	1.6	2001.0

```
# Variável Y contém apenas a variável target - Faixa Salarial
Y = dados_num['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
```

```
# 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(5)
```

```
(141396, 5)
```

	fipe_code	fuel	gear	engine_size	year_model
87070	5230.2	2	1	1.0	2007.0
91277	1471.0	2	1	1.3	2017.0
23088	5300.7	2	1	1.6	2012.0
91243	1453.2	1	0	2.0	2016.0
145664	4321.4	2	1	1.0	2011.0

```
# Observando os dados de teste
print(X_test.shape)
X_test.head(5)
```

```
(47132, 5)
```

	fipe_code	fuel	gear	engine_size	year_model
108867	3413.4	2	1	1.0	2019.0
192650	5166.7	2	1	1.8	2003.0
14648	5204.3	2	1	4.2	2012.0
83252	1494.0	2	1	1.0	2019.0
106763	1273.4	2	1	1.0	2014.0

c)

Random Forest

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

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

RandomForestRegressor
RandomForestRegressor()
```

XGBoost

```
model_xgboost = XGBRegressor()

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

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, ...)
```

d)

Random Forest

```
valores_preditos_rf = model_rf.predict(X_test)

# Valores preditos
valores_preditos_rf

array([ 49043.05234066,  18845.81343542,  98411.39885607, ...,
        20197.92684231, 109453.70386949,  54174.45132139])
```

XGBoost

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

```
array([ 48534.586,  19662.412,  98370.46 , ...,  20613.16 , 110101.62 ,
        56844.715], dtype=float32)
```

e)

Random Forest

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

	importance
year_model	0.580859
engine_size	0.263079
fipe_code	0.087390
fuel	0.042384
gear	0.026288

MSE - calcula o erro quadrático médio das predições do nosso modelo. Quanto maior o MSE, pior é o modelo. Mean Square Error

```
mse = mean_squared_error(Y_test, valores_preditos_rf)
mse.round()
```

24786770.0

O MAE calcula a média da diferença absoluta entre o valor predito e o valor real. Nesse caso, os erros são penalizados linearmente, ou seja, todos terão o mesmo peso na média. Mean Absolute Error

```
mae = mean_absolute_error(Y_test, valores_preditos_rf)
mae.round()
```

3397.0

O R^2 é uma métrica que varia entre 0 e 1 e é uma razão que indica o quão bom o nosso modelo. Quanto maior seu valor, melhor é o modelo

```
r2_score(Y_test, valores_preditos_rf).round(3)
```

0.965

XGBoost

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

	importance
year_model	0.350736
engine_size	0.262805
fuel	0.258725
gear	0.102001
fipe_code	0.025734

f)

Random Forest - O ano modelo foi avaliado com 58% de importância, tornando-se a variável mais importante, o que de fato faz bastante sentido se considerarmos uma situação real. Em segundo lugar, a motorização, com 26%, também relevante. Modelo com 8% não influenciou muito e combustível e câmbio foram considerados irrelevantes. Random Forest - Parâmetros apresentou resultados similares. No caso do xgboost, o ano modelo foi avaliado com 35% de importância. Em segundo lugar, a motorização com 26%. Combustível com 25%. O câmbio, com 10%, não influenciou muito e o fipe_code foi considerado irrelevante pelo modelo.

g)

MELHOR MODELO = RANDOM FOREST SEM PARÂMETROS SETADOS.

GEROU MENORES ERROS MSE E MAE, ALÉM DE MAIOR VALOR DE R^2 --> 96,5% >

```
#Random Forest
mse = mean_squared_error(Y_test, valores_preditos_rf)
mae = mean_absolute_error(Y_test, valores_preditos_rf)
print( mse.round(), mae.round(), r2_score(Y_test, valores_preditos_rf).round(3))
```

```
24786770.0 3397.0 0.965
```

```
# Random Forest - Parâmetros
mse_param = mean_squared_error(Y_test, valores_preditos_rf_parametros)
mae_param = mean_absolute_error(Y_test, valores_preditos_rf_parametros)
```

```
print( mse_param.round(), mae_param.round(), r2_score(Y_test,
valores_preditos_rf_parametros).round(3))

26436399.0 3478.0 0.963



#XGBoost
mse_xgb = mean_squared_error(Y_test, valores_preditos_xgboost)
mae_xgb = mean_absolute_error(Y_test, valores_preditos_xgboost)
r2_score(Y_test, valores_preditos_xgboost).round(3)
print( mse_xgb.round(), mae_xgb.round(), r2_score(Y_test, valores_preditos_xgboost).round(3))

39538207.0 4233.0 0.944
```

h)

Melhor Modelo = Random Forest SEM parâmetros setados.
Gerou menores erros MSE Erro Médio Quadrátic) 24.786 e MAE Erro Médio Absolut) 3.397.0,
além de maior valor de R² (proporção da variância dos dados) --> 96,50%
Acurácia geral alta, 96,5%. Demais modelos também apresentaram acurácia alta.

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

    - Coeficiente de determinação: R^2

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

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

    - Erro padrão da estimativa: S_{yx}

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

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

1.1 -

```
> install.packages("mlbench")
> library(mlbench)
> data(Satellite)
> str(Satellite)
'data.frame':   6435 obs. of  37 variables:
 $ x.1    : num  92 84 84 80 84 80 76 76 76 76 ...
 $ x.2    : num  115 102 102 102 94 94 102 102 89 94 ...
 $ x.3    : num  120 106 102 102 102 98 106 106 98 98 ...
 $ x.4    : num  94 79 83 79 79 76 83 87 76 76 ...
 $ x.5    : num  84 84 80 84 80 80 76 80 76 76 ...
 $ x.6    : num  102 102 102 94 94 102 102 98 94 98 ...
 $ x.7    : num  106 102 102 102 98 102 106 106 98 102 ...
 $ x.8    : num  79 83 79 79 76 79 87 79 76 72 ...
 $ x.9    : num  84 80 84 80 80 76 80 76 76 76 ...
 $ x.10   : num  102 102 94 94 102 102 98 94 98 94 ...
 $ x.11   : num  102 102 102 98 102 102 106 102 102 90 ...
 $ x.12   : num  83 79 79 76 79 79 79 76 72 76 ...
 $ x.13   : num  101 92 84 84 84 76 80 80 80 76 ...
 $ x.14   : num  126 112 103 99 99 99 107 112 95 91 ...
 $ x.15   : num  133 118 104 104 104 104 118 118 104 104 ...
 $ x.16   : num  103 85 81 78 81 81 88 88 74 74 ...
 $ x.17   : num  92 84 84 84 76 76 80 80 76 76 ...
```

```
 $ x.18   : num  112 103 99 99 99 99 112 107 91 95 ...
 $ x.19   : num  118 104 104 104 104 108 118 113 104 100 ...
 $ x.20   : num  85 81 78 81 81 85 88 85 74 78 ...
 $ x.21   : num  84 84 84 76 76 76 80 80 76 76 ...
 $ x.22   : num  103 99 99 99 99 103 107 95 95 91 ...
 $ x.23   : num  104 104 104 104 108 118 113 100 100 100 ...
 $ x.24   : num  81 78 81 81 85 88 85 78 78 74 ...
 $ x.25   : num  102 88 84 84 84 84 79 79 75 75 ...
 $ x.26   : num  126 121 107 99 99 103 107 103 91 91 ...
 $ x.27   : num  134 128 113 104 104 104 113 104 96 96 ...
 $ x.28   : num  104 100 87 79 79 79 87 83 75 71 ...
 $ x.29   : num  88 84 84 84 84 79 79 79 75 79 ...
 $ x.30   : num  121 107 99 99 103 107 103 103 91 87 ...
 $ x.31   : num  128 113 104 104 104 109 104 104 96 93 ...
 $ x.32   : num  100 87 79 79 79 87 83 79 71 71 ...
 $ x.33   : num  84 84 84 84 79 79 79 79 79 79 ...
 $ x.34   : num  107 99 99 103 107 107 103 95 87 87 ...
 $ x.35   : num  113 104 104 104 109 109 104 100 93 93 ...
 $ x.36   : num  87 79 79 79 87 87 79 79 71 67 ...
 $ classes: Factor w/ 6 levels "red soil","cotton crop",..: 3 3 3 3 3 3 3 3 4 4 …

> summary(Satellite)
      x.1             x.2             x.3             x.4
 Min.   : 39.0   Min.   : 27.00   Min.   : 53.00   Min.   : 33.00
 1st Qu.: 60.0   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 69.00
 Median : 68.0   Median : 87.00   Median :101.00   Median : 81.00
 Mean   : 69.4   Mean   : 83.59   Mean   : 99.29   Mean   : 82.59
 3rd Qu.: 80.0   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.00
 Max.   :104.0   Max.   :137.00   Max.   :140.00   Max.   :154.00
      x.5             x.6             x.7             x.8
 Min.   : 39.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.0
 1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 69.0
 Median : 68.00   Median : 85.00   Median :101.00   Median : 81.0
 Mean   : 69.15   Mean   : 83.24   Mean   : 99.11   Mean   : 82.5
 3rd Qu.: 80.00   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.0
 Max.   :104.00   Max.   :137.00   Max.   :145.00   Max.   :157.0
      x.9             x.10            x.11            x.12
 Min.   : 40.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.00
 1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 68.00
 Median : 67.00   Median : 85.00   Median :100.00   Median : 81.00
 Mean   : 68.91   Mean   : 82.89   Mean   : 98.85   Mean   : 82.39
 3rd Qu.: 79.00   3rd Qu.:102.00   3rd Qu.:113.00   3rd Qu.: 92.00
 Max.   :104.00   Max.   :130.00   Max.   :145.00   Max.   :157.00
      x.13            x.14            x.15            x.16
 Min.   : 39.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.00
 1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 69.00
 Median : 68.00   Median : 85.00   Median :101.00   Median : 81.00
 Mean   : 69.29   Mean   : 83.48   Mean   : 99.31   Mean   : 82.64
```

```
3rd Qu.: 80.00   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.00
Max.   :104.00   Max.   :137.00   Max.   :145.00   Max.   :154.00
     x.17             x.18             x.19             x.20
Min.   : 40.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.0
1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 69.0
Median : 68.00   Median : 85.00   Median :100.00   Median : 81.0
Mean   : 69.05   Mean   : 83.17   Mean   : 99.15   Mean   : 82.6
3rd Qu.: 79.00   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.0
Max.   :104.00   Max.   :130.00   Max.   :145.00   Max.   :157.0
     x.21             x.22             x.23             x.24
Min.   : 39.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.00
1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 68.00
Median : 67.00   Median : 84.00   Median :100.00   Median : 81.00
Mean   : 68.84   Mean   : 82.86   Mean   : 98.95   Mean   : 82.47
3rd Qu.: 79.00   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.00
Max.   :104.00   Max.   :130.00   Max.   :145.00   Max.   :157.00
     x.25             x.26             x.27             x.28
Min.   : 39.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.00
1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 69.00
Median : 68.00   Median : 85.00   Median :100.00   Median : 81.00
Mean   : 69.16   Mean   : 83.37   Mean   : 99.21   Mean   : 82.66
3rd Qu.: 79.00   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.00
Max.   :104.00   Max.   :131.00   Max.   :140.00   Max.   :154.00
     x.29             x.30             x.31             x.32
Min.   : 39.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.00
1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 69.00
Median : 68.00   Median : 85.00   Median :100.00   Median : 81.00
Mean   : 68.94   Mean   : 83.15   Mean   : 99.11   Mean   : 82.62
3rd Qu.: 79.00   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.00
Max.   :104.00   Max.   :130.00   Max.   :145.00   Max.   :157.00
     x.33             x.34             x.35             x.36
Min.   : 39.00   Min.   : 27.00   Min.   : 50.00   Min.   : 29.00
1st Qu.: 60.00   1st Qu.: 71.00   1st Qu.: 85.00   1st Qu.: 68.00
Median : 67.00   Median : 84.00   Median :100.00   Median : 81.00
Mean   : 68.73   Mean   : 82.86   Mean   : 98.93   Mean   : 82.51
3rd Qu.: 79.00   3rd Qu.:103.00   3rd Qu.:113.00   3rd Qu.: 92.00
Max.   :104.00   Max.   :130.00   Max.   :145.00   Max.   :157.00
               classes
red soil           :1533
cotton crop        : 703
grey soil          :1358
damp grey soil     : 626
vegetation stubble : 707
very damp grey soil:1508
```

```
1.2 - >
> install.packages("mlbench")
> install.packages("e1071")
> install.packages("randomForest")
> install.packages("kernlab")
> install.packages("caret")
> library(mlbench)
> library(caret)
> data(Satellite)

Criar bases de Treino e Teste

> dados <- Satellite
> set.seed(7)
> indices <- createDataPartition(dados$classes, p=0.80, list=FALSE)
> treino <- dados[indices,]
> teste <- dados[-indices,]
> str(dados)
'data.frame':   6435 obs. of  37 variables:
 $ x.1    : num  92 84 84 80 84 80 76 76 76 76 ...
 $ x.2    : num  115 102 102 102 94 94 102 102 89 94 ...
 $ x.3    : num  120 106 102 102 102 98 106 106 98 98 ...
 $ x.4    : num  94 79 83 79 79 76 83 87 76 76 ...
 $ x.5    : num  84 84 80 84 80 80 76 80 76 76 ...
 $ x.6    : num  102 102 102 94 94 102 102 98 94 98 ...
 $ x.7    : num  106 102 102 102 98 102 106 106 98 102 ...
 $ x.8    : num  79 83 79 79 76 79 87 79 76 72 ...
 $ x.9    : num  84 80 84 80 80 76 80 76 76 76 ...
 $ x.10   : num  102 102 94 94 102 102 98 94 98 94 ...
 $ x.11   : num  102 102 102 98 102 102 106 102 102 90 ...
 $ x.12   : num  83 79 79 76 79 79 79 76 72 76 ...
 $ x.13   : num  101 92 84 84 84 76 80 80 80 76 ...
 $ x.14   : num  126 112 103 99 99 99 107 112 95 91 ...
 $ x.15   : num  133 118 104 104 104 104 118 118 104 104 ...
 $ x.16   : num  103 85 81 78 81 81 88 88 74 74 ...
 $ x.17   : num  92 84 84 84 76 76 80 80 76 76 ...
 $ x.18   : num  112 103 99 99 99 99 112 107 91 95 ...
 $ x.19   : num  118 104 104 104 104 108 118 113 104 100 ...
 $ x.20   : num  85 81 78 81 81 85 88 85 74 78 ...
 $ x.21   : num  84 84 84 76 76 76 80 80 76 76 ...
 $ x.22   : num  103 99 99 99 99 103 107 95 95 91 ...
 $ x.23   : num  104 104 104 104 108 118 113 100 100 100 ...
 $ x.24   : num  81 78 81 81 85 88 85 78 78 74 ...
 $ x.25   : num  102 88 84 84 84 84 79 79 75 75 ...
 $ x.26   : num  126 121 107 99 99 103 107 103 91 91 ...
 $ x.27   : num  134 128 113 104 104 104 113 104 96 96 ...
 $ x.28   : num  104 100 87 79 79 79 87 83 75 71 ...
 $ x.29   : num  88 84 84 84 84 79 79 79 75 79 ...
```

```
 $ x.30   : num  121 107 99 99 103 107 103 103 91 87 ...
 $ x.31   : num  128 113 104 104 104 109 104 104 96 93 ...
 $ x.32   : num  100 87 79 79 79 87 83 79 71 71 ...
 $ x.33   : num  84 84 84 84 79 79 79 79 79 79 ...
 $ x.34   : num  107 99 99 103 107 107 103 95 87 87 ...
 $ x.35   : num  113 104 104 104 109 109 104 100 93 93 ...
 $ x.36   : num  87 79 79 79 87 87 79 79 71 67 ...
 $ classes: Factor w/ 6 levels "red soil","cotton crop",..: 3 3 3 3 3 3 3 3 4
```

1.3 -

Treinar RF, SVM e RNA com a base de Treino

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

1.4 -

Aplicar modelos treinados na base de Teste

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

Criar as matrizes de confusão e comparar os resultados

```
> confusionMatrix(predict.rf, teste$classes)
> confusionMatrix(predict.svm, teste$classes)
> confusionMatrix(predict.rna, teste$classes)
```

#RF

```
> confusionMatrix(predict.rf, teste$classes)
Confusion Matrix and Statistics

                     Reference
Prediction            red soil cotton crop grey soil damp grey soil
red soil                   301           0          3              1
cotton crop                  0         138          0              1
grey soil                    3           0        263             26
damp grey soil               0           0          3             77
vegetation stubble           2           0          0              0
very damp grey soil          0           2          2             20

                Reference
Prediction                  vegetation stubble very damp grey soil
red soil                                      7                0
cotton crop                                   1                1
grey soil                                     0                4
damp grey soil                                0               14
vegetation stubble                          125                3
```

```
very damp grey soil                        8            279

Overall Statistics

               Accuracy : 0.9213
                 95% CI : (0.9052, 0.9355)
   No Information Rate : 0.2383
   P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9025

Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: red soil Class: cotton crop Class: grey soil
Sensitivity                   0.9837           0.9857              0.9705
Specificity                   0.9888           0.9974              0.9674
Pos Pred Value                0.9647           0.9787              0.8885
Neg Pred Value                0.9949                0.9983             0.9919
Prevalence                    0.2383                0.1090              0.2111
Detection Rate                0.2344                0.1075             0.2048
Detection Prevalence     0.2430                0.1098          0.2305
Balanced Accuracy        0.9862                0.9915              0.9690

                     Class: damp grey soil Class: vegetation stubble
Sensitivity                        0.61600              0.88652
Specificity                        0.98533                     0.99563
Pos Pred Value                     0.81915                     0.96154
Neg Pred Value                     0.95966                     0.98614
Prevalence                         0.09735                     0.10981
Detection Rate                     0.05997                     0.09735
Detection Prevalence         0.07321                     0.10125

Balanced Accuracy                  0.80067                     0.94108

                     Class: very damp grey soil
Sensitivity                            0.9269
Specificity                            0.9674
Pos Pred Value                         0.8971
Neg Pred Value                         0.9774
Prevalence                             0.2344
Detection Rate                         0.2173
Detection Prevalence                0.2422
Balanced Accuracy                      0.9472

#SVM
> confusionMatrix(predict.svm, teste$classes)
```

```
Confusion Matrix and Statistics

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

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

Overall Statistics

               Accuracy : 0.9112
                 95% CI : (0.8943, 0.9262)
    No Information Rate : 0.2383
    P-Value [Acc > NIR] : < 2.2e-16
                  Kappa : 0.8901
Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: red soil Class: cotton crop Class: grey soil
Sensitivity                   0.9902               0.9857            0.9631
Specificity                   0.9928               0.9930            0.9645
Pos Pred Value                0.9774               0.9452            0.8788
Neg Pred Value                0.9969               0.9982            0.9899
Prevalence                    0.2383               0.1090            0.2111
Detection Rate                0.2360               0.1075            0.2033
Detection Prevalence     0.2414               0.1137            0.2313

Balanced Accuracy             0.9915               0.9894            0.9638
                     Class: damp grey soil Class: vegetation stubble
Sensitivity                        0.59200                    0.89362
Specificity                        0.97584                    0.99563
Pos Pred Value                     0.72549                    0.96183
Neg Pred Value                     0.95685                    0.98699
Prevalence                         0.09735                    0.10981
Detection Rate                     0.05763                    0.09813
```

```
Detection Prevalence           0.07944                       0.10202
Balanced Accuracy                    0.78392                       0.94462
                        Class: very damp grey soil
Sensitivity                                  0.8904
Specificity                                  0.9695
Pos Pred Value                               0.8993
Neg Pred Value                               0.9665
Prevalence                                   0.2344
Detection Rate                               0.2087
Detection Prevalence                  0.2321
Balanced Accuracy                            0.9299
#RNA
> confusionMatrix(predict.rna, teste$classes)
Confusion Matrix and Statistics

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

Overall Statistics
                Accuracy : 0.5802
                  95% CI : (0.5527, 0.6074)
   No Information Rate : 0.2383
   P-Value [Acc > NIR] : < 2.2e-16
                   Kappa : 0.4692
Mcnemar's Test P-Value : NA

Statistics by Class:

                          Class: red soil Class: cotton crop Class: grey soil
Sensitivity                        0.8889         0.9714         0.14760
Specificity                        0.9939         0.8969         0.97137
Pos Pred Value                     0.9784         0.5354         0.57971
Neg Pred Value                     0.9662         0.9961         0.80988
Prevalence                         0.2383         0.1090         0.21106
```

```
Detection Rate                      0.2118         0.1059         0.03115
Detection Prevalence         0.2165         0.1978         0.05374
Balanced Accuracy                   0.9414         0.9341         .55949

                            Class: damp grey soil Class: vegetation stubble
Sensitivity                                 0.00000                     0.0070922
Specificity                                 1.00000                     1.0000000
Pos Pred Value                              NaN                         1.0000000
Neg Pred Value                              0.90265                     0.8908807
Prevalence                                  0.09735                     0.1098131
Detection Rate                              0.00000                     0.0007788
Detection Prevalence                0.00000                       0.0007788
Balanced Accuracy                           0.50000                     0.5035461
                            Class: very damp grey soil
Sensitivity                                       0.9834
Specificity                                       0.6073
Pos Pred Value                                    0.4340
Neg Pred Value                                    0.9917
Prevalence                                        0.2344
Detection Rate                                    0.2305
Detection Prevalence                0.5312
Balanced Accuracy                                 0.7954
```

1.5 -

O algoritmo Random Forest apresentou melhor resultado, utilizando-se como métrica o índice de acurácia (92,13%), contra 91,12% do SVM - Support Vector Machines e 58,02% do RNA - Neural Network.

```
#RF !!!!!!!
Overall Statistics

                     Accuracy : 0.9213
                       95% CI : (0.9052, 0.9355)
       No Information Rate : 0.2383
       P-Value [Acc > NIR] : < 2.2e-16

#SVM
Overall Statistics
                     Accuracy : 0.9112
                       95% CI : (0.8943, 0.9262)
       No Information Rate : 0.2383
       P-Value [Acc > NIR] : < 2.2e-16
                        Kappa : 0.8901

#RNA
Overall Statistics
                     Accuracy : 0.5802
```

```
                    95% CI : (0.5527, 0.6074)
       No Information Rate : 0.2383
       P-Value [Acc > NIR] : < 2.2e-16
                     Kappa : 0.4692
```

ADENDO -

Foi realizada uma nova bateria de treinamento utilizando apenas os quatro valores espectrais para o pixel central (atributos 17, 18, 19 e 20), no entanto, a acurácia dos resultados obtidos ficou aquém da observada no treinamento com a base completa.

```
> dados_2 = dados[c('x.17', 'x.18', 'x.19', 'x.20', 'classes')]
> set.seed(7)
> indices_2 <- createDataPartition(dados$classes, p=0.80, list=FALSE)
> treino_2 <- dados_2[indices_2,]
> teste_2 <- dados_2[-indices_2,]

> rf_2 <- train(classes~., data=treino_2, method="rf")
> svm_2 <- train(classes~., data=treino_2, method="svmRadial")
> rna_2 <- train(classes~., data=treino_2, method="nnet", trace=FALSE)

> predict.rf_2 <- predict(rf_2, teste_2)
> predict.svm_2 <- predict(svm_2, teste_2)
> predict.rna_2 <- predict(rna_2, teste_2)

> confusionMatrix(predict.rf_2, teste_2$classes)
> confusionMatrix(predict.svm_2, teste_2$classes)

Confusion Matrix and Statistics
Overall Statistics

#RF
                    Accuracy : 0.8419
                      95% CI : (0.8208, 0.8614)
       No Information Rate : 0.2383
       P-Value [Acc > NIR] : < 2.2e-16
                     Kappa : 0.8047
#SVM
                    Accuracy : 0.8707
                      95% CI : (0.8511, 0.8886)
       No Information Rate : 0.2383
       P-Value [Acc > NIR] : < 2.2e-16
                     Kappa : 0.8399
#RNA
                    Accuracy : 0.8084
                      95% CI : (0.7858, 0.8296)
       No Information Rate : 0.2383
       P-Value [Acc > NIR] : < 2.2e-16
                     Kappa : 0.7595
```

```
2.1 -
install.packages("e1071")
install.packages("randomForest")
install.packages("kernlab")
install.packages("caret")
install.packages("RSNNS")
library("caret")
library(RSNNS)
> install.packages("e1071")
package ‘e1071’ successfully unpacked and MD5 sums checked
...
> install.packages("randomForest")
package ‘randomForest’ successfully unpacked and MD5 sums checked
...
> install.packages("kernlab")
package ‘kernlab’ successfully unpacked and MD5 sums checked
...
> install.packages("caret")
package ‘caret’ successfully unpacked and MD5 sums checked
...
> install.packages("RSNNS")
package ‘RSNNS’ successfully unpacked and MD5 sums checked
...
> vol <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv")
> str(vol)
'data.frame': 100 obs. of 5 variables:
$ NR : int 1 2 3 4 5 6 7 8 9 10 ...
$ DAP: num 34 41.5 29.6 34.3 34.5 29.9 28.4 29.5 36.3 36.3 ...
$ HT : num 27 27.9 26.4 27.1 26.2 ...
$ HP : num 1.8 2.75 1.15 1.95 1 1.9 2.3 2.4 1.8 1.5 ...
$ VOL: num 0.897 1.62 0.801 1.079 0.98 ...
> head(vol)
NR DAP HT HP VOL
1 1 34.0 27.00 1.80 0.8971441
2 2 41.5 27.95 2.75 1.6204441
3 3 29.6 26.35 1.15 0.8008181
4 4 34.3 27.15 1.95 1.0791682
5 5 34.5 26.20 1.00 0.9801112
6 6 29.9 27.10 1.90 0.9067022

2.2 -
> vol$NR <- NULL
> head(vol)
DAP HT HP VOL
1 34.0 27.00 1.80 0.8971441
2 41.5 27.95 2.75 1.6204441
3 29.6 26.35 1.15 0.8008181
4 34.3 27.15 1.95 1.0791682
```

```
5 34.5 26.20 1.00 0.9801112
6 29.9 27.10 1.90 0.9067022
```

2.3 -

Particionar a bases em treino (80%) e teste (20%)

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

2.4 -

```
# Random Forest (rf),
> rf <- caret::train(VOL~., data=treino, method="rf")
# SVM (svmRadial),
> svm <- caret::train(VOL~., data=treino, method="svmRadial")
# Redes Neurais (neuralnet)
> rna <- caret::train(VOL~., data=treino, method="nnet")
# modelo alométrico de SPURR
§ O modelo alométrico é dado por: Volume = b0 + b1 * dap2 * Ht
> alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, data=treino, start=list(b0=0.5, b1=0.5))
```

2.5 -

```
# Random Forest (rf),
> predicoes.rf <- predict(rf, teste)
# SVM (svmRadial),
> predicoes.svm <- predict(svm, teste)
# Redes Neurais (neuralnet)
> predicoes.rna <- predict(rna, teste)
# Predições do modelo alométrico Spurr
> predicoes.spurr <- predict(alom, newdata = teste)
```

2.6 -

§ Coeficiente de determinação: R2

onde $y!$ é o valor observado, $y""$ é o valor predito e $y\#$ é a média dos valores $y!$ observados.

Quanto mais perto de 1 melhor é o modelo;

```
# Função para cálculo do R²
> calc_r2 <- function( teste, predicoes) {
+ y_real <- teste
+ y_pred <- predicoes
+
+ media_observada <- mean(y_real)
+ soma_quad_residuo <- sum((y_real - y_pred)^2)
+ soma_total_quad <- sum((y_real - media_observada)^2)
```

```
+ r2 <- 1 - (soma_quad_residuo / soma_total_quad)
+ return(r2)
+ }
> r2_rf <- calc_r2(teste$VOL, predicoes.rf)
> r2_svm <- calc_r2(teste$VOL, predicoes.svm)
> r2_rna <- calc_r2(teste$VOL, predicoes.rna)
> r2_spurr <- calc_r2(teste$VOL, predicoes.spurr)
> print(c("Random Forest:", r2_rf,
+ "SVM:", r2_svm,
+ "Rede Neural:", r2_rna,
+ "Spurr:", r2_spurr))
[1] "Random Forest:" "0.853564671341675" "SVM:"
[4] "0.848465216045881" "Rede Neural:" "-0.724494571013155"
[7] "Spurr:" "0.82631339006152"
!!![1] - "Random Forest:" "0.85"
§ Erro padrão da estimativa: Syx
esta métrica indica erro, portanto quanto mais perto de 0 melhor é o modelo;

> calc_syx <- function(y_real, y_pred) {
+ n <- length(y_real)
+ soma_quad_residuo <- sum((y_real - y_pred)^2)
+ syx <- sqrt(soma_quad_residuo / (n - 2))
+ return(syx)
+ }
> syx_rf <- calc_syx(teste$VOL, predicoes.rf)
> syx_svm <- calc_syx(teste$VOL, predicoes.svm)
> syx_rna <- calc_syx(teste$VOL, predicoes.rna)
> syx_spurr <- calc_syx(teste$VOL, predicoes.spurr)
> print(c("Random Forest:", syx_rf,
+ "SVM:", syx_svm,
+ "Rede Neural:", syx_rna,
+
+ "Spurr:", syx_spurr))
[1] "Random Forest:" "0.144552702228724" "SVM:"
[4] "0.147048110566199" "Rede Neural:" "0.496059986670714"
[7] "Spurr:" "0.157429621324422"
!!!!! [1] "Random Forest:" "0.144"
§ Syx% - Erro padrão percentual da estimativa
esta métrica indica porcentagem de erro, portanto quanto mais perto de 0 melhor é
o modelo;
# Função para cálculo do Syx em porcentagem (Syx%)
> calc_syx_perc <- function(y_real, y_pred) {
+ syx <- calc_syx(y_real, y_pred)
+ media_observada <- mean(y_real)
+ syx_perc <- (syx / media_observada) * 100
+ return(syx_perc)
+ }
> syx_perc_rf <- calc_syx_perc(teste$VOL, predicoes.rf)
```

```
> syx_perc_svm <- calc_syx_perc(teste$VOL, predicoes.svm)
> syx_perc_rna <- calc_syx_perc(teste$VOL, predicoes.rna)
> syx_perc_spurr <- calc_syx_perc(teste$VOL, predicoes.spurr)
> print(c("Random Forest:", syx_perc_rf,
+ "SVM:", syx_perc_svm,
+ "Rede Neural:", syx_perc_rna,
+
+ "Spurr:", syx_perc_spurr))
[1] "Random Forest:" "11.0765834729613" "SVM:"
[4] "11.2677981533029" "Rede Neural:" "38.0113949115951"
[7] "Spurr:" "12.0632981247039"
!!!! [1] "Random Forest:" "11.07"
```

2.7 -

O algoritmo Random Forest apresentou melhor resultado, utilizando-se como métricas o coeficiente de determinação: R2 ("0.85"), erro padrão da estimativa: Syx ("0.144") e o erro padrão da estimativa percentual: Syx% ("11.07%"), contra, respectivamente,
("0,84" / "0,147" / "11,26%") do SVM - Support Vector Machines,
("-0,72" / "0,49" / "38,01%") do RNA - Neural Network e
("0,82" / "0,157" / "12,06%") do modelo alométrico de SPURR.
Nota - valor negativo para o coeficiente de determinação R^2 do algoritmo RNA.
(Warning message:
In nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
There were missing values in resampled performance measures.

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

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

B – RESOLUÇÃO

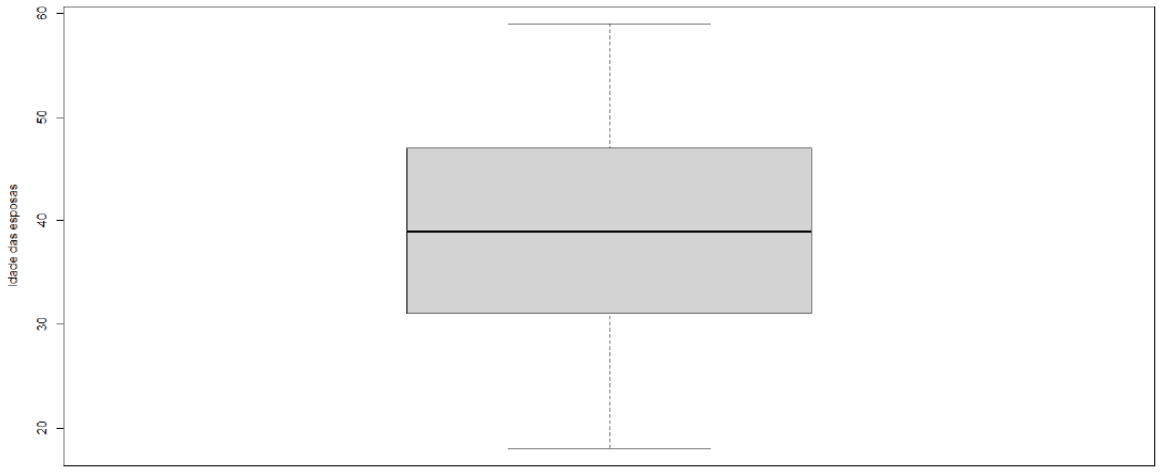
Apresentar a resolução (somente o resultado) das questões do trabalho.

1 -

a)

```
load("salarios.RData")
library (car)
```

```
> Boxplot( ~ age, data=salarios, id=list(method="y"),
+ ylab="Idade das esposas")
```

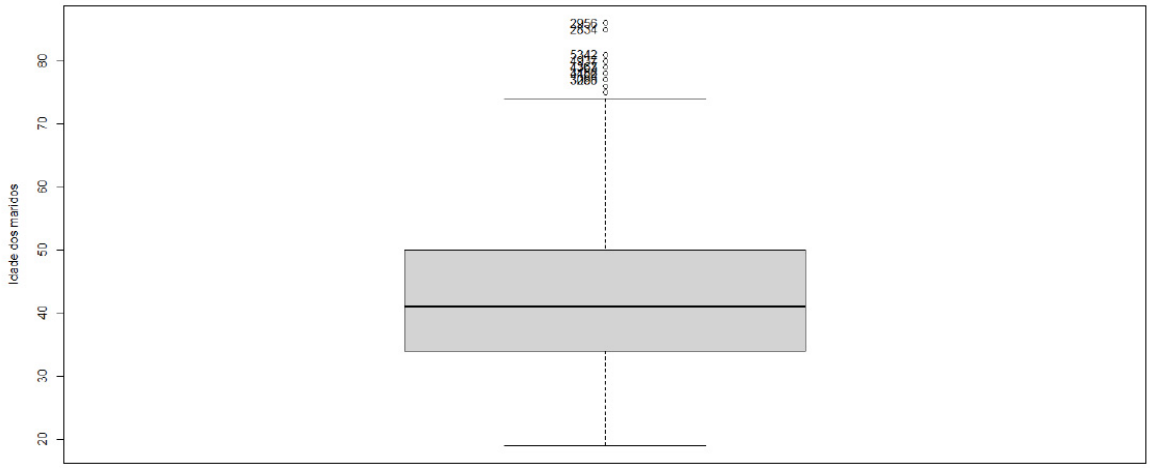


Idades das Esposas - Observações

```
> summary(salarios$age)
Min. 1st Qu. Median Mean 3rd Qu. Max.
18.00 31.00 39.00 39.43 47.00 59.00
> quantile(salarios$age)
0% 25% 50% 75% 100%
18 31 39 47 59
```

```
# Valor mínimo igual a 18.
# Média igual a 39.43.
# Mediana igual a 39.
# Valor máximo igual a 59.
# Gráfico não apresenta outliers.
```

```
> Boxplot( ~ husage, data=salarios, id=list(method="y"),
+ ylab="Idade dos maridos")
[1] "2956" "2834" "5342" "4937" "4363" "4367" "3456" "4182" "3090" "3288"
```



Idades dos Maridos - Observações:

```
> summary(salarios$husage)
Min. 1st Qu. Median Mean 3rd Qu. Max.
19.00 34.00 41.00 42.45 50.00 86.00
> quantile(salarios$husage)
0% 25% 50% 75% 100%
19 34 41 50 86
```

Valor mínimo igual a 19.

Média igual a 42.45.

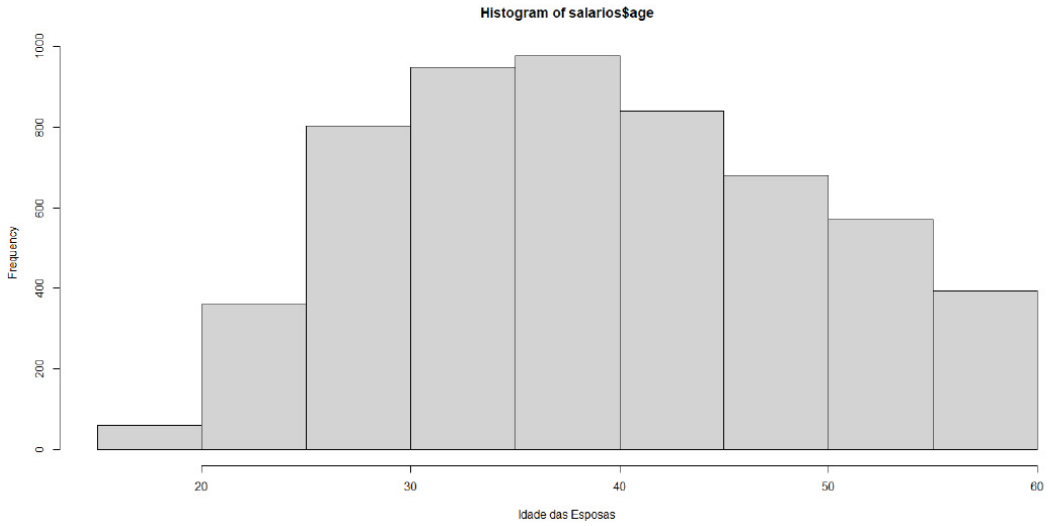
Mediana igual a 41.

Valor máximo igual a 86.

Gráfico apresenta alguns outliers.

Valores apresentam maior amplitude (67) que a das esposas.

```
> hist(salarios$age, xlab="Idade das Esposas",
+ ylab = "Frequency")
```



Idades das Esposas - Observações:

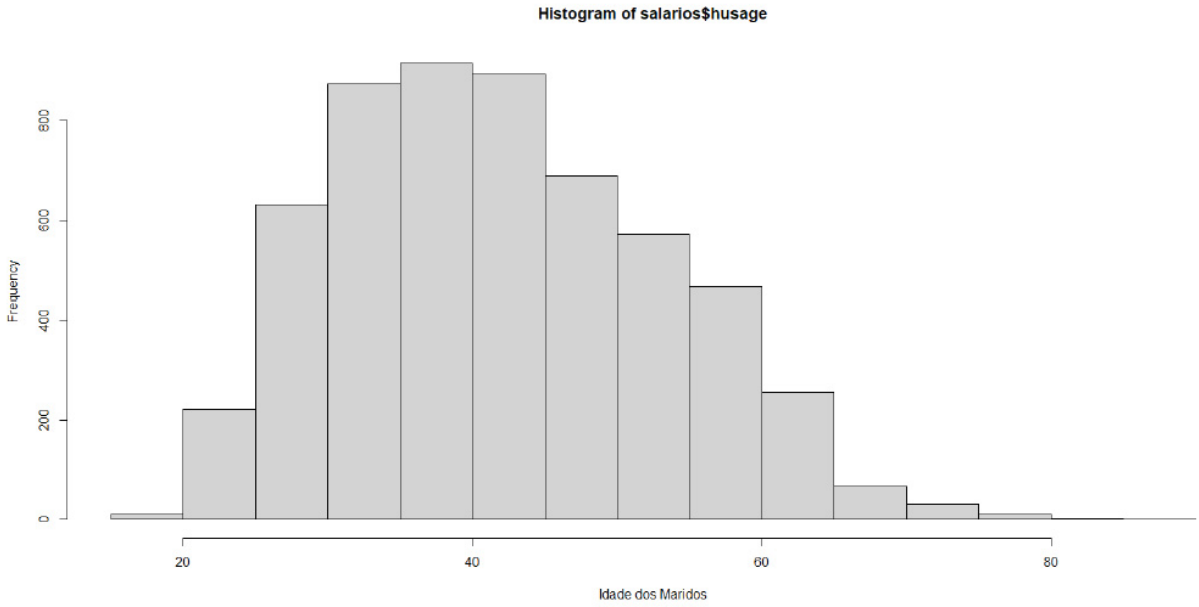
Histograma unimodal

Leve assimetria para a direita

#Maior predominância na faixa dos 35-40 anos

Ligeiro achatamento das barras indicando leve dispersão dos dados

```
> hist(salarios$husage, xlab="Idade dos Maridos",
+ ylab = "Frequency")
```



Idades dos Maridos - Observações:

Histograma unimodal

Sutil assimetria para a direita

#Maior predominância na faixa dos 35-40 anos

Concentração dos dados próximos do valor central (pouco achatamento das barras)
Ligeira assimetria

b)

```
library(fdth)
> table <- fdt(salarios$husage)
```

```
> print (table_m)
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
```

```
> table_e <- fdt(salarios$age)
```

```
> print (table_e)
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
```

Distribuição: Ambas as tabelas apresentam uma distribuição similar, ligeiramente assimétrica positiva, com a maior concentração de frequências nas classes centrais e declínio gradual nas extremidades.

Amplitude: O intervalo de idades coberto pelas tabelas é parecido, indo de 19 anos até 86 anos para os maridos e 18 anos até 59 anos para as esposas.

Frequências: Apesar da similaridade no formato geral da distribuição, há algumas diferenças pontuais nas frequências entre as classes equivalentes. Por exemplo, na faixa etária de 23.671 a 28.531 anos, há uma proporção maior de maridos (8.27%) do que esposas (5.54%).

2 -

a)

```
> mean(salarios$husage)
[1] 42.45296
> mean(salarios$age)
[1] 39.42758
> 42.45296 / 39.42758
[1] 1.076733
```

```
# A média da idade dos maridos na amostra é 42.45 anos
# A média da idade das esposas na amostra é 39.42 anos.
# se dividirmos a idade média dos maridos pela das esposas temos 1.07
# Portanto, a idade média dos maridos é ligeiramente maior que a das esposas.
```

```
> median(salarios$husage)
[1] 41
> median(salarios$age)
[1] 39
> ((41/39)-1)*100
[1] 5.128205
```

```
# A mediana da idade dos maridos na amostra é 41 anos
# A mediana da idade das esposas na amostra é 39 anos.
# se dividirmos mediana da idade dos maridos pela das esposas temos
5.12
# Em termos de mediana a idade dos maridos 5,12 % maior que a das esposas.
```

```
> # Para a idade dos maridos
```

```
> table(salarios$husage)
19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
5 6 25 31 35 71 59 115 113 108 137 157 172 174 169 196 161 174 195 169 200 175
41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
192 172 178 201 149 149 130 147 135 127 112 128 102 104 126 103 106 86 91 81 74 62
63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 85 86
45 45 30 18 14 14 13 8 7 9 6 2 7 4 2 2 2 1 1 1 1
> # Para a idade das esposas
> table(salarios$age)
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
12 18 31 47 47 67 84 114 114 174 161 170 184 191 187 180 204 187 205 217 202 171
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59
181 158 172 185 185 140 164 128 131 125 133 117 138 103 109 104 91 104 104 95
```

```
# A moda da idade dos maridos é de 44 anos, com 201 pessoas
# A moda da idade das esposas é de 37 anos, com 217 pessoas
# Portanto, a moda da idade dos maridos é maior que a das
# esposas:
44/37
# aproximadamente 18,92% maior.
```

b)

```
-> var(salarios$husage)
[1] 126.0717
> var(salarios$age)
[1] 99.75234
> ((126.0717/ 99.75234)-1)*100
[1] 26.3847
```

```
# A variância da idade dos maridos é 126.07
# A variância da idade das esposas é 99.75
# Se dividirmos a variância das idades dos maridos pela das esposas:
((126.0717 / 99.75234) -1 ) * 100
26.3847
# Portanto, a variância das idades dos maridos é 26.38% maior que a das
esposas.
```

```
> sd(salarios$husage)
[1] 11.22817
> sd(salarios$age)
[1] 9.98761
> ((11.22817/9.98761)-
```

```
# O desvio padrão da idade dos maridos é 11.22817
```

```
# O desvio padrão da idade das esposas é 9.98761
# Se dividirmos o desvio padrão da idade dos maridos pelo das esposas:
((11.22817/9.98761)-1)*100
[1] 12.42099
# O desvio padrão das idades dos maridos é 12.42% maior que a das esposas.
```

```
> mean(salarios$husage)
[1] 42.45296
```

```
# A média das idades dos maridos é 42.45296
```

```
> mean(salarios$age)
[1] 39.42758
```

```
# A média das idades das esposas é 39.42758
```

```
(sd(salarios$husage)/mean(salarios$husage))*100
[1] 26.44849
```

```
# O coeficiente de variação das idades dos maridos é 26.44849%
```

```
> (sd(salarios$age)/mean(salarios$age))*100
[1] 25.33153
```

```
# O coeficiente de variação das idades das esposas 25.33153%

# Isso quer dizer que as idades dos maridos e esposas variam de forma
equilibrada na amostra. Ainda, pode-se concluir que as idades das esposas
variam menos que as dos maridos.

# Regra de bolso:
# Quando o CV for:
# a) CV < 15% existe baixa dispersão: dados homogêneos
# b) 15% =< CV <= 30% existe media dispersão
# c) CV > 30% existe alta dispersão: dados heterogêneos


3 -
a)
# Carregar os pacotes necessários para os testes
```

```
library(carData)
library(datasets)
library(BSDA)
library(nortest)
library(stats)
library(rcompanion)
library(dplyr)
library(tigerstats)
library(misty)
```

Carregar a base de dados

```
> load("salarios.RData")
```

Calcular a média e o desvio padrão da variável "age" de toda amostra

```
> mean(salarios$age)
[1] 39.42758
```

Calcular a média e o desvio padrão da variável "husage" de toda amostra

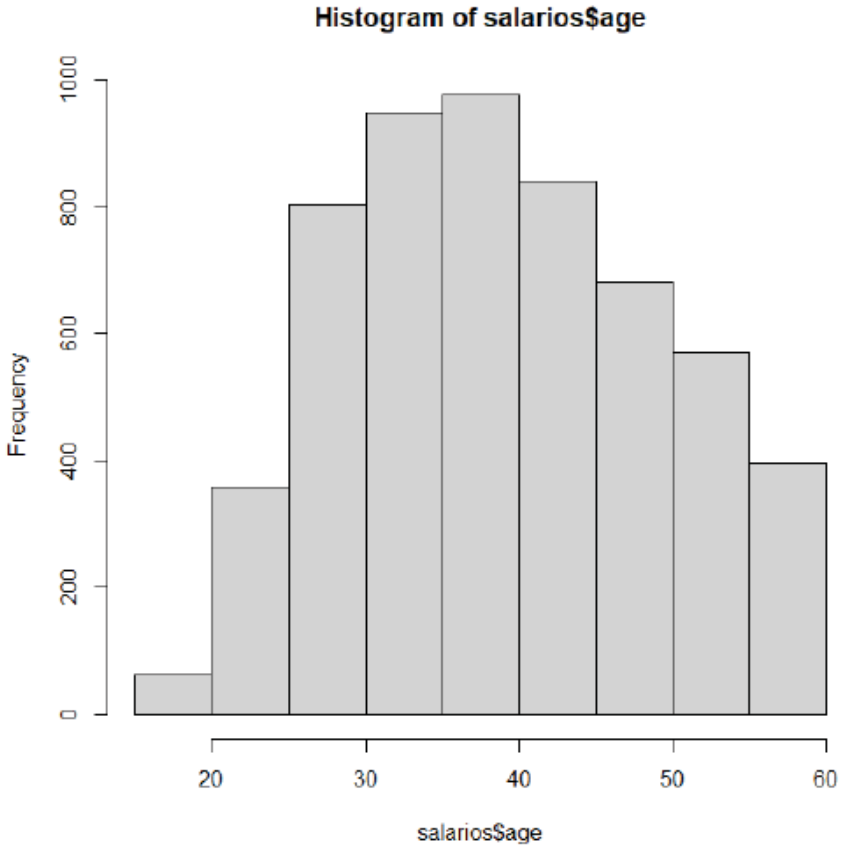
```
> mean(salarios$husage)
[1] 42.45296
```

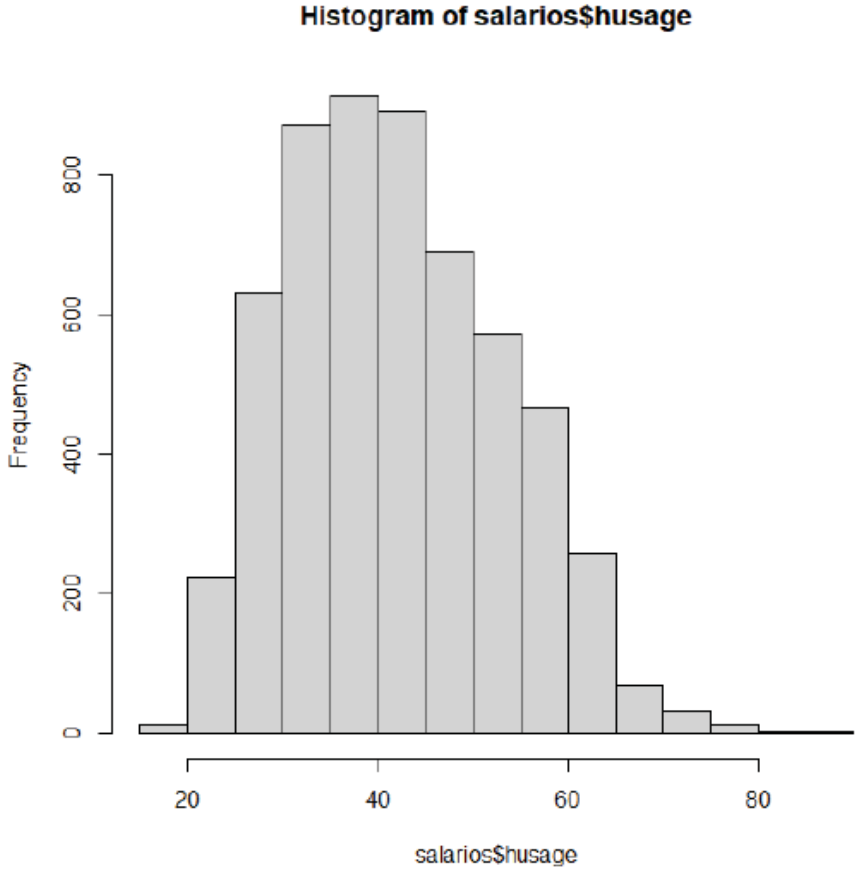
Calcular o desvio padrão de toda amostra:

```
> sd(salarios$age)
[1] 9.98761
> sd(salarios$husage)
[1] 11.22817
```

histograma:

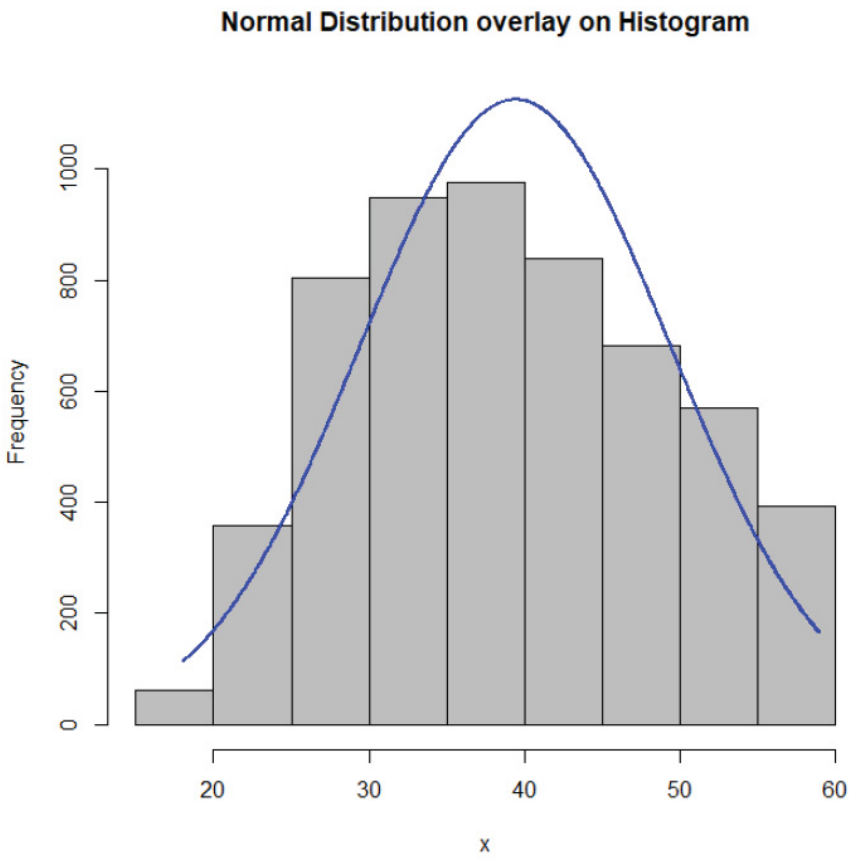
```
> hist(salarios$age)
> hist(salarios$husage)
```

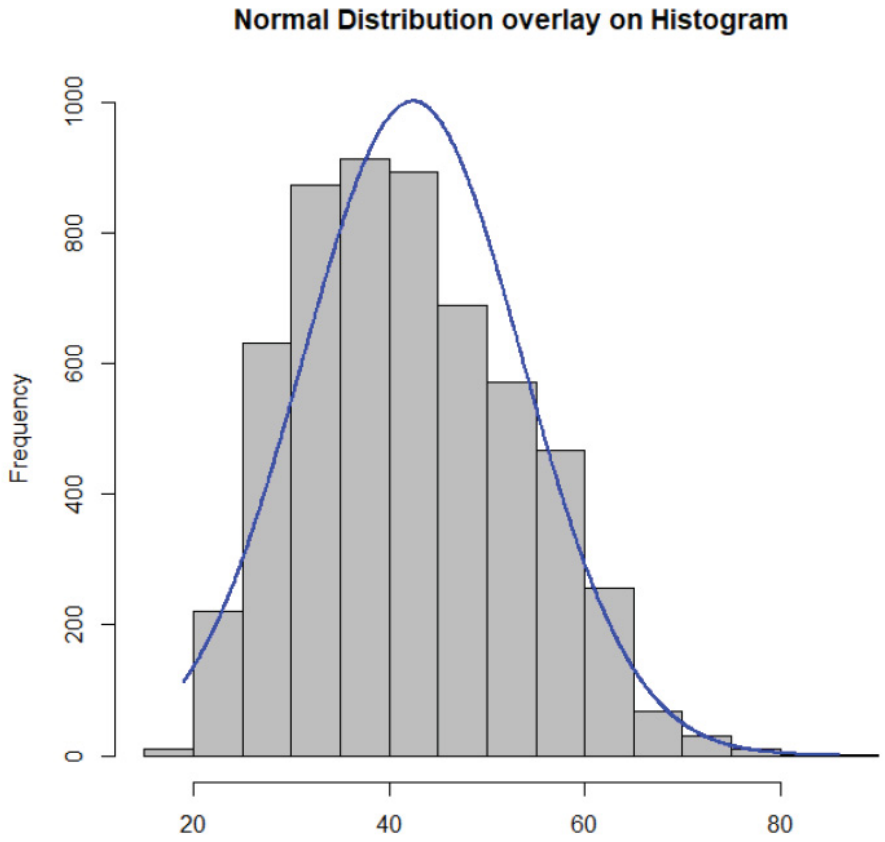

Histogram of salarios$age
Frequency
0
200
400
600
800
1000
20
30
40
50
60
salarios$age


Histogram of salarios$husage
Frequency
0
200
400
600
800
20
40
60
80
salarios$husage

Plotar a curva normal sobre o histograma:

```
> plotNormalHistogram(salarios$age, prob = FALSE,
main = "Normal Distribution overlay on Histogram",
length = 1000 )

> plotNormalHistogram(salarios$husage, prob = FALSE,
main = "Normal Distribution overlay on Histogram",
length = 1000 )
```





Executar teste de normalidade de Kolmogorov-Smirnov

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

D = 0.059662, p-value < 0.00000000000000022
```

Regra de bolso: **p-value > 0.05**, variável = normalmente distribuída
se não for identificada normalidade
deve-se utilizar um teste equivalente **não paramétrico**,
portanto, não utilizar o teste "Z" nem "t"

Devido ao p-value sensivelmente menor que o valor de referência 0,05, apesar da aparente "normalidade", optouse por realizar o teste não paramétrico "U de Mann-Whitney" em vez do "Z" normal.

Instalando pacotes

```
install.packages("dplyr")
install.packages("ggpubr")
```

Carregando os pacotes

```
library("dplyr")
library("ggpubr")

options(scipen = 999)
```

Preparar os dados em dois vetores numéricos idades de maridos e esposas

```
> idade_marido <- as.vector(salarios$husage)
> idade_esposa <- as.vector(salarios$age)
```

Criar um data frame com o nome "idade"

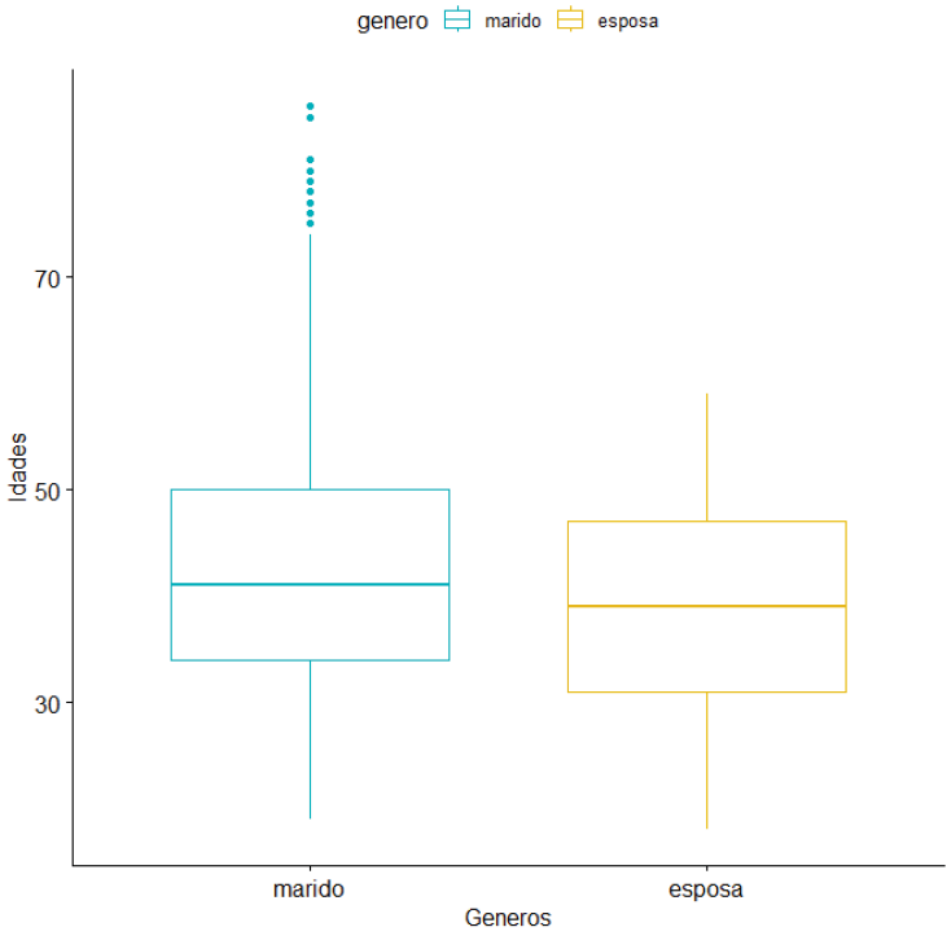
```
> idade <- data.frame(
+ genero = rep(c("marido", "esposa"), each = 5634),
+ idade = c(idade_marido, idade_esposa)
+ )
```

calcular um sumario estatístico

```
> group_by(idade, genero) %>%
+ summarise(
+ count = n(),
+ median = median(idade, na.rm = TRUE),
+ IQR = IQR(idade, na.rm = TRUE)
+ )

# A tibble: 2 x 4
genero count median IQR
<chr> <int> <dbl> <dbl>
1 esposa 5634 39 16
2 marido 5634 41 16
```

```
# visualizar os dados usando box-plots
# Plotar as idades por gênero
```

```
> ggboxplot(idade, x = "genero", y = "idade",
+ color = "genero", palette=c("#00AFBB", "#E7B800"),
+ ylab = "Idades", xlab = "Generos")
```



 # Testar* se a idade mediana das esposas é estatisticamente igual à dos maridos

 # Hipóteses do teste:

H0: A idade mediana das esposas é estatisticamente **igual** à dos maridos;

Ha: A idade mediana das esposas **NÃO** é estatisticamente **igual** à dos maridos;

* A título de verificação, os testes foram executados tanto no data frame "idade", criado a partir dos dados extraídos de "salários.RData", quanto na própria base "salários.RData". Os resultados foram os mesmos.

```
> wilcox.test(salarios$age, salarios$husage,
+ exact = FALSE, conf.int=TRUE)
Wilcoxon rank sum test with continuity correction
data: salarios$age and salarios$husage
W = 13619912, p-value < 0.00000000000000022
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
-3.000024 -2.000033
sample estimates:
difference in location
-2.999966
```

```
> options(scipen = 999)
> wilcox.test(idade ~ genero, data = idade,
+ exact = FALSE, conf.int=TRUE)
Wilcoxon rank sum test with continuity correction
data: idade by genero
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
```

p-value < 0.00000000000000022

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 **NÃO** é estatisticamente igual à dos maridos;
(rejeitamos H0).
O intervalo de confiança da diferença entre as medianas esta
entre -3.000024 -2.000033, com uma mediana de -2.999966.

 # Testar se a idade mediana das esposas é estatisticamente menor que a dos maridos
 # Hipoteses do teste:

H0: A idade mediana das esposas **NÃO** é estatisticamente **menor** que dos maridos;
Ha: A idade mediana das esposas **é** estatisticamente **menor** que a dos maridos;

```
> wilcox.test(salarios$age, salarios$husage,
+ exact = FALSE, alternative = "less",
+ conf.int=TRUE)
Wilcoxon rank sum test with continuity correction
data: salarios$age and salarios$husage
W = 13619912, p-value < 0.00000000000000022
alternative hypothesis: true location shift is less than 0
95 percent confidence interval:
-Inf -2.000046
sample estimates:
difference in location
-2.999966
```

```
> wilcox.test(idade ~ genero, data = idade,
+ exact = FALSE, alternative = "less",
+ conf.int=TRUE)
Wilcoxon rank sum test with continuity correction
data: idade by genero
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
```

p-value < 0.00000000000000022

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 **menor** que a dos maridos;
(rejeitamos H0).
O intervalo de confiança da diferença é um valor menor que -2.000046, com uma mediana de -2.999966.
 # Testar se a idade mediana das esposas é estatisticamente maior que a dos maridos
 # Hipoteses do teste:

H0: A idade mediana das esposas **NÃO** é estatisticamente **maior** que dos maridos;
Ha: A idade mediana das esposas **é** estatisticamente **maior** que a dos maridos;

```
> wilcox.test(salarios$age, salarios$husage,
+ exact = FALSE, alternative = "greater",
+ conf.int=TRUE)
Wilcoxon rank sum test with continuity correction
data: salarios$age and salarios$husage
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
```

```
> wilcox.test(idade ~ genero, data = idade,
+ exact = FALSE, alternative = "greater",
+ conf.int=TRUE)
Wilcoxon rank sum test with continuity correction
data: idade by genero
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
```

p-value = 1

O **p-value** do teste é = 1, que **é maior** que o nível de significância 0,05. Podemos concluir que a idade mediana das esposas **NÃO é** estatisticamente **maior** que a dos maridos;

 # (aceitamos H0).

O intervalo de confiança da diferença é um valor maior que 3.000034, com uma mediana de -2.999966.

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

1) Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

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

Procedimentos comuns aos 3 modelos –

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

# Carregar o dataset
load("~/Pós Graduação/UFPR/Inteligência Artificial Aplicada/IAA005 - Estatística
II/Trabalho/Dados/tr
abalhosalarios.RData")

# Salvar o dataset em um objeto chamado "dat"
> dat <- trabalhosalarios

# Visualizar o dataset
> View(dat)

# Visualizar parte do dataset
> glimpse(dat)
Rows: 2,574
Columns: 17
$ husage <dbl> 56, 31, 33, 34, 42, 45, 33, 31, 31, 44, 45, 22, 66, 43, 26~
$ husunion <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0~
...

# Visualizar a memoria
> gc()
         used (Mb) gc   trigger  (Mb)   max used (Mb)
Ncells 13521749 722.2 20822830 1112.1 20487266 1094.2
Vcells 50590652 386.0 92911914 708.9 143589241 1095.5

# Definir a semente para gerar números aleatórios
```

```
> set.seed(302)
```

Criar um índice para particionar o dataset em 80% para treinamento

```
> index = sample(1:nrow(dat),0.8*nrow(dat))
```

Criar a base de dados de treinamento

```
> train = dat[index,]
```

Criar a base de dados de teste

```
> test = dat[-index,]
```

Visualizar sumário estatístico das variáveis padronizadas de cada dataset

```
> summary(train)
husage husunion husearns huseduc
Min. :-2.1039 Min. :0.0000 Min. :-1.7145 Min. :-5.0191
1st Qu.:-0.8087 1st Qu.:0.0000 1st Qu.:-0.6615 1st Qu.:-0.5568
Median :-0.1113 Median :0.0000 Median :-0.2166 Median :-0.1849
Mean : 0.0000 Mean :0.2205 Mean : 0.0000 Mean : 0.0000
3rd Qu.: 0.6857 3rd Qu.:0.0000 3rd Qu.: 0.4212 3rd Qu.: 0.9306
Max. : 2.8775 Max. :1.0000 Max. : 3.9006 Max. : 1.6744
...
```

```
...
> summary(test)
husage husunion husearns huseduc
Min. :-1.80501 Min. :0.0000 Min. :-1.72343 Min. :-5.0191
1st Qu.:-0.70910 1st Qu.:0.0000 1st Qu.:-0.72231 1st Qu.:-0.5568
Median :-0.01171 Median :0.0000 Median :-0.20173 Median :-0.1849
Mean : 0.05078 Mean :0.2272 Mean :-0.03584 Mean :-0.0167
3rd Qu.: 0.68568 3rd Qu.:0.0000 3rd Qu.: 0.42118 3rd Qu.: 0.9306
Max. : 2.47898 Max. :1.0000 Max. : 3.90062 Max. : 1.6744
...
```

Verificar as dimensões das bases de treinamento e teste

```
> dim(train)
[1] 2059 17
> dim(test)
[1] 515 17
```

```
# A base de treinamento possui 2.059 linhas (famílias) e 17 colunas (variáveis)
# A base de teste possui 515 linhas (famílias) e 17 colunas (variáveis)

# Padronizar as variáveis
# Criar um objeto com as variáveis para padronizar
```

```
 cols = c('husage', 'husearns', 'huseduc', 'hushrs', 'earns',
+ 'age', 'educ', 'exper', 'lwage')
```

```
# Padronizando a base de treinamento e teste
```

```
> pre_proc_val <- preProcess(train[,cols],
+ method = c("center", "scale"))
> train[,cols] = predict(pre_proc_val, train[,cols])
> test[,cols] = predict(pre_proc_val, test[,cols])
```

```
# Criar um objeto com as variáveis que serão utilizadas no modelo
```

```
> cols_reg = c('husage', 'husunion', 'husearns', 'huseduc', 'hushrs',
+ 'age', 'educ', 'lwage', 'husblck',
+ 'hushisp', 'kidge6', 'black', 'hispanic',
+ 'union', 'kidlt6', 'exper')
```

```
# Criar variáveis dummies para organizar os datasets em objetos tipo matriz
# O objetivo é estimar o salário-hora (lwage)
```

```
> dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+hushrs+
+ age+educ+husblck+hushisp+
+ kidge6+black+hispanic+union+kidlt6+exper,
+ 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
```

```
# Salvar a matriz de dados de treinamento das variáveis explicativas para o
modelo em um objeto chamado "x"
```

```
> x = as.matrix(train_dummies)
```

Salvar o vetor de dados de treinamento da variável dependente para o modelo em um objeto chamado "y_train"

```
> y_train = train$lwage
```

Salvar a matriz de dados de teste das variáveis explicativas para o modelo em um objeto chamado "x_test"

```
> x_test = as.matrix(test_dummies)
```

Salvar o vetor de dados de teste da variável dependente para o modelo em um objeto chamado "y_test"

```
> y_test = test$lwage
```

```
############################################################
#                        REGRESSAO RIDGE                   #
############################################################
```

Calcular o valor ótimo de lambda;
alpha = "0", é para regressão Ridge
Testar os lambdas de 10^-3 até 10^2, a cada 0.1

```
> lambdas <- 10^seq(2, -3, by = -.1)
```

Calcular o lambda:

```
> ridge_lamb <- cv.glmnet(x, y_train, alpha = 0,
+ lambda = lambdas)
```

Verificar qual o lambda ótimo

```
> best_lambda_ridge <- ridge_lamb$lambda.min
> best_lambda_ridge
[1] 0.03162278
```

Para contar o tempo de processamento do modelo, utiliza-se:

```
> start <- Sys.time()
```

Estimando o modelo Ridge

```
> ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
+ family = 'gaussian',
+ lambda = best_lambda_ridge)
> end <- Sys.time()
> difftime(end, start, units="secs")
Time difference of 2.766889 secs
```

Visualizar o resultado (valores) da estimativa (coeficientes)

```
> ridge_reg[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"
s0
husage -0.004620700
husunion 0.001307959
husearns 0.230243852
huseduc 0.051375052
hushrs -0.069513832
age 0.069225415
educ 0.327075567
husblck 0.217274522
hushisp 0.086155394
kidge6 -0.181935190
black -0.277311982
hispanic -0.011408328
union 0.365669856
kidlt6 -0.042470270
exper -0.016346382
```

Calcular o R^2 dos valores verdadeiros e preditos conforme a seguinte função:

```
> 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
+ )
```

```
+ }
```

Predição e avaliação nos dados de treinamento:

```
> predictions_train <- predict(ridge_reg,
+ s = best_lambda_ridge,
+ newx = x)
```

As métricas da base de treinamento são:

```
> eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.8411446 0.2921319
```

Predição e avaliação nos dados de teste:

```
> predictions_test <- predict(ridge_reg,
+ s = best_lambda_ridge,
+ newx = x_test)
```

As métricas da base de teste são:

```
> eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.9893323 0.2590848
```

Comparando-se as métricas de treinamento e teste, podem-se observar erros muito altos, tanto na base de treinamento quanto na de teste.
O poder explicativo do modelo é baixo - R^2 = 29% / 25% (treinamento e teste) .
Especula-se que o alto RMSE e o baixo valor do coeficiente de determinação se devam à ausência da variável earns(rendimento da esposa).

 # Predição para o exemplo

Efetuar predição para:

husage = 40 anos (idade do marido)

```
> husage = (40-pre_proc_val[["mean"]][["husage"]])/
```

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

# husearns = 600 (US$ renda do marido por semana)
> husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
+ pre_proc_val[["std"]][["husearns"]]

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

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

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

# hushrs = 40 (horas semanais de trabalho do marido)
> hushrs = (40-pre_proc_val[["mean"]][["hushrs"]])/
+ pre_proc_val[["std"]][["hushrs"]]

# kidge6 = 1 (possui filhos maiores de 6 anos)
> kidge6 = 1

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

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

```
# educ = 13 (anos de estudo da esposa)

> educ = (13-pre_proc_val[["mean"]][["educ"]])/
+ pre_proc_val[["std"]][["educ"]]

# hispanic = 1 (a esposa é hispânica)

> hispanic = 1

# union = 0 (esposa não possui união estável)

> union = 0

# (possui filhos menores de 6 anos)

> kidlt6 = 1

#exper = 18 (anos de experiência de trabalho da esposa)

> exper = 18

# Construir uma matriz de dados para a predição

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

# Fazendo a predição:

> predict_our_ridge <- predict(ridge_reg,
+ s = best_lambda_ridge,
```

```
+ newx = our_pred)
```

O resultado da predição é:

```
> predict_our_ridge
s1
[1,] -0.2012915
```

Converter o resultado (valor padronizado) para o valor nominal consistente com o dataset original

```
> wage_pred_ridge=(predict_our_ridge*
+ pre_proc_val[["std"]][["lwage"]])+
+ pre_proc_val[["mean"]][["lwage"]]
```

O resultado é:

```
> wage_pred_ridge
s1
[1,] 2.093978
```

#necessário aplicar o antilog

```
> exp(wage_pred_ridge)
s1
[1,] 8.117144
```

Este é o valor predito do salário por hora (US$) segundo as características atribuídas.

O intervalo de confiança para o nosso exemplo é:

```
> n <- nrow(train) # tamanho da amostra
> m <- exp(wage_pred_ridge) # valor medio predito corrigido pelo antilog
> s <- pre_proc_val[["std"]][["lwage"]] # desvio padrao
> dam <- s/sqrt(n) # distribuicao da amostragem da media
> CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
> CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior
```

Os valores são:

```
> CIlwr_ridge
s1
[1,] 8.095286
```

```
> CIupr_ridge
s1
```

Então, segundo as características atribuídas, o salário-hora da esposa é
em média **US$8.12** e pode
variar entre **US$8.09** e **US$8.13**

```
#################################################################
#                          REGRESSAO LASSO                      #
#################################################################
```

Criar o intervalo para o tunning do melhor lambda

```
> lambdas <- 10^seq(2, -3, by = -.1)
```

Atribuir alpha = 1 para implementar a regressão lasso

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

Salvar o lambda "ótimo" em um objeto chamado best_lambda_lasso

```
> best_lambda_lasso <- lasso_lamb$lambda.min
> best_lambda_lasso
[1] 0.01
```

Estimar o modelo Lasso

```
> lasso_model <- glmnet(x, y_train, alpha = 1,
+ lambda = best_lambda_lasso,
+ standardize = TRUE)
```

Visualizar os coeficientes estimados

```
> lasso_model[["beta"]]
15 x 1 sparse Matrix of class "dgCMatrix"
s0
husage .
husunion .
```

```
husearns 0.231584496
huseduc 0.032323583
hushrs -0.062852083
age 0.047762218
educ 0.340458058
husblck .
hushisp 0.015873084
kidge6 -0.151422859
black -0.040962474
hispanic .
union 0.349680278
kidlt6 -0.007917178
exper .
```

Alguns coeficientes estão zerados pois não são significativos

Efetuar as predições na base de treinamento e avaliar a regressão Lasso

```
> predictions_train <- predict(lasso_model,
+ s = best_lambda_lasso,
+ newx = x)
```

Calcular o R^2 dos valores verdadeiros e preditos conforme a seguinte função:

```
> eval_results <- function(true, predicted, df) {
+ SSE <- sum((predicted - true)^2)
+ SST <- sum((true - mean(true))^2)
+ R_square <- 1 - SSE / SST
+ RMSE = sqrt(SSE/nrow(df))
+
```

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

As métricas da base de treinamento são:

```
> eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.84203 0.290641
```

Executar predições na base de teste

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

As métricas da base de teste são:

```
> eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.9894882 0.2588513
```

Comparando-se as métricas de treinamento e teste, podem-se observar erros muito altos, tanto na base de treinamento quanto na de teste.
O poder explicativo do modelo é baixo - R^2 = 29% / 25% (treinamento e teste).
Especula-se que o alto RMSE e o baixo valor do coeficiente de determinação se devam à ausência da variável **earns** (rendimento da esposa).

 # Predição para o exemplo

Efetuar predição para:

husage = 40 anos (idade do marido)

```
> husage = (40-pre_proc_val[["mean"]][["husage"]])/
```

husunion = 0 (marido não possui união estável)

```
> husunion = 0
```

husearns = 600 (US$ renda do marido por semana)

```
> husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
+ pre_proc_val[["std"]][["husearns"]]
```

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

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

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

# hushrs = 40 (horas semanais de trabalho do marido)
> hushrs = (40-pre_proc_val[["mean"]][["hushrs"]])/
+ pre_proc_val[["std"]][["hushrs"]]

# kidge6 = 1 (possui filhos maiores de 6 anos)
> kidge6 = 1

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

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

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

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

```
# union = 0 (esposa não possui união estável)
> union = 0

# (possui filhos menores de 6 anos)
> kidlt6 = 1

#exper = 18 (anos de experiência de trabalho da esposa)
> exper = 18

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

# Fazendo a predição:
> predict_our_lasso <- predict(lasso_model,
+ s = best_lambda_lasso,
+ newx = our_pred)

# O resultado da predição é:
> predict_our_lasso
s1
[1,] -0.1361857
```

Converter o resultado (valor padronizado) para o valor nominal consistente com o dataset original

```
> wage_pred_lasso=(predict_our_lasso*
+ pre_proc_val[["std"]][["lwage"]])+
+ pre_proc_val[["mean"]][["lwage"]]
```

O resultado é:

```
> wage_pred_lasso
s1
[1,] 2.126926
```

#necessário aplicar o antilog

```
> exp(wage_pred_lasso)
s1
[1,] 8.389036
```

Este é o valor predito do salário por hora (US$) segundo as características atribuídas.

O intervalo de confiança para o nosso exemplo é:

```
> n <- nrow(train) # tamanho da amostra
> m <- exp(wage_pred_lasso) # valor medio predito corrigido pelo antilog
> s <- pre_proc_val[["std"]][["lwage"]] # desvio padrao
> dam <- s/sqrt(n) # distribuicao da amostragem da media
> CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
> CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior
```

Os valores são:

```
> CIlwr_lasso
s1
[1,] 8.367177
> CIupr_lasso
s1
[1,] 8.410894
```

Então, segundo as características atribuídas, o salário-hora da esposa é

```
# em média US$8.38 e pode
# variar entre US$8.36 e US$8.41
```

```
################################################################
#                   REGRESSAO ELASTICNET                       #
################################################################
```

```
# Configurar o treinamento do modelo por cross validation, com 10 folders, 5
repetições e busca aleatória dos componentes das amostras de treinamento.
```

```
> train_cont <- trainControl(method = "repeatedcv",
+ number = 10,
+ repeats = 5,
+ search = "random",
+ verboseIter = TRUE)
```

```
# Treinar o modelo
```

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

```
# O melhor parâmetro alpha escolhido é:
```

```
> elastic_reg$bestTune
alpha lambda
1 0.0728174 0.01250429
```

```
# E os parâmetros sao:
```

```
> elastic_reg[["finalModel"]][["beta"]]
15 x 84 sparse Matrix of class "dgCMatrix"
[[ suppressing 33 column names ‘s0’, ‘s1’, ‘s2’ ... ]]
husage . . . . . .
husunion . . . . . .
husearns . . . 0.001138484 0.006818273 0.012466340
```

```
huseduc . . . . 0.001161564 0.005825951
hushrs . . . . . .
age . . . . . .
educ . 0.006428553 0.01327619 0.020490484 0.027725149 0.034879840
husblck . . . . . .
hushisp . . . . . .
kidge6 . . . . . .
black . . . . . .
hispanic . . . . . .
union . . . . . .
kidlt6 . . . . . .
exper . . . . . .
...
husage ......
husunion ......
...
kidge6 ......
black ......
hispanic ......
union ......
kidlt6 ......
exper ......
.....suppressing 51 columns in show(); maybe adjust options(max.print=, width=)
.............................
```

Efetuar predições no modelo de treinamento:

```
> predictions_train <- predict(elastic_reg, x)
```

Calcular o R^2 dos valores verdadeiros e preditos conforme a seguinte função:

```
> eval_results <- function(true, predicted, df) {
+ SSE <- sum((predicted - true)^2)
+ SST <- sum((true - mean(true))^2)
+ R_square <- 1 - SSE / SST
+ RMSE = sqrt(SSE/nrow(df))
+
+ # As metricas de performace do modelo:
+ data.frame(
+ RMSE = RMSE,
+ Rsquare = R_square
+ )
+ }
```

As métricas de performance na base de treinamento são:

```
> eval_results(y_train, predictions_train, train)
```

```
RMSE Rsquare
1 0.8410461 0.2922977
```

Executar predições na base de teste

```
> predictions_test <- predict(elastic_reg, x_test)
```

As métricas de performance na base de teste são:

```
> eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.9894192 0.2589546
```

Comparando-se as métricas de treinamento e teste, podem-se observar erros muito altos, tanto na base de treinamento quanto na de teste.
O poder explicativo do modelo é baixo - R^2 = 29% / 25% (treinamento e teste).
Especula-se que o alto RMSE e o baixo valor do coeficiente de determinação se devam à ausência da variável earns (rendimento da esposa).

 # Predição para o exemplo

Efetuar predição para:

husage = 40 anos (idade do marido)

```
> husage = (40-pre_proc_val[["mean"]][["husage"]])/
```

husunion = 0 (marido não possui união estável)

```
> husunion = 0
```

husearns = 600 (US$ renda do marido por semana)

```
> husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
+ pre_proc_val[["std"]][["husearns"]]
```

huseduc = 13 (anos de estudo do marido)

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

husblck = 1 (o marido é preto)

```
> husblck = 1
```

hushisp = 0 (o marido não é hispânico)

```
> hushisp = 0
```

hushrs = 40 (horas semanais de trabalho do marido)

```
> hushrs = (40-pre_proc_val[["mean"]][["hushrs"]])/
+ pre_proc_val[["std"]][["hushrs"]]
```

kidge6 = 1 (possui filhos maiores de 6 anos)

```
> kidge6 = 1
```

age = 38 (anos - idade da esposa)

```
> age = (38-pre_proc_val[["mean"]][["age"]])/
+ pre_proc_val[["std"]][["age"]]
```

black = 0 (a esposa não é preta)

```
> black = 0
```

educ = 13 (anos de estudo da esposa)

```
> educ = (13-pre_proc_val[["mean"]][["educ"]])/
+ pre_proc_val[["std"]][["educ"]]
```

hispanic = 1 (a esposa é hispânica)

```
> hispanic = 1
```

union = 0 (esposa não possui união estável)

```
> union = 0
```

(possui filhos menores de 6 anos)

```
> kidlt6 = 1
```

#exper = 18 (anos de experiência de trabalho da esposa)

```
> exper = 18
```

Construir uma matriz de dados para a predição

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

Efetuar a predição com base nos parâmetros selecionados:

```
> predict_our_elastic <- predict(elastic_reg,our_pred)
```

O resultado da predição é:

```
predict_our_elastic
[1] -0.04678866
```

Converter o resultado (valor padronizado) para o valor nominal consistente com o dataset original

```
> wage_pred_elastic=(predict_our_elastic*
+ pre_proc_val[["std"]][["lwage"]])+
+ pre_proc_val[["mean"]][["lwage"]]
```

O resultado é:

```
> wage_pred_elastic
[1] 2.172166
```

#necessário aplicar o antilog

```
> exp(wage_pred_elastic)
[1] 8.777271
```

Este é o valor predito do salário por hora (US$) segundo as características atribuídas.

O intervalo de confiança para o nosso exemplo é:

```
> n <- nrow(train) # tamanho da amostra
> m <- exp(wage_pred_elastic) # valor medio predito corrigido pelo antilog
> s <- pre_proc_val[["std"]][["lwage"]] # desvio padrao
> dam <- s/sqrt(n) # distribuicao da amostragem da media
> CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
> CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior
```

Os valores são:

```
> CIlwr_elastic
[1] 8.755413
> CIupr_elastic
[1] 8.799129
```

Então, segundo as características atribuídas, o salário-hora da esposa é
em média **US$8.77** e pode
variar entre **US$8.75** e **US$8.79**

Mostrar em uma tabela as estatísticas dos modelos (RMSE e R2) e concluir qual o melhor modelo entre os três:

	Treinamento		Teste	
	RMSE	R^2	RMSE	R^2
REGRESSAO **RIDGE**	0.8411446	0.2921319	0.9893323	0.2590848
REGRESSAO **LASSO**	0.84203	0.290641	0.9894882	0.2588513
REGRESSAO **ELASTICNET**	0.8410461	0.2922977	0.9894192	0.2589546

O modelo **Ridge** se saiu ligeiramente melhor na base de teste, porém os números são tão próximos que seria possível afirmar que há um empate entre os três.

"Assim falaram os dados..."

Adendo – A título de comparação, foi feita uma simulação na base de dados, considerando a variável "earns":
Notou-se uma melhora considerável tanto na Raiz do Erro Quadrático Médio quanto no Coeficiente de Determinação.

Regressão **RIDGE:**

```
> # As metricas da base de treinamento sao:
> eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.5346909 0.7139667
> # Predicao e avaliacao nos dados de teste:
> predictions_test <- predict(ridge_reg,
+ s = best_lambda_ridge,
+ newx = x_test)
> # As metricas da base de teste sao:
> eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.7114185 0.6168803
```

Regressão **LASSO:**

```
> # As metricas da base de treinamento sao:
> eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.5351692 0.7134547
```

```
> # Vamos fazer as predicoes na base de teste
> predictions_test <- predict(lasso_model,
+ s = best_lambda_lasso,
+ newx = x_test)
> # As metricas da base de teste sao:
> eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.7097167 0.618711
```

Regressão **ELASTICNET:**

```
> # As metricas de performance na base de treinamento
> # sao:
> eval_results(y_train, predictions_train, train)
RMSE Rsquare
1 0.5346149 0.714048
> # Vamos fazer as predicoes na base de teste
> predictions_test <- predict(elastic_reg, x_test)
> # As metricas de performance na base de teste sao:
> eval_results(y_test, predictions_test, test)
RMSE Rsquare
1 0.7108196 0.6175252
```

Modelos com a variável "**earns**"	Treinamento		Teste	
	RMSE	**R^2**	**RMSE**	**R^2**
REGRESSAO **RIDGE**	0.5346909	0.7139667	0.7114185	0.6168803
REGRESSAO **LASSO**	0.5351692	0.7134547	0.7097167	0.618711
REGRESSAO **ELASTICNET**	0.5346149	0.714048	0.7108196	0.6175252

O modelo **Lasso** se saiu ligeiramente melhor na base de teste, porém, novamente, os números são tão próximos que seria possível afirmar que há um empate entre os três.

APÊNDICE 6 – ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

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

2 -

Para baixar direto da web e tratar arquivos compactados, sem o uso de arquivos locais.

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

# Definir as colunas para os dataframes
wine_colunas = ['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
'chlorides', 'free sulfur dioxide',
               'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol', 'quality']

# Baixar e abrir o arquivo zip
r = requests.get('https://archive.ics.uci.edu/static/public/186/wine+quality.zip')
z = zipfile.ZipFile(io.BytesIO(r.content))

# Ler os arquivos CSV
dadosfp_r = z.open('winequality-red.csv')
dadosfp_w = z.open('winequality-white.csv')

# Ler os dados dos arquivos CSV em dataframes
dados_r = pd.read_csv(dadosfp_r, sep=';', header=0, names=wine_colunas)
dados_w = pd.read_csv(dadosfp_w, sep=';', header=0, names=wine_colunas)

# Adicionar a coluna 'color' aos dataframes CODIFICAÇÃO
dados_r['color'] = 0 #'red'
dados_w['color'] = 1 #'white'

# Reordenar as colunas para colocar 'color' antes de 'quality'
colunas_reordenadas = wine_colunas[:-1] + ['color', 'quality']
dados_r = dados_r[colunas_reordenadas]
dados_w = dados_w[colunas_reordenadas]

# Concatenar os dataframes ( SELEÇÃO )
wine = pd.concat([dados_r, dados_w], ignore_index=True)

# Mostrar as primeiras linhas do dataframe concatenado
print(wine.head())

# Descrever o dataframe concatenado
print(wine.describe())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
```

```
4                11.0                 34.0   0.9978  3.51       0.56


   alcohol  color  quality
0      9.4      0        5
1      9.8      0        5
2      9.8      0        5
3      9.8      0        6
4      9.4      0        5
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    6497.000000       6497.000000  6497.000000     6497.000000
mean        7.215307          0.339666     0.318633        5.443235
std         1.296434          0.164636     0.145318        4.757804
min         3.800000          0.080000     0.000000        0.600000
25%         6.400000          0.230000     0.250000        1.800000
50%         7.000000          0.290000     0.310000        3.000000
75%         7.700000          0.400000     0.390000        8.100000
max        15.900000          1.580000     1.660000       65.800000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  6497.000000          6497.000000           6497.000000  6497.000000
mean      0.056034            30.525319            115.744574     0.994697
std       0.035034            17.749400             56.521855     0.002999
min       0.009000             1.000000              6.000000     0.987110
25%       0.038000            17.000000             77.000000     0.992340
50%       0.047000            29.000000            118.000000     0.994890
75%       0.065000            41.000000            156.000000     0.996990
max       0.611000           289.000000            440.000000     1.038980

                pH    sulphates      alcohol        color      quality
count  6497.000000  6497.000000  6497.000000  6497.000000  6497.000000
mean      3.218501     0.531268    10.491801     0.753886     5.818378
std       0.160787     0.148806     1.192712     0.430779     0.873255
min       2.720000     0.220000     8.000000     0.000000     3.000000
25%       3.110000     0.430000     9.500000     1.000000     5.000000
50%       3.210000     0.510000    10.300000     1.000000     6.000000
75%       3.320000     0.600000    11.300000     1.000000     6.000000
max       4.010000     2.000000    14.900000     1.000000     9.000000
```

```
wine.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         6497 non-null   float64
 1   volatile acidity      6497 non-null   float64
 2   citric acid           6497 non-null   float64
 3   residual sugar        6497 non-null   float64
 4   chlorides             6497 non-null   float64
 5   free sulfur dioxide   6497 non-null   float64
```

```
 6   total sulfur dioxide  6497 non-null   float64
 7   density               6497 non-null   float64
 8   pH                    6497 non-null   float64
 9   sulphates             6497 non-null   float64
 10  alcohol               6497 non-null   float64
 11  color                 6497 non-null   int64
 12  quality               6497 non-null   int64
dtypes: float64(11), int64(2)
memory usage: 660.0 KB
```

```
wine.shape, wine['quality']. value_counts()
((6497, 13),
 quality
 6    2836
 5    2138
 7    1079
 4     216
 8     193
 3      30
 9       5
 Name: count, dtype: int64)
```

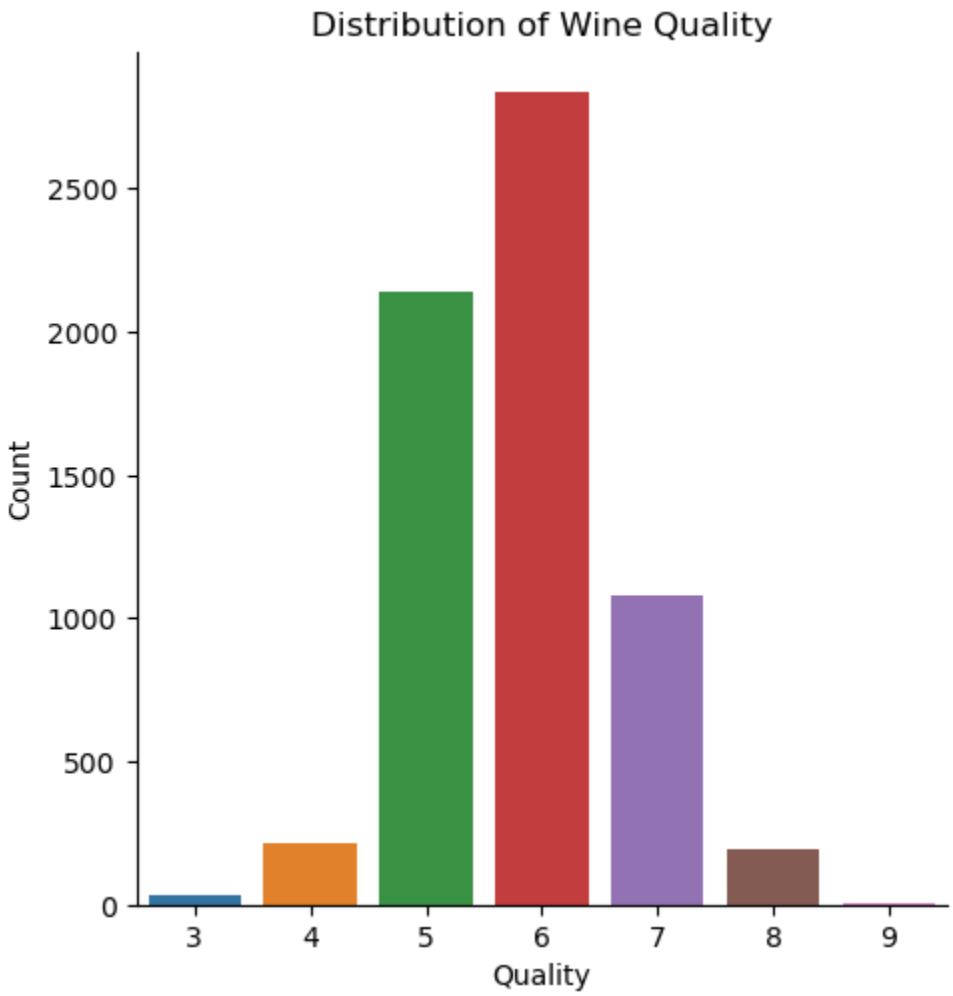
```
# Duplicatas
wine.groupby(list(wine.columns)).size().to_frame('Contagem').reset_index().quer
y('Contagem > 1')
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	color	quality	Contagem
27	4.9	0.335	0.14	1.3	0.036	69.0	168.0	0.99212	3.47	0.46	10.466667	1	5	2
28	4.9	0.345	0.34	1.0	0.068	32.0	143.0	0.99138	3.24	0.40	10.100000	1	5	2
38	5.0	0.270	0.32	4.5	0.032	58.0	178.0	0.98956	3.45	0.31	12.600000	1	7	2
43	5.0	0.330	0.16	1.5	0.049	10.0	97.0	0.99170	3.48	0.44	10.700000	1	6	3
46	5.0	0.350	0.25	7.8	0.031	24.0	116.0	0.99241	3.39	0.40	11.300000	1	6	2

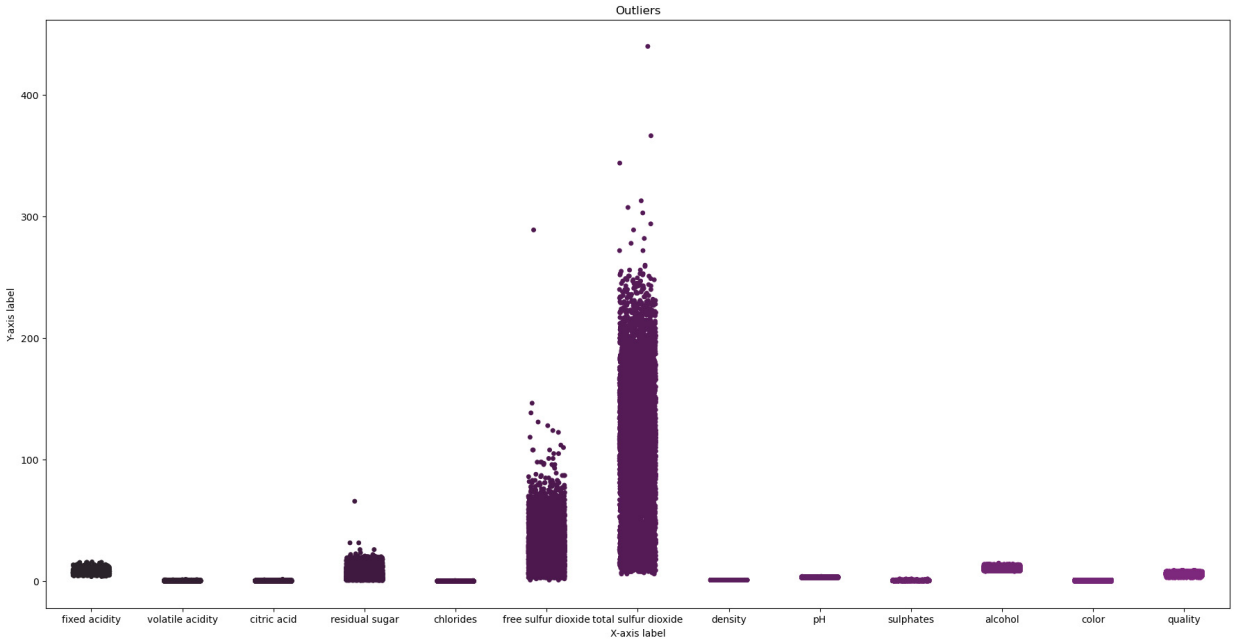
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...												...	...	...
5302	13.0	0.470	0.49	4.3	0.085	6.0	47.0	1.00210	3.30	0.68	12.700000	0	6	2
5304	13.2	0.460	0.52	2.2	0.071	12.0	35.0	1.00060	3.10	0.56	9.000000	0	6	2
5310	13.7	0.415	0.68	2.9	0.085	17.0	43.0	1.00140	3.06	0.80	10.000000	0	6	2
5315	15.0	0.210	0.44	2.2	0.075	10.0	24.0	1.00005	3.07	0.84	9.200000	0	7	2
5316	15.5	0.645	0.49	4.2	0.095	10.0	23.0	1.00315	2.92	0.74	11.100000	0	5	2

992 rows × 14 columns

```
import matplotlib.pyplot as plt
import seaborn as sns
# Create a count plot using seaborn
sns.catplot(data=wine, x='quality', kind='count')

# Add labels and title to the plot
plt.title('Distribution of Wine Quality')
plt.xlabel('Quality')
plt.ylabel('Count')

# Display the plot
plt.show()
```

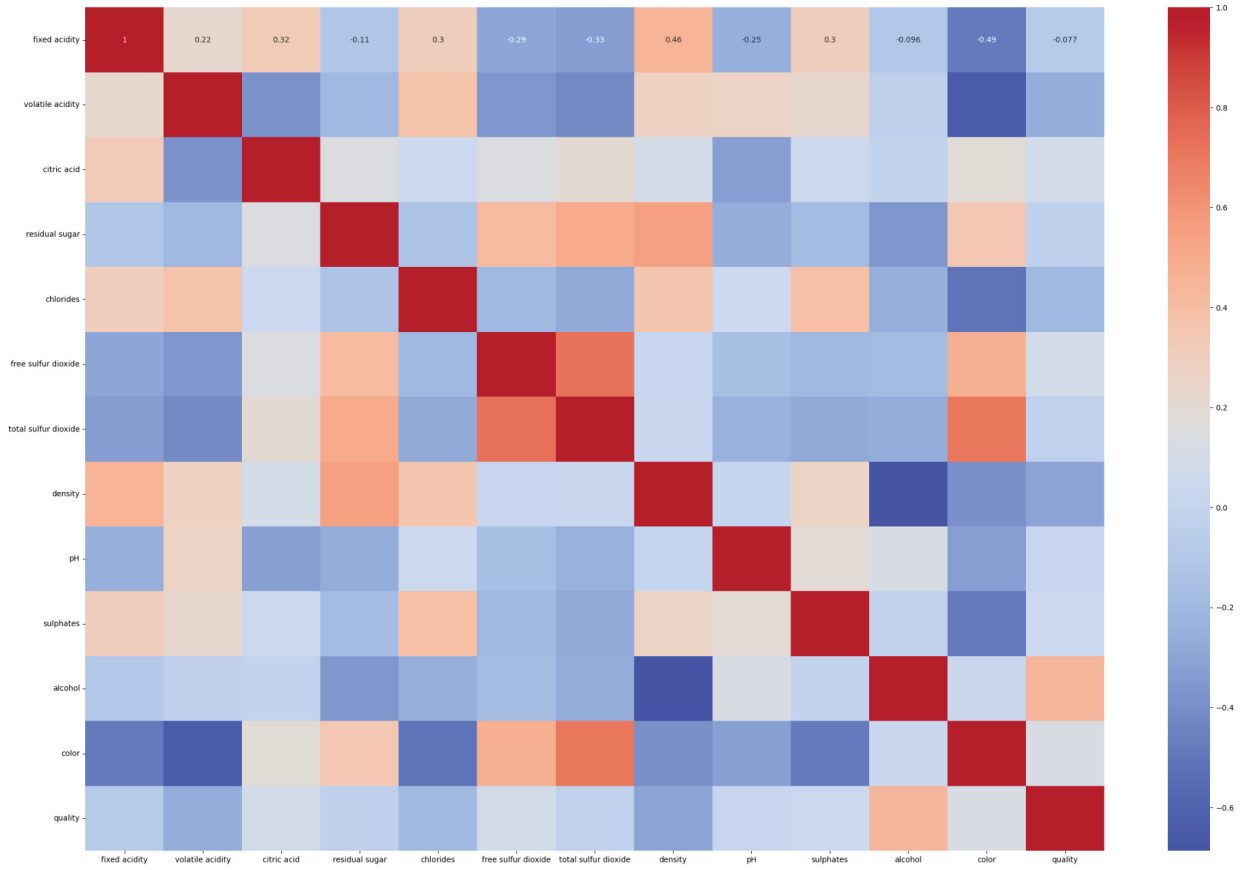



```
# Set the figure size
plt.figure(figsize=(22, 11))

# Add outliers to the plot
sns.stripplot(data=wine, color="purple", jitter=0.2, size=5)

# Set the axis labels and title
plt.title("Outliers")
plt.xlabel("X-axis label")
plt.ylabel("Y-axis label")

# Show the plot
plt.show()
```

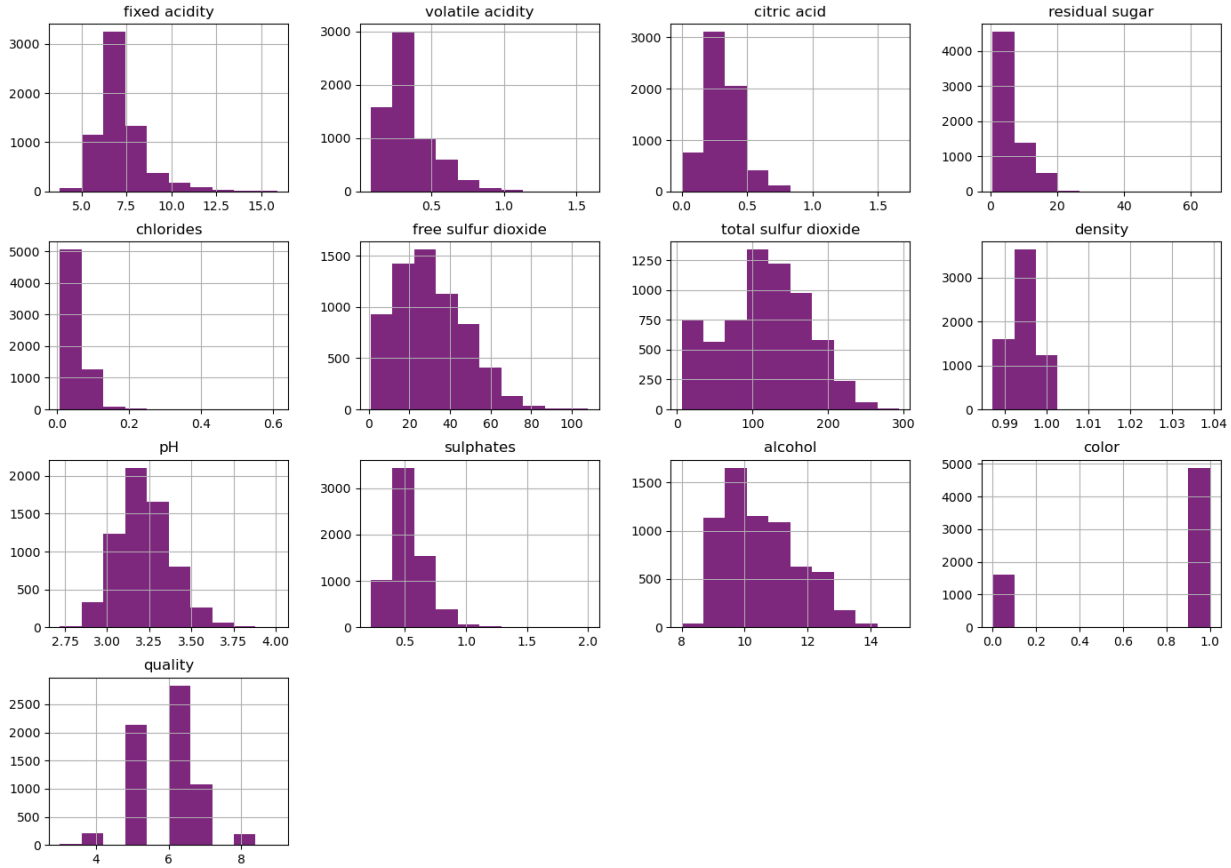


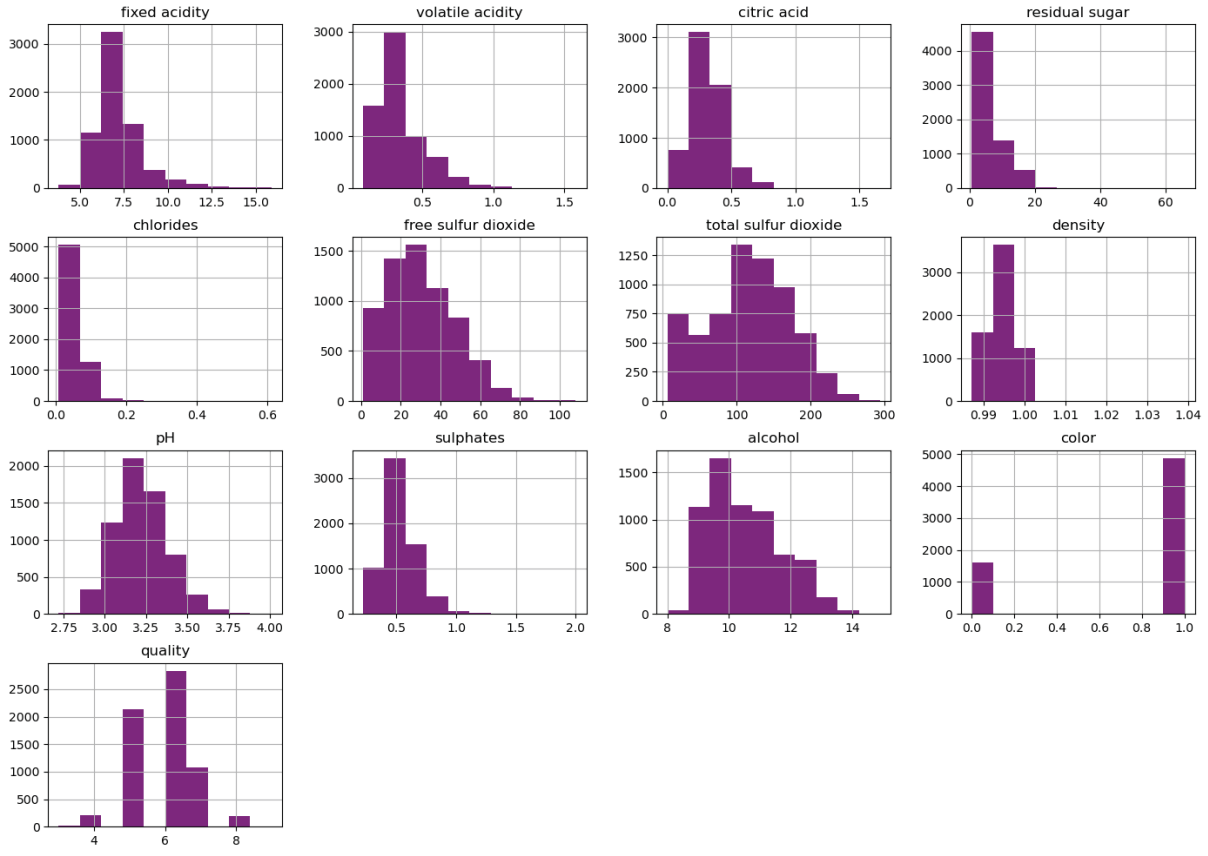

```
# Delete the outliers
# The data before deleting outliers
print("Before Removing the outliers", wine.shape)

# Deleting outliers (Filtering by conditions on each column)
wine = wine[ (wine['total sulfur dioxide'] < 300) & (wine['free sulfur dioxide'] < 110) ]

#The data after deleting outliers
print("After Removing the outliers", wine.shape)

Before Removing the outliers (6497, 13)
After Removing the outliers (6484, 13)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
#olhando a correlação entre os atributos
plt.figure(figsize=(30,20))
wine_cor = wine.corr()
#sns.set_theme(font_scale=1.3)
sns.heatmap(wine_cor,annot=True,cmap='coolwarm')
plt.show()
```




```
# PLotting Histogram

wine.hist(figsize=(17,12),color='Purple')
plt.show()
```


fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
color
quality


fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
color
quality

```
#pairplot

sns.pairplot(wine, hue = 'quality')
plt.show()
```


```
"""
X = wine.iloc[:,2:]
cols = wine_colunas[2:]
print(X.head())
Y = wine['quality']
Y_orig = wine['quality']
print(Y.unique())
```

```
"""

# Selecionar todas as colunas, a partir da primeira, exceto a última
X = wine.iloc[:, :-1]
cols = wine_colunas[:-1]

print(X.head())

# Selecionar a última coluna como Y
Y = wine.iloc[:, -1]
Y_orig = wine.iloc[:, -1]

# Imprimir valores únicos em 'quality'
print(Y.unique())
print(X.shape, Y.shape, Y_orig.shape)
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

    free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                  11.0                  34.0   0.9978  3.51       0.56
1                  25.0                  67.0   0.9968  3.20       0.68
2                  15.0                  54.0   0.9970  3.26       0.65
3                  17.0                  60.0   0.9980  3.16       0.58
4                  11.0                  34.0   0.9978  3.51       0.56

    alcohol  color
0       9.4      0
1       9.8      0
2       9.8      0
3       9.8      0
4       9.4      0
[5 6 7 4 8 3 9]
(6484, 12) (6484,) (6484,)
```

```
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd
from imblearn.over_sampling import SMOTE
from sklearn.feature_selection import SelectKBest, f_classif

X_orig =  X.copy()

print(X_orig.head())
```

```
print(Y_orig.unique() )

"""
# Remover duplicatas
X = X.drop_duplicates()
Y = Y.iloc[X.index]
"""


# Aplicar SMOTE para CORREÇÃO DE PREVALÊNCIA
smote = SMOTE(k_neighbors=4)
X, Y = smote.fit_resample(X, Y)

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


"""selector = SelectKBest(score_func=f_classif, k=6)
X = selector.fit_transform(X, Y)
selected_columns = [col for col, support in zip(cols, selector.get_support()) if support]
X = pd.DataFrame(X, columns=selected_columns)"""


"""
z_scores = np.abs(zscore(X))
outliers = z_scores > 3
X[outliers] = np.sign(X[outliers]) * 3
Y = Y.iloc[X.index]
"""

print(X_orig.head())
print(X.head())

print(X.shape, X_orig.shape, Y.shape, Y_orig.shape)

# A normalização pode ser manual, usando o pandas... ex:
#X['radius_mean']        =        (X_orig['radius_mean']-X_orig['radius_mean'].mean())        /
X_orig['radius_mean'].std()
# normalização min-max
#X['texture_mean']       =       (X_orig['texture_mean']-X_orig['texture_mean'].min())       /
(X_orig['texture_mean'].max()-X_orig['texture_mean'].min())

   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4             0.70         0.00             1.9      0.076
1            7.8             0.88         0.00             2.6      0.098
2            7.8             0.76         0.04             2.3      0.092
3           11.2             0.28         0.56             1.9      0.075
4            7.4             0.70         0.00             1.9      0.076

    free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                  11.0                  34.0   0.9978  3.51       0.56
1                  25.0                  67.0   0.9968  3.20       0.68
2                  15.0                  54.0   0.9970  3.26       0.65
```

```
3                17.0                  60.0   0.9980  3.16       0.58
4                11.0                  34.0   0.9978  3.51       0.56

   alcohol  color
0      9.4      0
1      9.8      0
2      9.8      0
3      9.8      0
4      9.4      0
[5 6 7 4 8 3 9]
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076

   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  color
0      9.4      0
1      9.8      0
2      9.8      0
3      9.8      0
4      9.4      0
         0         1         2         3         4         5         6   \
0  0.297521  0.413333  0.000000  0.019939  0.111296  0.093458  0.097222
1  0.330579  0.533333  0.000000  0.030675  0.147841  0.224299  0.211806
2  0.330579  0.453333  0.024096  0.026074  0.137874  0.130841  0.166667
3  0.611570  0.133333  0.337349  0.019939  0.109635  0.149533  0.187500
4  0.297521  0.413333  0.000000  0.019939  0.111296  0.093458  0.097222

         7         8         9         10   11
0  0.206092  0.612403  0.191011  0.202899  0.0
1  0.186813  0.372093  0.258427  0.260870  0.0
2  0.190669  0.418605  0.241573  0.260870  0.0
3  0.209948  0.341085  0.202247  0.260870  0.0
4  0.206092  0.612403  0.191011  0.202899  0.0
(19845, 12) (6484, 12) (19845,) (6484,)
```

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

# com os dados originais
X_oring_train, X_orig_test, y_orig_train, y_orig_test = train_test_split(X_orig,
                      Y_orig, test_size=0.25, stratify=Y_orig,random_state=10)

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

```
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
[[   0    0    0   19    0    0    0]
 [   0    0    5  155    0    0    0]
 [   0    0  153 1447    0    0    0]
 [   0    0  124 2002    0    0    0]
 [   0    0   16  793    0    0    0]
 [   0    0    4  141    0    0    0]
 [   0    0    0    4    0    0    0]]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00        19
           4       0.00      0.00      0.00       160
```

```
           5       0.51      0.10      0.16      1600
           6       0.44      0.94      0.60      2126
           7       0.00      0.00      0.00       809
           8       0.00      0.00      0.00       145
           9       0.00      0.00      0.00         4

    accuracy                           0.44      4863
   macro avg       0.14      0.15      0.11      4863
weighted avg       0.36      0.44      0.31      4863

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[  0   0   1   5   0   0   0]
 [  0   0   1  53   0   0   0]
 [  0   0  43 490   0   0   0]
 [  0   0  48 661   0   0   0]
 [  0   0   5 265   0   0   0]
 [  0   0   2  46   0   0   0]
 [  0   0   0   1   0   0   0]]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         6
           4       0.00      0.00      0.00        54
           5       0.43      0.08      0.14       533
           6       0.43      0.93      0.59       709
           7       0.00      0.00      0.00       270
           8       0.00      0.00      0.00        48
           9       0.00      0.00      0.00         1

    accuracy                           0.43      1621
   macro avg       0.12      0.14      0.10      1621
weighted avg       0.33      0.43      0.30      1621
```

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

treinador = svm.SVC()  #algoritmo escolhido

modelo = treinador.fit(X_train, y_train)

# predição com os mesmos dados usados para treinar
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))

# predição com os mesmos dados usados para testar
```

```
print('Matriz de confusão - com os dados ORIGINAIS 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
[[1605  229  128  125   39    0    0]
 [ 222 1377  353  128   23   23    0]
 [ 163  378 1137  300   94   50    5]
 [  81  172  491  653  424  279   26]
 [  39   41  126  324  885  651   60]
 [  84   55    7   99  442 1358   81]
 [   0    0    0    0    0    0 2126]]
              precision    recall  f1-score   support

           3       0.73      0.75      0.74      2126
           4       0.61      0.65      0.63      2126
           5       0.51      0.53      0.52      2127
           6       0.40      0.31      0.35      2126
           7       0.46      0.42      0.44      2126
           8       0.58      0.64      0.61      2126
           9       0.93      1.00      0.96      2126

    accuracy                           0.61     14883
   macro avg       0.60      0.61      0.61     14883
weighted avg       0.60      0.61      0.61     14883

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[552  74  32  41  10   0   0]
 [ 89 418 139  37  11  14   1]
 [ 53 127 372 110  32  14   0]
 [ 24  59 156 206 156 104   4]
 [ 12  14  32 103 302 226  20]
 [ 41  21   3  39 134 435  36]
 [  0   0   0   0   0   0 709]]
              precision    recall  f1-score   support

           3       0.72      0.78      0.75       709
           4       0.59      0.59      0.59       709
           5       0.51      0.53      0.52       708
           6       0.38      0.29      0.33       709
           7       0.47      0.43      0.45       709
           8       0.55      0.61      0.58       709
           9       0.92      1.00      0.96       709

    accuracy                           0.60      4962
   macro avg       0.59      0.60      0.59      4962
weighted avg       0.59      0.60      0.59      4962
```

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

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

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

CLASSIFICAÇÃO

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – CV	Size = 41 decay = 0.7	0.856	**Reference** **Prediction** bus opel saab van bus 42 0 0 0 opel 1 31 11 0 saab 0 11 31 0 van 0 0 1 39
SVM – CV	C = 100 Sigma = 0.015	0.856	**Reference** **Prediction** bus opel saab van bus 42 0 1 0 opel 0 33 13 0 saab 0 9 29 0 van 1 0 2 39
RF – Hold-out	Mtry = 2	0.808	**Reference** **Prediction** bus opel saab van bus 43 0 1 0 opel 0 27 13 0 saab 0 14 26 0 van 0 1 3 39
RF – CV	Mtry = 9	0.796	**Reference** **Prediction** bus opel saab van bus 42 0 2 0 opel 0 28 14 0 saab 0 14 24 0 van 1 0 3 39
SVM – Hold-out	C = 1 Sigma = 0.065	0.766	**Reference** **Prediction** bus opel saab van bus 42 0 1 0 opel 0 24 17 0 saab 0 18 23 0 van 1 0 2 39
KNN	k = 1	0.751	**Reference** **Prediction** bus opel saab van bus 47 1 1 3 opel 1 23 11 1 saab 3 15 24 1 van 1 2 2 33
RNA – Hold-out	Size = 5 decay = 0.1	0.677	**Reference** **Prediction** bus opel saab van bus 37 0 2 0 opel 0 3 3 2 saab 4 39 36 0 van 2 0 2 37

 # RNA – CV

```
> print(resultado)
Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis
844 106 54 101 222 67 12 222 30 25 173 228
845 86 36 78 146 58 7 135 50 18 124 155
846 85 36 66 123 55 5 120 56 17 128 140
ScVarmaxis RaGyr SkewMaxis Skewmaxis Kurtmaxis KurtMaxis HollRa predict.rna
```

```
844 721 200 70 3 4 187 201 saab
845 270 148 66 0 25 190 195 opel
846 212 131 73 1 18 186 190 van
```

Instalação dos pacotes necessários

```
install.packages("caret")
install.packages("e1071")
install.packages("mlbench")
install.packages("mice")

library(caret)
library(mlbench)
library(mice)
```

Leitura dos dados

```
getwd()
setwd("C:/Users/guilherme.kroger/Documents/Pós Graduação/UFPR/Inteligência Artificial
Aplicada/IAA008 -
Aprendizado de Máquina/08 - Trabalho/Dados")
dados <- read.csv("6 - Veiculos - Dados.csv", header=T)
View(dados)
str(dados)
```

Tratar o Id e Missing Values

```
dados$a <- NULL
#temp_dados$Id <- NULL
#imp <- mice(temp_dados)
#dados <- complete(imp, 1)

dados_novos_casos <- dados[844:846, ]
dados_novos_casos
dados_novos_casos$tipo <- "?"
dados <- dados[1:843, ]
```

Criar bases de Treino e Teste

```
set.seed(202491)
indices <- createDataPartition(dados$tipo, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

Treinar o modelo com Hold-out

```
set.seed(202491)
rna <- train(tipo~., data=treino, method="nnet",trace=FALSE)
```

```
rna
```

Predições dos valores do conjunto de teste

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

Matriz de confusão

```
confusionMatrix(predict.rna, as.factor(teste$tipo))
```

#Usando Cross-validation

indica o método cv e numero de folders 10

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

executa a RNA com esse ctrl

```
set.seed(202491)
rna <- train(tipo~., data=treino, method="nnet",trace=FALSE, trControl=ctrl)

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

#Parametrização da RNA

size, decay

```
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay = seq(from = 0.1, to = 0.9,
by = 0.3))
set.seed(202491)
rna <- train(
form = tipo~. ,
data = treino ,
method = "nnet" ,
tuneGrid = grid ,
trControl = ctrl ,
maxit = 2000,trace=FALSE)
```

Verifica o resultado do Treinamento

```
rna
```

Faz as predições e mostra matriz de confusão

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

PREDIÇÕES DE NOVOS CASOS

```
View(dados_novos_casos)

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

 # SVM – CV

```
> print(resultado)
Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis
844 106 54 101 222 67 12 222 30 25 173 228
845 86 36 78 146 58 7 135 50 18 124 155
846 85 36 66 123 55 5 120 56 17 128 140
ScVarmaxis RaGyr SkewMaxis Skewmaxis Kurtmaxis KurtMaxis HollRa predict.svm
844 721 200 70 3 4 187 201 saab
845 270 148 66 0 25 190 195 opel
846 212 131 73 1 18 186 190 van
```

Pacotes necessários:

```
install.packages("e1071")
install.packages("kernlab")
install.packages("caret")
install.packages("mice")

library("caret")
library(mice)
```

Leitura dos dados

```
getwd()
setwd("C:/Users/guilherme.kroger/Documents/Pós     Graduação/UFPR/Inteligência     Artificial
Aplicada/IAA008 -
Aprendizado de Máquina/08 - Trabalho/Dados")
dados <- read.csv("6 - Veiculos - Dados.csv", header=T)
View(dados)
str(dados)
```

Tratar o Id e Missing Values

```
dados$a <- NULL
#temp_dados$Id <- NULL
#imp <- mice(temp_dados)
#dados <- complete(imp, 1)

dados_novos_casos <- dados[844:846, ]
dados_novos_casos
dados_novos_casos$tipo <- "?"
dados <- dados[1:843, ]
```

Criar bases de Treino e Teste

```
set.seed(202491)
indices <- createDataPartition(dados$tipo, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

Treinar o modelo com Hold-out

```
set.seed(202491)
svm <- train(tipo~., data=treino, method="svmRadial",trace=FALSE)
svm
```

Aplicar modelos treinados na base de Teste

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

Matriz de confusão

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

Cross-validation SVM

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

executa a RNA com esse ctrl

```
set.seed(202491)
svm <- train(tipo~., data=treino, method="svmRadial",trace=FALSE, trControl=ctrl)
svm
```

Matriz de confusão

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

Vários C e sigma

```
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202491)
svm <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl, tuneGrid=tuneGrid)
svm
```

Matriz de confusão

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

PREDIÇÕES DE NOVOS CASOS

```
dados_novos_casos$Id <- NULL
View(dados_novos_casos)
predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
resultado$tipo <- NULL
View(resultado)
```

CLASSIFICAÇÃO

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	C = 2 Sigma = 0.015	0.763	**Reference** **Prediction** neg pos neg 87 24 pos 12 29
RF – CV	Mtry = 5	0.757	**Reference** **Prediction** neg pos neg 84 22 pos 15 31
SVM – Hold-out	C = 1 Sigma = 0.13	0.750	**Reference** **Prediction** neg pos neg 85 24 pos 14 29
RF – Hold-out	Mtry = 5	0.750	**Reference** **Prediction** neg pos neg 83 22 pos 16 31
RNA – Hold-out	Size = 5 decay = 0.1	0.737	**Reference** **Prediction** neg pos neg 84 25 pos 15 28
KNN	k = 9	0.712	**Reference** **Prediction** neg pos neg 84 27 pos 17 25
RNA – CV	Size = 11 decay = 0.4	0.664	**Reference** **Prediction** neg pos neg 77 29 pos 22 24

SVM – CV

```
> print(resultado)
preg0nt glucose pressure triceps insulin mass pedigree age predict.svm
766 5 121 72 23 112 26.2 0.245 30 neg
767 1 126 60 0 0 30.1 0.349 47 neg
768 1 93 70 31 0 30.4 0.315 23 neg
```

Pacotes necessários:

```
install.packages("e1071")
install.packages("kernlab")
install.packages("caret")
install.packages("mice")

library("caret")
library(mice)
```

Leitura dos dados

```
getwd()
setwd("C:/Users/guilherme.kroger/Documents/Pós     Graduação/UFPR/Inteligência     Artificial
Aplicada/IAA008 -
Aprendizado de Máquina/08 - Trabalho/Dados")
dados <- read.csv("10 - Diabetes - Dados.csv", header=T)
View(dados)
str(dados)
```

Tratar o Id e Missing Values

```
dados$num <- NULL
#temp_dados$Id <- NULL
#imp <- mice(temp_dados)
#dados <- complete(imp, 1)
dados_novos_casos <- dados[766:768, ]
dados_novos_casos
dados_novos_casos$diabetes <- "?"
dados <- dados[1:765, ]
```

Criar bases de Treino e Teste

```
set.seed(202491)
indices <- createDataPartition(dados$diabetes, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

Treinar o modelo com Hold-out

```
set.seed(202491)
svm <- train(diabetes~., data=treino, method="svmRadial",trace=FALSE)
svm
```

Aplicar modelos treinados na base de Teste

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

Matriz de confusão

```
confusionMatrix(predict.svm, as.factor(teste$diabetes))
```

Cross-validation SVM

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

executa a SVM com esse ctrl

```
set.seed(202491)
svm <- train(diabetes~., data=treino, method="svmRadial",trace=FALSE, trControl=ctrl)
svm
```

Matriz de confusão

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

Vários C e sigma

```
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202491)
svm <- train(diabetes~., data=treino, method="svmRadial", trControl=ctrl, tuneGrid=tuneGrid)
svm
```

Matriz de confusão

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

PREDIÇÕES DE NOVOS CASOS

```
dados_novos_casos$Id <- NULL
View(dados_novos_casos)
predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
resultado$diabetes <- NULL
View(resultado)
```

REGRESSÃO

Para o experimento de Regressão:

· Ordenar por R2 descendente, ou seja, a técnica de melhor R2 ficará em primeiro na tabela.

· Após o quadro, colocar:

- Um **resultado com 3 linhas** com a predição de **novos casos** para a técnica/parâmetro de **maior R2** (criar um arquivo com novos casos à sua escolha)
- O **Gráfico de Resíduos** para a técnica/parâmetro de maior R2
- A **lista de comandos** emitidos no RStudio para conseguir os resultados obtidos

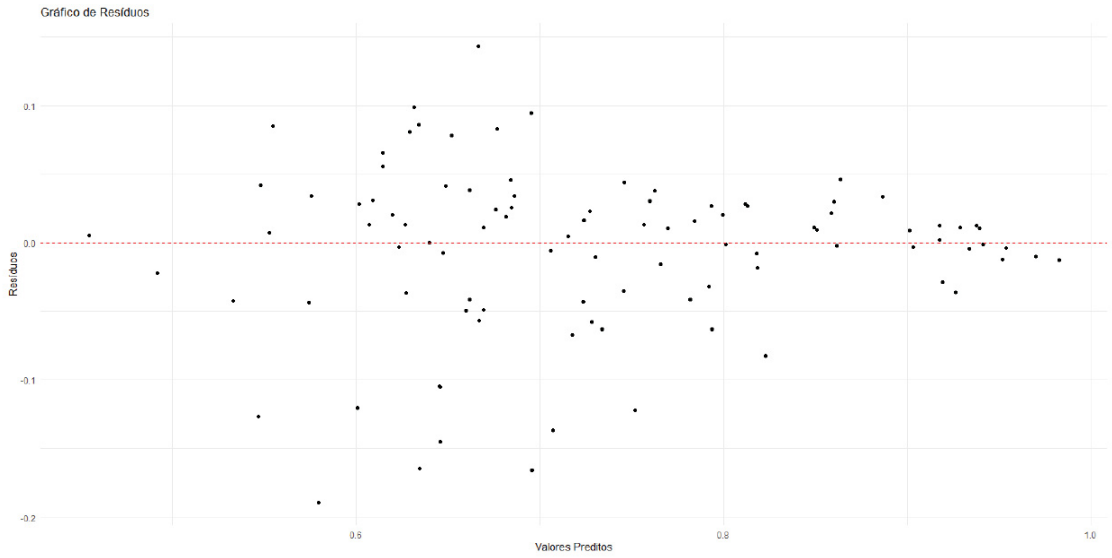
REGRESSÃO

Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV	C = 2 Sigma = 0.01	0.81	0.060	0.90	0.060	0.043
RF – Hold-out	Mtry = 2	0.81	0.061	0.90	0.061	0.044
RF – CV	Mtry = 2	0.81	0.061	0.90	0.061	0.044
RNA – CV	Size = 10 decay = 0.1	0.78	0.065	0.89	0.065	0.048
SVM – Hold-out	C = 0.5 Sigma = 0.2	0.77	0.067	0.88	0.067	0.047
RNA – Hold-out	Size = 5 decay = 0.1	0.76	0.068	0.87	0.068	0.051
KNN	k = 9	0.75	0.190	0.87	0.07	0.057

SVM – CV

```
> print(resultado)
GRE.Score TOEFL.Score University.Rating SOP LOR CGPA Research predict.svm
498 330 120 5 4.5 5.0 9.6 1 0.94
499 312 103 4 4.0 5.0 8.4 0 0.71
500 327 113 4 4.5 4.5 9.0 0 0.83
```



Instalação dos pacotes (são os mesmos da classificação)

```
install.packages("kernlab")
install.packages("caret")
install.packages("e1071")
install.packages("mlbench")
install.packages("mice")
install.packages("ggplot2")
library(ggplot2)
library(mlbench)
library(caret)
library(mice)
```

Leitura dos dados

```
getwd()
setwd("C:/Users/guilherme.kroger/Documents/Pós Graduação/UFPR/Inteligência Artificial
Aplicada/IAA008 -
Aprendizado de Máquina/08 - Trabalho/Dados")
dados <- read.csv("9 - Admissao - Dados.csv", header=T)
View(dados)
str(dados)
```

Retira o atributo ID

```
dados$num <- NULL
dados_novos_casos <- dados[498:500, ]
dados_novos_casos
dados_novos_casos$ChanceOfAdmit <- "?"
dados <- dados[1:497, ]
```

Cria arquivo de treino e teste

```
set.seed(202491)
indices <- createDataPartition(dados$ChanceOfAdmit, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

Treino com Hold-Out

```
set.seed(202491)
svm <- train(ChanceOfAdmit~., data=treino, method="svmRadial")
svm
```

Aplicar modelos treinados na base de Teste

```
predicoes.svm <- predict(svm, teste)
```

Pacote para cálculo das métriscas (rmse)

```
install.packages("Metrics")
library(Metrics)
```

Mostra as métricas

RMSE - Raiz Quadrada do Erro Médio

```
rmse(teste$ChanceOfAdmit, predicoes.svm)
```

R^2 - Coeficiente de Determinação Múltipla

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

#Syx (Erro padrão da estimativa)

```
syx <- function(predito, observado) {
n <- length(observado)
#p <- length(svm$finalModel$k)
sqrt(sum((observado - predito)^2) / (n))
}
syx_value <- syx(predicoes.svm, teste$ChanceOfAdmit)
cat("Erro padrão da estimativa (Syx):", syx_value, "\n")
```

#Coeficiente de Pearson

```
pearson_correlation <- cor(teste$ChanceOfAdmit, predicoes.svm, method = "pearson")
pearson_correlation
```

MAE - Média Absoluta do Erro

```
mae_value <- mae(teste$ChanceOfAdmit, predicoes.svm)
print(paste("MAE:", mae_value))
```

CV e parametrizacao da svm

```
control <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(size = seq(from = 1, to = 10, by = 1), decay = seq(from = 0.1, to =
0.9, by = 0.3))
set.seed(202491)
svm    <-    train(ChanceOfAdmit~.,    data=treino,    method="nnet",    trainControl=control,
tuneGrid=tuneGrid, linout=T,
MaxNWts=10000, maxit=2000, trace=F)
svm
```

Predições e métricas

```
predicoes.svm <- predict(svm, teste)
rmse(teste$ChanceOfAdmit, predicoes.svm)
r2(predicoes.svm, teste$ChanceOfAdmit)
```

#Syx (Erro padrão da estimativa)

```
syx <- function(predito, observado) {
n <- length(observado)
#p <- length(svm$finalModel$k)
sqrt(sum((observado - predito)^2) / (n))
}
syx_value <- syx(predicoes.svm, teste$ChanceOfAdmit)
cat("Erro padrão da estimativa (Syx):", syx_value, "\n")
```

#Coeficiente de Pearson

```
pearson_correlation <- cor(teste$ChanceOfAdmit, predicoes.svm, method = "pearson")
pearson_correlation
```

MAE - Média Absoluta do Erro

```
```

```
mae_value <- mae(teste$ChanceOfAdmit, predicoes.svm)
print(paste("MAE:", mae_value))
```

Vários C e sigma

```
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
14
set.seed(202491)
svm <- train(ChanceOfAdmit~., data=treino, method="svmRadial", trControl=control,
tuneGrid=tuneGrid)
svm
```

Aplicar modelos treinados na base de Teste

```
predicoes.svm <- predict(svm, teste)
```

Gráfico de Resíduos

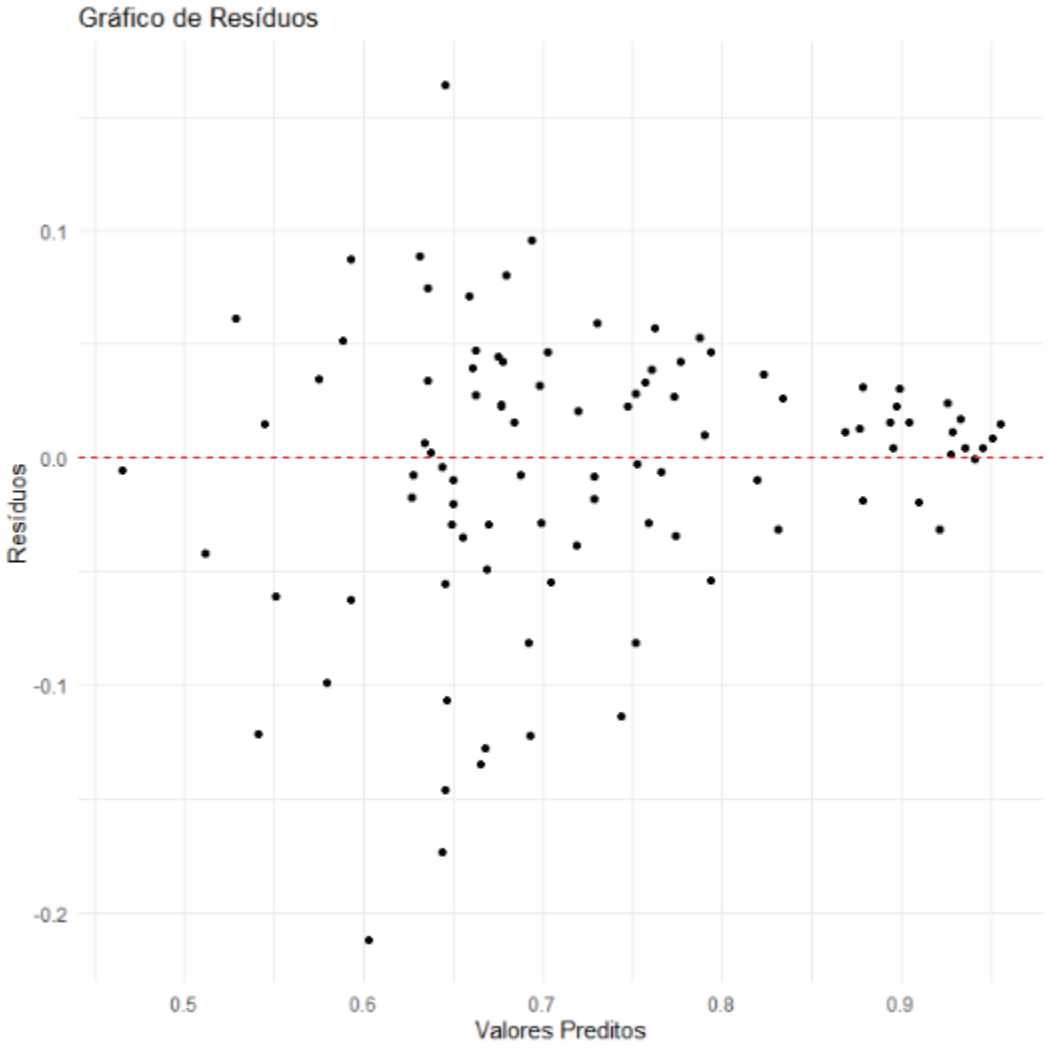
```
residuos <- teste$ChanceOfAdmit - predicoes.svm
residuo_df <- data.frame(Preditos = predicoes.svm, Residuos = residuos)

ggplot(residuo_df, aes(x = Preditos, y = Residuos)) +
geom_point() +
geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
labs(title = "Gráfico de Resíduos", x = "Valores Preditos", y = "Resíduos") +
theme_minimal()
```

PREDIÇÕES DE NOVOS CASOS

```
View(dados_novos_casos)
dados_novos_casos$ChanceOfAdmit <- NULL
predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
View(resultado)
```

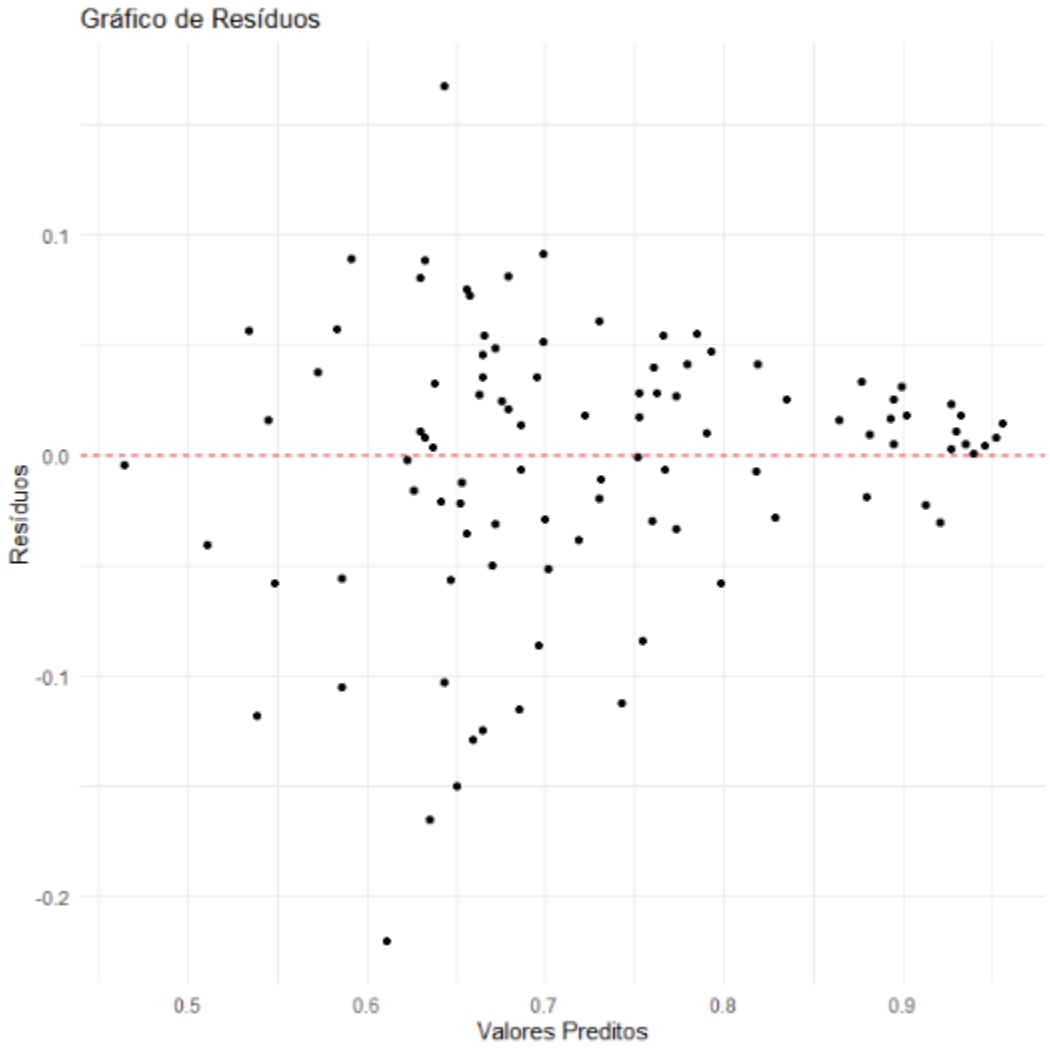
RF – Hold-out

```
> print(resultado)
GRE.Score TOEFL.Score University.Rating SOP LOR CGPA Research predict.rf
498 330 120 5 4.5 5.0 9.6 1 0.94
499 312 103 4 4.0 5.0 8.4 0 0.72
500 327 113 4 4.5 4.5 9.0 0 0.81
```



RF – CV

```
> print(resultado)
GRE.Score TOEFL.Score University.Rating SOP LOR CGPA Research predict.rf
498 330 120 5 4.5 5.0 9.6 1 0.94
499 312 103 4 4.0 5.0 8.4 0 0.72
500 327 113 4 4.5 4.5 9.0 0 0.80
```



Pacotes necessários:

```
install.packages("e1071")
install.packages("kernlab")
install.packages("caret")
install.packages("Metrics")
library(ggplot2)
library(mlbench)
library(caret)
library(mice)
library(Metrics)
```

Leitura dos dados

```
getwd()
setwd("C:/Users/guilherme.kroger/Documents/Pós Graduação/UFPR/Inteligência Artificial
Aplicada/IAA008 -
Aprendizado de Máquina/08 - Trabalho/Dados")
```

```
dados <- read.csv("9 - Admissao - Dados.csv", header=T)
View(dados)
str(dados)
```

Retira o atributo ID

```
dados$num <- NULL
dados_novos_casos <- dados[498:500, ]
dados_novos_casos
dados_novos_casos$ChanceOfAdmit <- "?"
dados <- dados[1:497, ]
```

Cria arquivo de treino e teste

```
set.seed(202491)
indices <- createDataPartition(dados$ChanceOfAdmit, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

Treinar Randon Forest com a base de Treino

```
set.seed(202491)
rf <- train(ChanceOfAdmit~., data=treino, method="rf")
rf
```

Aplicar modelos treinados na base de Teste

```
predicoes.rf <- predict(rf, teste)
```

Mostra as métricas

RMSE - Raiz Quadrada do Erro Médio

```
rmse(teste$ChanceOfAdmit, predicoes.rf)
```

R^2 - Coeficiente de Determinação Múltipla

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

#Syx (Erro padrão da estimativa)

```
syx <- function(predito, observado) {
n <- length(observado)
#p <- length(svm$finalModel$k)
sqrt(sum((observado - predito)^2) / (n))
}
syx_value <- syx(predicoes.rf, teste$ChanceOfAdmit)
cat("Erro padrão da estimativa (Syx):", syx_value, "\n")
```

#Coeficiente de Pearson

```
pearson_correlation <- cor(teste$ChanceOfAdmit, predicoes.rf, method = "pearson")
pearson_correlation
```

MAE - Média Absoluta do Erro

```
mae_value <- mae(teste$ChanceOfAdmit, predicoes.rf)
print(paste("MAE:", mae_value))
```

Gráfico de Resíduos

```
residuos <- teste$ChanceOfAdmit - predicoes.rf
residuo_df <- data.frame(Preditos = predicoes.rf, Residuos = residuos)
ggplot(residuo_df, aes(x = Preditos, y = Residuos)) +
geom_point() +
geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
labs(title = "Gráfico de Resíduos", x = "Valores Preditos", y = "Resíduos") +
theme_minimal()
```

Cross-validation RF

```
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202491)
rf <- train(ChanceOfAdmit~., data=treino, method="rf", trControl=ctrl)
rf
predicoes.rf <- predict(rf, teste)
```

Ver Métricas

Vários mtry

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

```
set.seed(202491)
rf <- train(ChanceOfAdmit~., data=treino, method="rf", trControl=ctrl, tuneGrid=tuneGrid)
rf
predicoes.rf <- predict(rf, teste)
```

```
# Ver Métricas
```

```
### PREDIÇÕES DE NOVOS CASOS
```

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

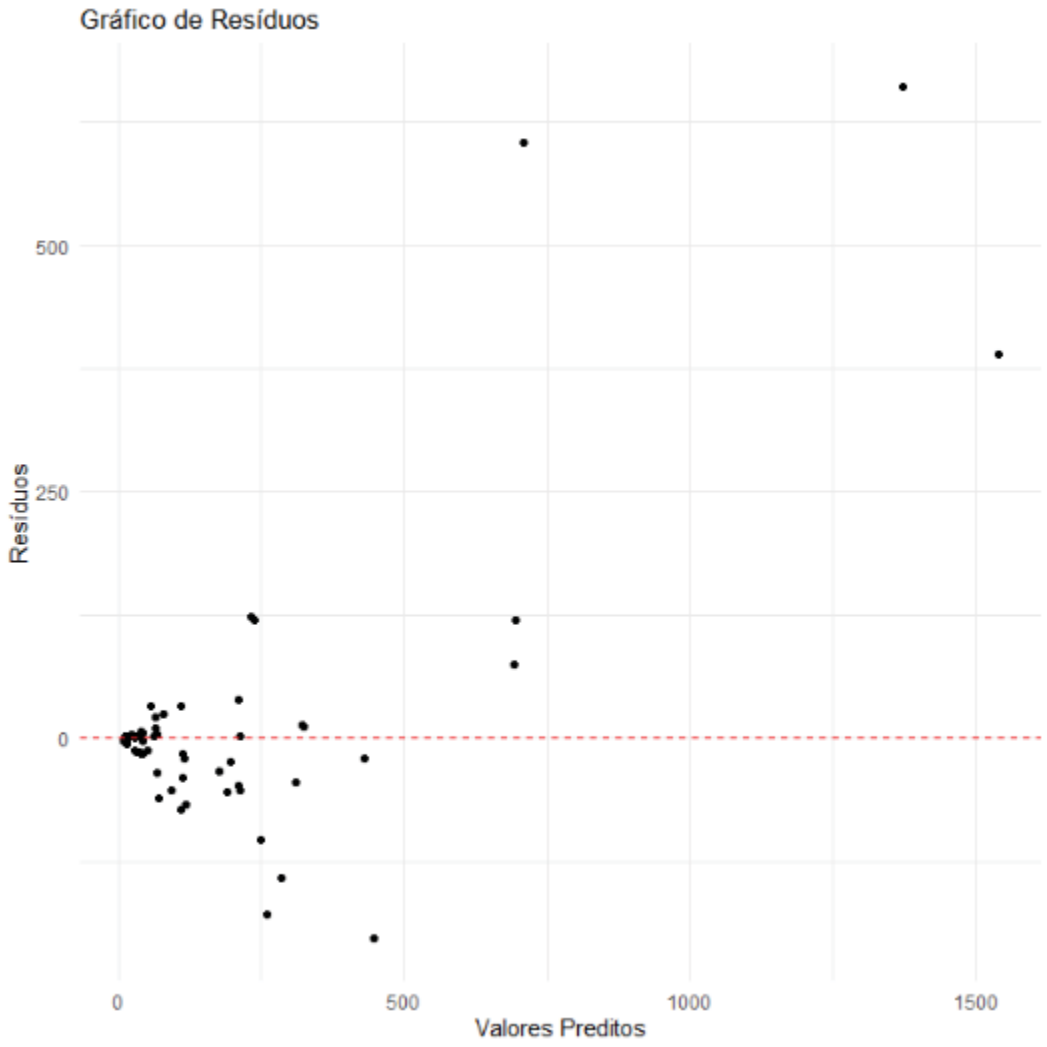
REGRESSÃO

Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	Mtry = 2	0.88	142	0.97	142	64.47
RF – CV	Mtry = 2	0.88	140	0.97	140	63.71
RNA – Hold-out	Size = 3 decay = 0.1	0.85	155	0.95	155	87.70
KNN	K = 1	0.84	162	0.96	160	79.07
SVM – CV	C = 100 Sigma = 0.01	0.84	163	0.96	163	109.01
SVM – Hold-out	C = 1 Sigma = 1.1	0.78	188	0.91	188	118.12
RNA – CV	Size = 6 decay = 0.7	0.63	245	0.80	245	91.05

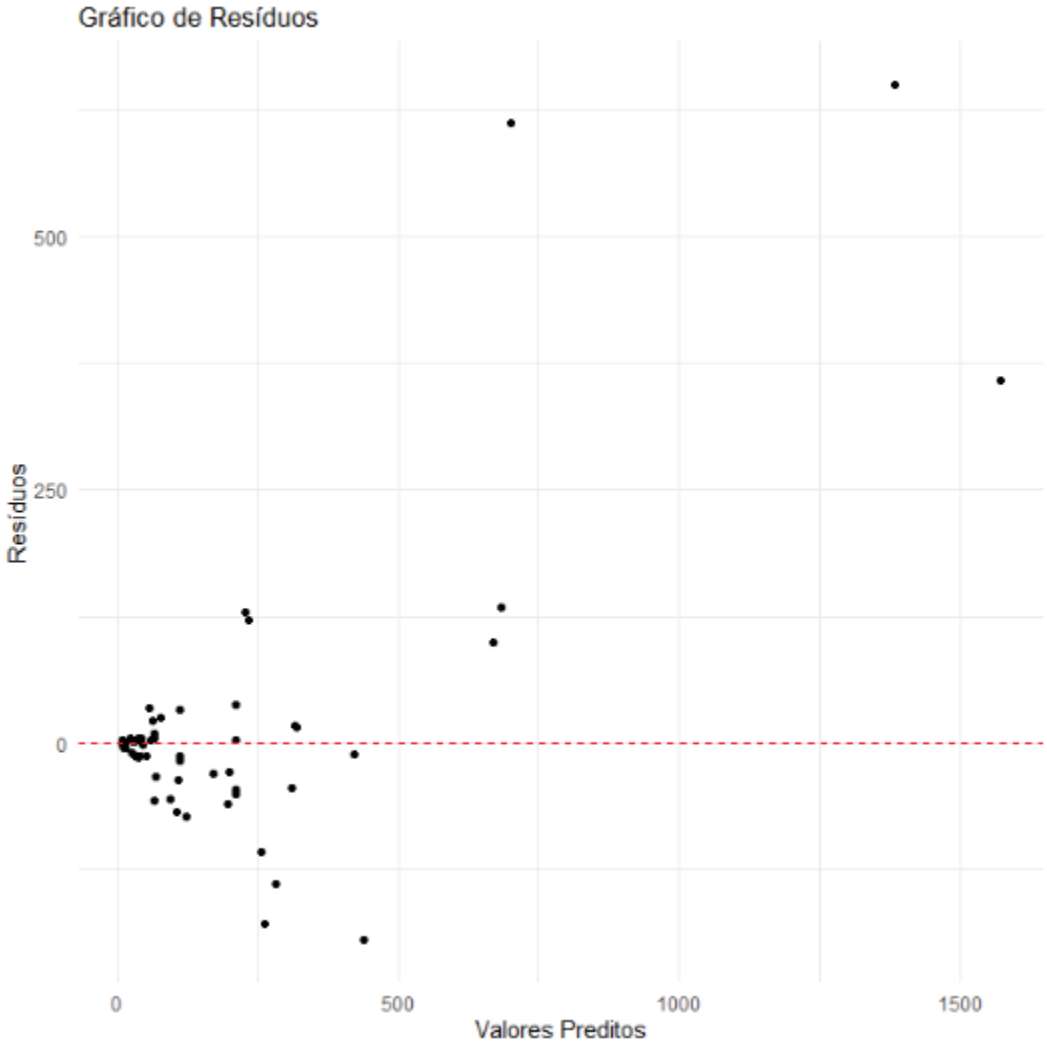
RF – Hold-out

```
> print(resultado)
dap h Me predict.rf
298 28 26 0.65 707
299 21 16 0.85 242
300 45 28 0.85 1937
```



RF – CV

```
> print(resultado)
dap h Me predict.rf
298 28 26 0.65 703
299 21 16 0.85 238
300 45 28 0.85 2029
```



Pacotes necessários:

```
install.packages("e1071")
install.packages("kernlab")
install.packages("caret")
install.packages("Metrics")
library(ggplot2)
library(mlbench)
library(caret)
library(mice)
library(Metrics)
```

Leitura dos dados

```
getwd()
```

```
setwd("C:/Users/guilherme.kroger/Documents/Pós Graduação/UFPR/Inteligência Artificial Aplicada/IAA008 -
Aprendizado de Máquina/08 - Trabalho/Dados")
dados <- read.csv("5 - Biomassa - Dados.csv", header=T)
View(dados)
str(dados)
```

Retira o atributo ID

#dados$num <- NULL

```
dados_novos_casos <- dados[298:300, ]
dados_novos_casos
dados_novos_casos$biomassa <- "?"
dados <- dados[1:297, ]
```

Cria arquivo de treino e teste

```
set.seed(202491)
indices <- createDataPartition(dados$biomassa, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

Treinar Randon Forest com a base de Treino

```
set.seed(202491)
rf <- train(biomassa~., data=treino, method="rf")
rf
```

Aplicar modelos treinados na base de Teste

```
predicoes.rf <- predict(rf, teste)
```

Mostra as métricas

RMSE - Raiz Quadrada do Erro Médio

```
rmse(teste$biomassa, predicoes.rf)
```

R^2 - Coeficiente de Determinação Múltipla

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

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

#Syx (Erro padrão da estimativa)

```
syx <- function(predito, observado) {
n <- length(observado)
#p <- length(svm$finalModel$k)
sqrt(sum((observado - predito)^2) / (n))
}
syx_value <- syx(predicoes.rf, teste$biomassa)
cat("Erro padrão da estimativa (Syx):", syx_value, "\n")
```

#Coeficiente de Pearson

```
pearson_correlation <- cor(teste$biomassa, predicoes.rf, method = "pearson")
pearson_correlation
```

MAE - Média Absoluta do Erro

```
mae_value <- mae(teste$biomassa, predicoes.rf)
print(paste("MAE:", mae_value))
```

Gráfico de Resíduos

```
residuos <- teste$biomassa - predicoes.rf
residuo_df <- data.frame(Preditos = predicoes.rf, Residuos = residuos)

ggplot(residuo_df, aes(x = Preditos, y = Residuos)) +
geom_point() +
geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
labs(title = "Gráfico de Resíduos", x = "Valores Preditos", y = "Resíduos") +
theme_minimal()
```

Cross-validation RF

```
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202491)
rf <- train(biomassa~., data=treino, method="rf", trControl=ctrl)
rf
predicoes.rf <- predict(rf, teste)
```

Ver Métricas

Vários mtry

```
tuneGrid = expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202491)
rf <- train(biomassa~., data=treino, method="rf", trControl=ctrl, tuneGrid=tuneGrid)
rf
predicoes.rf <- predict(rf, teste)
```

Ver Métricas

PREDIÇÕES DE NOVOS CASOS

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

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

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

Leitura dos dados

```
getwd()
setwd("C:/Users/guilherme.kroger/Documents/Pós Graduação/UFPR/Inteligência Artificial
Aplicada/IAA008 - Aprendizado de Máquina/08 - Trabalho/Dados")
dados <- read.csv("6 - Veiculos - Dados.csv", header=T)
View(dados)
str(dados)
```

Retira o atributo ID

```
dados$a <- NULL
```

Gráfico com os dados

```
ggplot(dados,
aes(Comp,Circ,DCirc,RadRa,PrAxisRa,MaxLRa,ScatRa,Elong,PrAxisRect,MaxLRect,ScVarMaxis,ScVarma
xis
,RaGyr,SkewMaxis,Skewmaxis,Kurtmaxis,KurtMaxis,HollRa, color = tipo)) + geom_point()
```

Executa o Kmeans

```
set.seed(202491)
veiculosCluster <- kmeans(dados[, 1:18], 10)
veiculosCluster
table(veiculosCluster$cluster, dados$tipo)
```

Cria um arquivo com todos os registros e mais os clusters de cada um

```
resultado <- cbind(dados, veiculosCluster$cluster)
resultado
```

REGRAS DE ASSOCIAÇÃO

Musculação

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

```
> summary(rules)
set of 155 rules

rule length distribution (lhs + rhs):sizes
2 3 4 5
23 56 55 21

Min. 1st Qu. Median Mean 3rd Qu. Max.
2.0 3.0 3.0 3.5 4.0 5.0

summary of quality measures:
       support        confidence     coverage        lift           count
       Min. :0.04     Min. :0.70      Min. :0.04    Min. :0.9      Min. : 1.0
       1st Qu.:0.08   1st Qu.:0.86   1st Qu.:0.08   1st Qu.:1.5     1st Qu.: 2.0
       Median :0.08    Median :1.00  Median :0.08   Median :1.7     Median : 2.0
       Mean :0.15     Mean :0.93     Mean :0.17     Mean :1.7      Mean : 3.8
       3rd Qu.:0.23   3rd Qu.:1.00   3rd Qu.:0.27   3rd Qu.:2.0     3rd Qu.: 6.0
       Max. :0.46      Max. :1.00     Max. :0.65    Max. :3.2      Max. :12.0

mining info:
data ntransactions support confidence
dados 26 0.001 0.7

call
apriori(data = dados, parameter = list(supp = 0.001, conf = 0.7, minlen = 2))
```

```
> ### Vamos ver as 5 primeiras regras ordenadas pela confiança:
> options(digits=2)
> inspect(sort(rules[1:5], by="confidence"))
lhs rhs support confidence coverage lift coun
t
[1] {Crucifixo} => {Afundo} 0.077 1.00 0.077 2.89 2
[2] {Crucifixo} => {Gemeos} 0.077 1.00 0.077 1.53 2
[3] {Crucifixo} => {LegPress} 0.077 1.00 0.077 1.24 2
[4] {Adutor} => {Agachamento} 0.115 1.00 0.115 3.25 3
[5] {Adutor} => {LegPress} 0.115 1.00 0.115 1.24 3
```

Instalação dos pacotes necessários

```
install.packages('arules', dep=T)
library(arules)
library(datasets)
```

Leitura dos dados

```
getwd()
```

```
setwd("C:/Users/guilherme.kroger/Documents/Pós     Graduação/UFPR/Inteligência     Artificial
Aplicada/IAA008 -
Aprendizado de Máquina/08 - Trabalho/Dados")
```

Frequência dos 10 primeiros itens:

```
dados <- read.transactions(file="2 - Musculacao - Dados.csv",format="basket",sep=";")
inspect(dados[1:5])
dados
```

Visão geral dos dados:

```
itemFrequencyPlot(dados, topN=10, type='absolute')
```

Obter as regras:
Primeiramente definimos o Suporte=0,001 e Confiança=0,7

```
set.seed(202491)
rules <- apriori(dados, parameter = list(supp = 0.001, conf = 0.7, minlen=2))
summary(rules)
```

5 primeiras regras ordenadas pela confiança:

```
options(digits=2)
inspect(sort(rules[1:5], by="confidence"))
set.seed(202491)
rules <- apriori(data=dados, parameter=list(supp=0.001,conf = 0.1,minlen=2),appearance =
list(default='rhs',lhs='Crucifixo'),
control = list(verbose=F))
inspect(sort(rules, by='confidence', decreasing=T))
```

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

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

1 -

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

Carga da base

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

# Já está separado em dados de treino e teste
# Não precisa separar
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
```

Normalização dos dados

```
# Normalização os dados
# Imagens em pixels de 0 - 255
# / 255.0 transforma em 0 - 1
x_train, x_test = x_train / 255.0, x_test / 255.0

# O dado y é a classe a qual faz parte
# O flattem torna os dados vetorizados
y_train, y_test = y_train.flatten(), y_test.flatten()

# Dimensão dos dados
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)
```

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

Rede completa

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

```
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)
# Model ( lista entrada, lista saída)
model = Model(i, x)
```

relatório sobre a arquitetura da rede

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

```
Model: "functional"
```

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

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

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

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

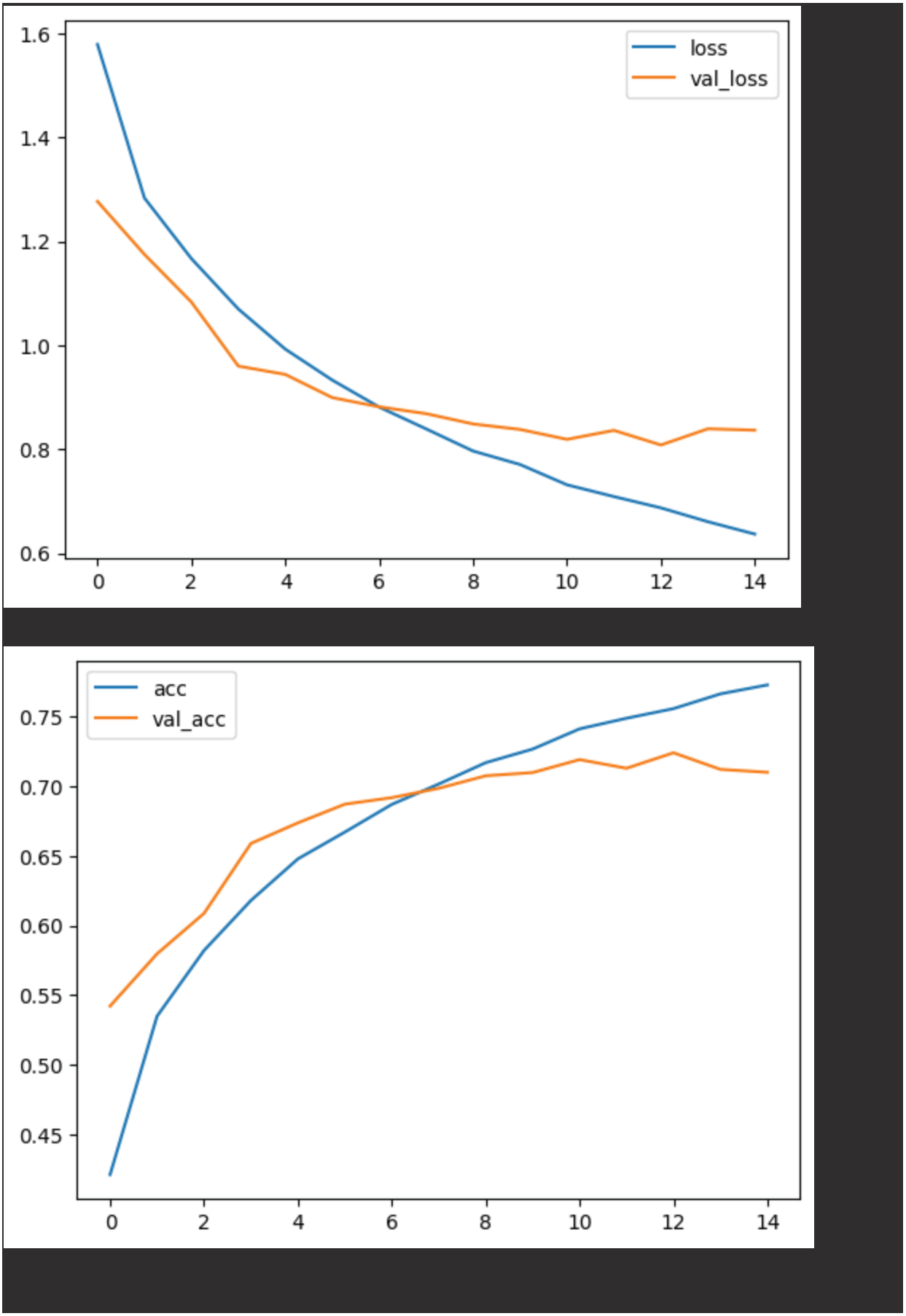
Compilar e treinar o modelo

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

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

```
Epoch 1/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 66s 41ms/step - accuracy: 0.3478 - loss: 1.7659 - val_accuracy: 0.5423 - val_loss: 1.2769
Epoch 2/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 78s 38ms/step - accuracy: 0.5271 - loss: 1.3082 - val_accuracy: 0.5798 - val_loss: 1.1750
Epoch 3/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 63s 40ms/step - accuracy: 0.5765 - loss: 1.1786 - val_accuracy: 0.6088 - val_loss: 1.0833
Epoch 4/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 80s 39ms/step - accuracy: 0.6112 - loss: 1.0797 - val_accuracy: 0.6590 - val_loss: 0.9600
Epoch 5/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 64s 41ms/step - accuracy: 0.6477 - loss: 0.9868 - val_accuracy: 0.6738 - val_loss: 0.9440
Epoch 6/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 79s 39ms/step - accuracy: 0.6684 - loss: 0.9296 - val_accuracy: 0.6872 - val_loss: 0.8994
Epoch 7/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 60s 39ms/step - accuracy: 0.6893 - loss: 0.8713 - val_accuracy: 0.6919 - val_loss: 0.8816
Epoch 8/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 64s 41ms/step - accuracy: 0.7072 - loss: 0.8246 - val_accuracy: 0.6986 - val_loss: 0.8685
Epoch 9/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 60s 38ms/step - accuracy: 0.7224 - loss: 0.7806 - val_accuracy: 0.7076 - val_loss: 0.8486
Epoch 10/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 85s 40ms/step - accuracy: 0.7335 - loss: 0.7506 - val_accuracy: 0.7099 - val_loss: 0.8381
Epoch 11/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 82s 40ms/step - accuracy: 0.7480 - loss: 0.7145 - val_accuracy: 0.7192 - val_loss: 0.8190
Epoch 12/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 79s 39ms/step - accuracy: 0.7521 - loss: 0.6932 - val_accuracy: 0.7130 - val_loss: 0.8363
Epoch 13/15
...
Epoch 14/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 79s 39ms/step - accuracy: 0.7721 - loss: 0.6484 - val_accuracy: 0.7122 - val_loss: 0.8393
Epoch 15/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 81s 38ms/step - accuracy: 0.7796 - loss: 0.6177 - val_accuracy: 0.7101 - val_loss: 0.8367
```

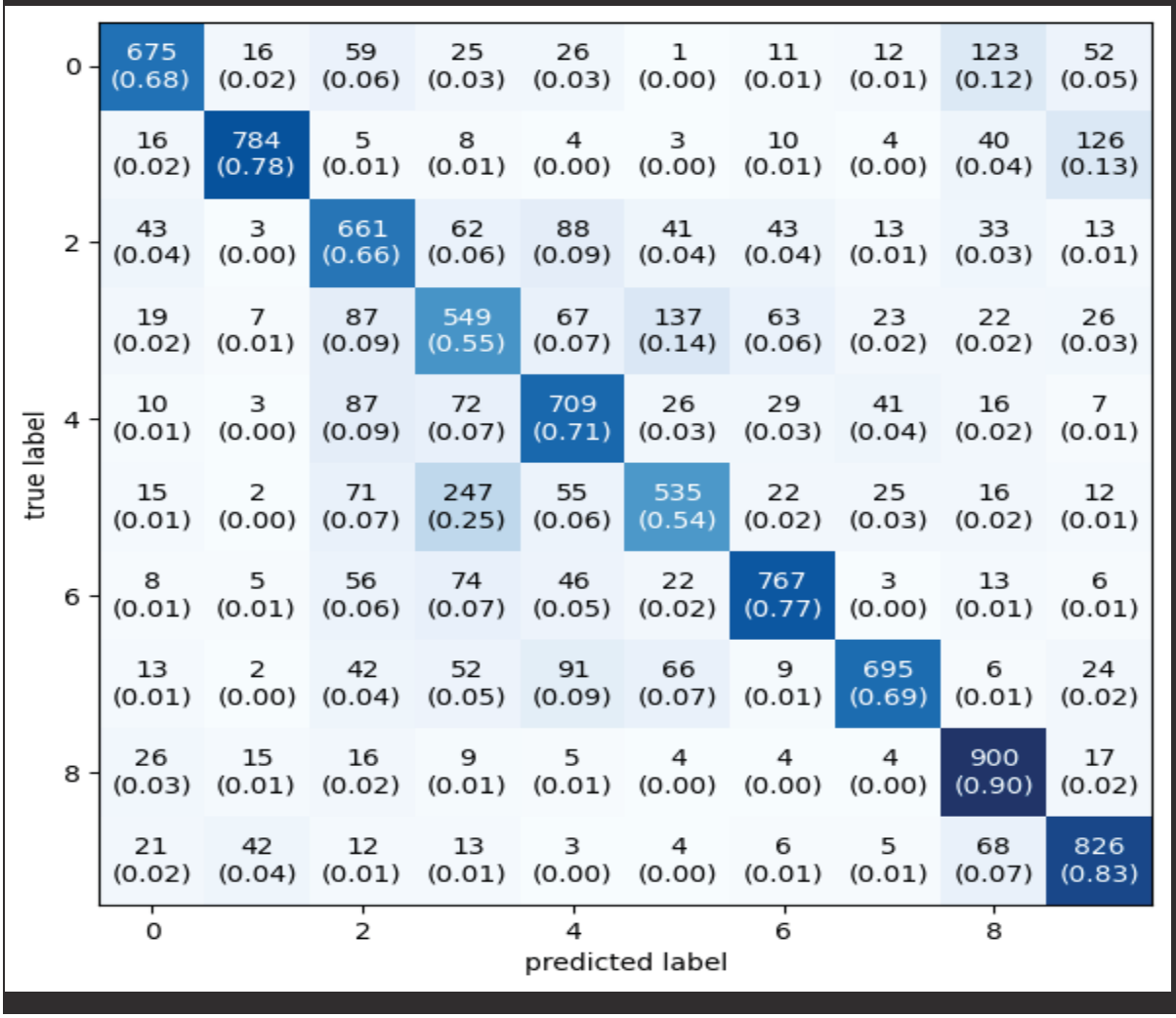
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
• Mostrar os gráficos do treino: loss e acurácia

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

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


loss
val_loss
1.6
1.4
1.2
1.0
0.8
0.6
0
2
4
6
8
10
12
14
acc
val_acc
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0
2
4
6
8
10
12
14

```
# Efetuar predição e mostrar Matriz de Confusão

# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)

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

313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step
```

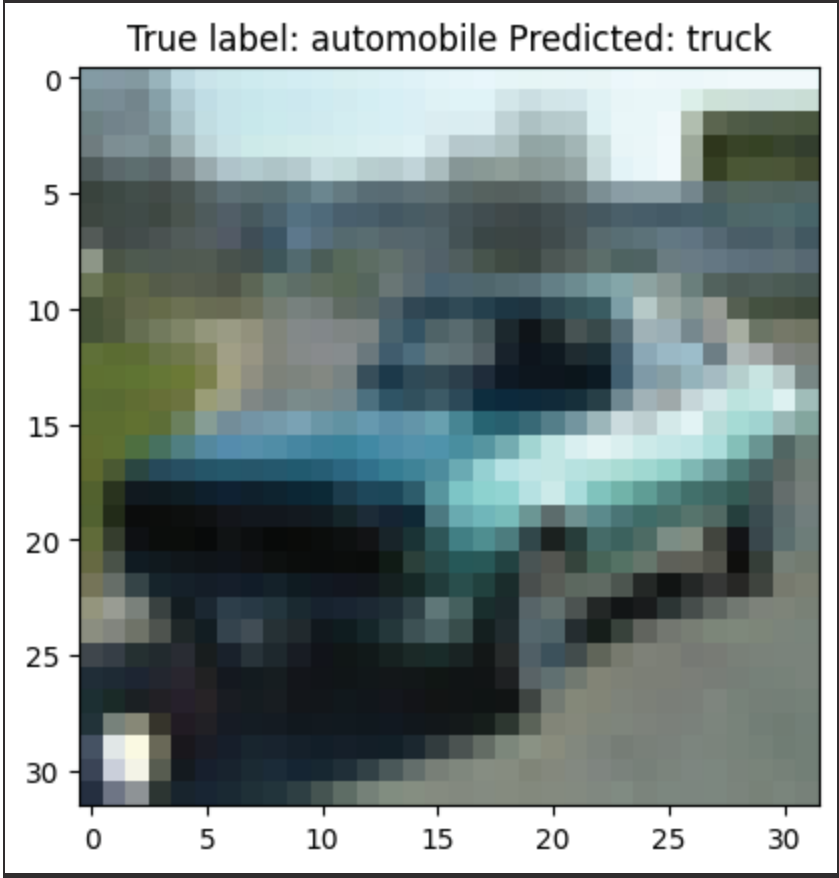
```
(<Figure size 700x700 with 1 Axes>,
 <Axes: xlabel='predicted label', ylabel='true label'>)
```



◉ Mostrar classificações erradas

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

```
Text(0.5, 1.0, 'True label: automobile Predicted: truck')
```



2 -

Importação das Bibliotecas

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

Carga da Base

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

```
--2024-08-17 20:16:17--  https://lazyprogrammer.me/course_files/spam.csv
Resolving   lazyprogrammer.me   (lazyprogrammer.me)...   104.21.23.210,   172.67.213.166,
2606:4700:3031::6815:17d2, ...
Connecting to lazyprogrammer.me (lazyprogrammer.me)|104.21.23.210|:443... 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 0.03s

2024-08-17 20:16:17 (15.3 MB/s) - 'spam.csv' saved [503663/503663]
```

Separação em Treino e Teste

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

```
test_size=0.33)
```

```
# Acerta a tokenização

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

```
7222 tokens
```

```
# Acerta o tamanho das sequências (padding)

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

data_train.shape:  (3733, 189)
data_test.shape:  (1839, 189)
```

```
# Definição do modelo

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

```
# Sumário do modelo
```

```
model.summary()
```

```
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 189)	0
embedding (Embedding)	(None, 189, 20)	144,460
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6

```
Total params: 144,986 (566.35 KB)

Trainable params: 144,986 (566.35 KB)

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

Compila e treina o modelo

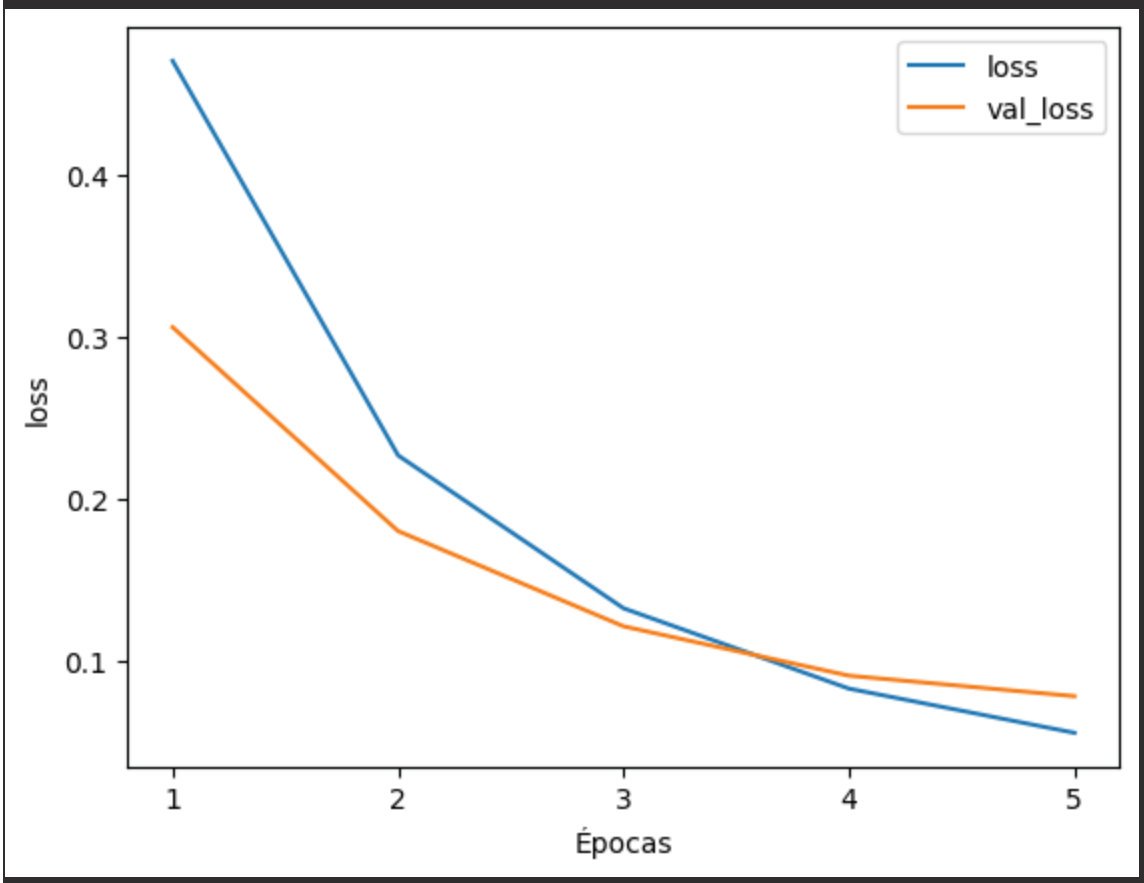
```
# Compila e treina o modelo
model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])
epochs = 5
r = model.fit(data_train, y_train, epochs=epochs, validation_data=(data_test,
y_test))
```

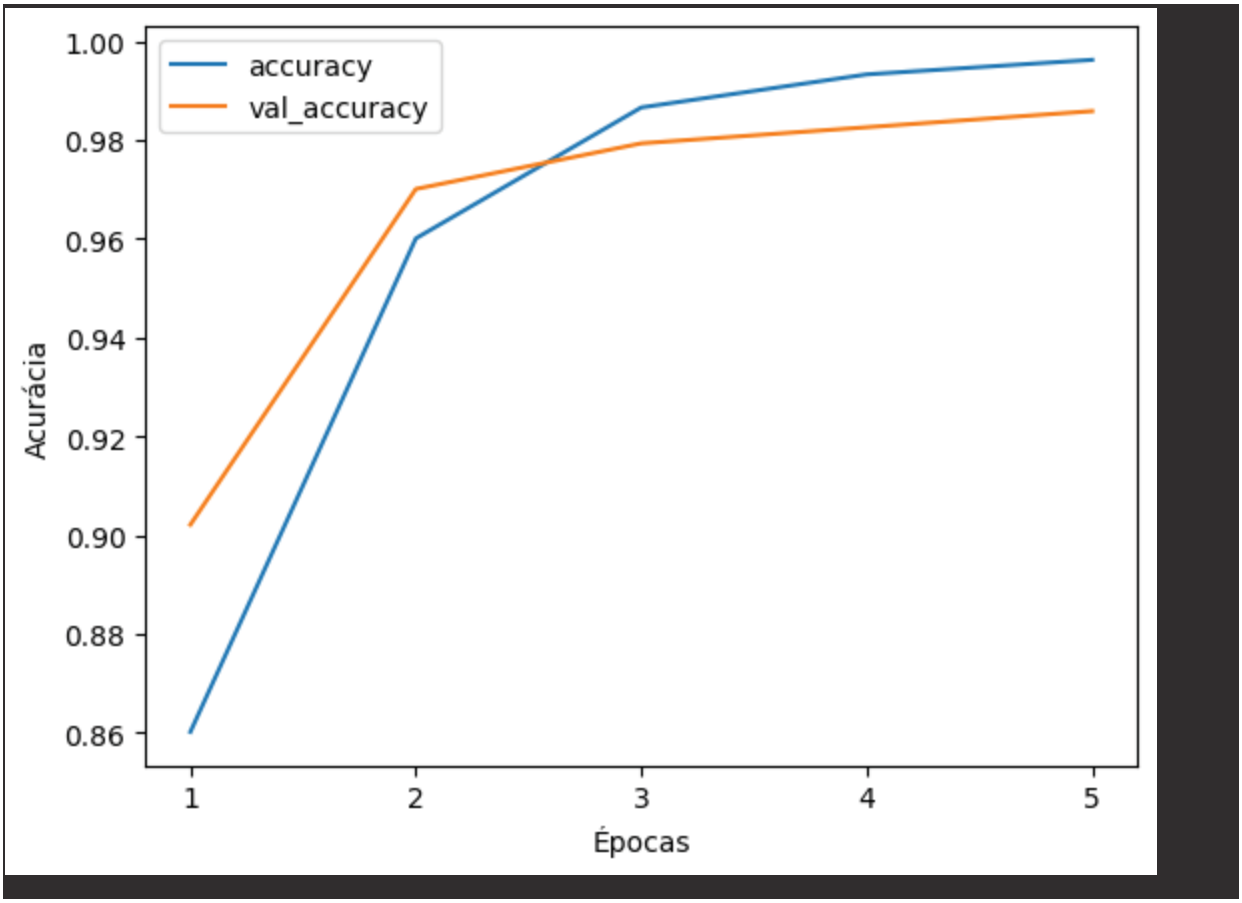
```
Epoch 1/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - accuracy: 0.8219 - loss: 0.5763 - val_accuracy: 0.9021 - val_loss: 0.3058
Epoch 2/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.9463 - loss: 0.2580 - val_accuracy: 0.9701 - val_loss: 0.1802
Epoch 3/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 2s 13ms/step - accuracy: 0.9848 - loss: 0.1446 - val_accuracy: 0.9793 - val_loss: 0.1214
Epoch 4/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.9934 - loss: 0.0889 - val_accuracy: 0.9826 - val_loss: 0.0910
Epoch 5/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - accuracy: 0.9954 - loss: 0.0608 - val_accuracy: 0.9859 - val_loss: 0.0783
```

Plotar função de perda e acurácia

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

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






```
# Predição de um novo texto

# Efetua a predição de um texto novo
#texto = "Hi, my name is Gary, I'm from the local Raccoon Federation, and want to
tell you something."
#"Where's the catnip?"
#"Is your car dirty? Discover our new product. Free for all. Click the link."
texto = "Is your car dirty? Discover our new product. Free for all. Click the
link."
seq_texto = tokenizer.texts_to_sequences([texto]) # Tokeniza
data_texto = pad_sequences(seq_texto, maxlen=T) # Padding
pred = model.predict(data_texto) # Predição
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")

1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
[[0.6220459]]
SPAM
```

3 -

```
Dígitos FAKE baseados na base MNIST.
Configurar o Colab para usar GPU.
Menu Editar | Configurações de Notebook.
Escolher T4 GPU.

◉ Instalação de acessórios para geração dos GIFs

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

```
◉ Importações

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

```
◉ Tratamento da base

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

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

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

# Cria o dataset (from_tensor_slices)
```

```
# Randomiza (shuffle)
# Combina elementos consecutivos em lotes (batch)
train_dataset = tf.data.Dataset.from_tensor_slices(train_images). \
shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
```

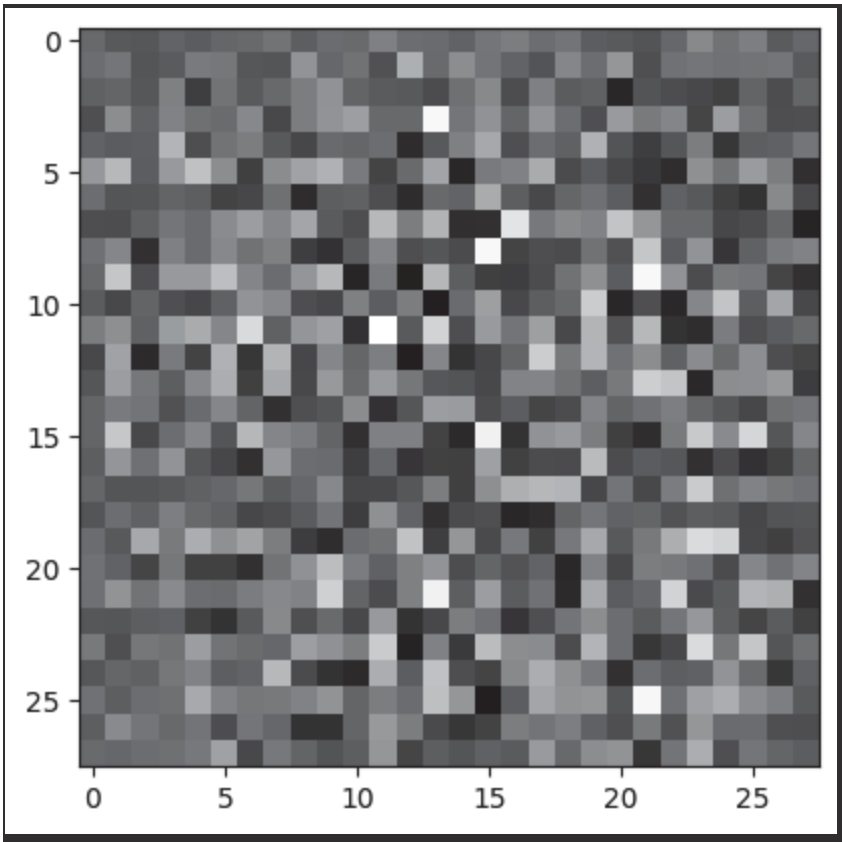
GERADOR

```
# Cria o GERADOR
def make_generator_model():
  model = tf.keras.Sequential()
  model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
  model.add(layers.BatchNormalization())
  model.add(layers.LeakyReLU())
  model.add(layers.Reshape((7, 7, 256)))
  assert model.output_shape == (None, 7, 7, 256) # Note: None is the batch size
      model.add(layers.Conv2DTranspose(128,   (5,   5),   strides=(1,   1),   padding='same',
use_bias=False)) # Fixed indentation
  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)) # Fixed indentation and a typo (''' to ''')
  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')) # Fixed indentation and a typo (''' to ''')
  assert model.output_shape == (None, 28, 28, 1)
  return model
```

Teste do Gerador

```
# Teste do GERADOR, ainda não treinado
generator = make_generator_model()
noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)
plt.imshow(generated_image[0, :, :, 0], cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x79f47847f3a0>
```



DISCRIMINADOR

```
# Cria o DISCRIMINADOR
def make_discriminator_model():
    model = tf.keras.Sequential() # Indent this line and all subsequent lines within the
function
    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 # Indent this line as well
```

Teste do Discriminador

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

```
tf.Tensor([[0.00028078]], shape=(1, 1), dtype=float32)
```

Funções de Perda

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

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

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

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

Checkpoints

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

```
# Configura o Loop de treinamento
EPOCHS = 100
```

```
noise_dim = 100
num_examples_to_generate = 16
# You will reuse this seed overtime (so it's easier)
# to visualize progress in the animated GIF)
seed = tf.random.normal([num_examples_to_generate, noise_dim])
```

```
◉ Passo de Treinamento

# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função
@tf.function
def train_step(images):
    # Indent the code block within the function
    noise = tf.random.normal([BATCH_SIZE, noise_dim])
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)
        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)
        gen_loss = generator_loss(fake_output)
        disc_loss = discriminator_loss(real_output, fake_output)
    gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
                        gradients_of_discriminator      =      disc_tape.gradient(disc_loss,
discriminator.trainable_variables)
                            generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
                                                discriminator.trainable_variables))
```

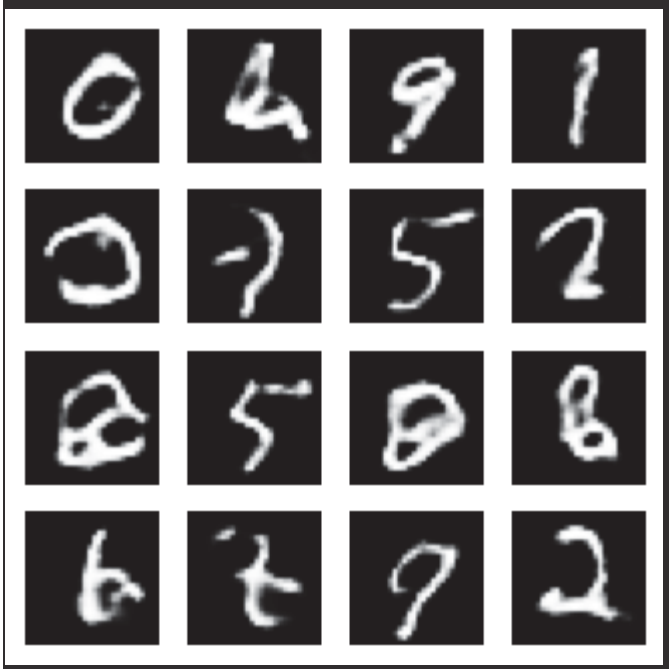
```
◉ Loop de Treinamento

# Treinamento completo/laço
def train(dataset, epochs):
      for epoch in range(epochs): # Indent this line and the following lines within the
function
        start = time.time()
        for image_batch in dataset:
            train_step(image_batch)
        # Produce images for the GIF as you go
        display.clear_output(wait=True)
        generate_and_save_images(generator, epoch + 1, seed)
        # Save the model every 15 epochs
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)
        print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-start))
    # Generate after the final epoch
    display.clear_output(wait=True)
    generate_and_save_images(generator, epochs, seed)
```

Salvamento de imagens

```
# Gerar e salvar imagens
def generate_and_save_images(model, epoch, test_input):
    # Notice `training` is set to False.
    # This is so all layers run in inference mode (batchnorm).
    predictions = model(test_input, training=False)
    fig = plt.figure(figsize=(4, 4))
    for i in range(predictions.shape[0]):
        plt.subplot(4, 4, i+1)
        plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
        plt.axis('off')
    plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
    plt.show()
```

Treinamento

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

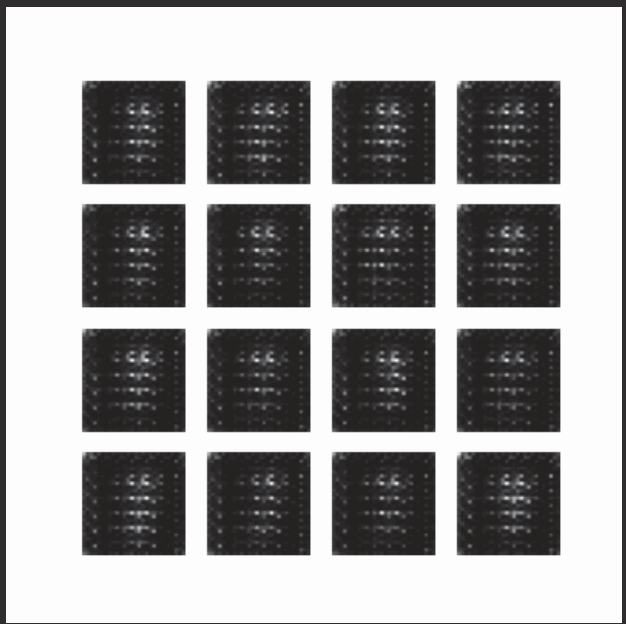


```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at
0x79f4967decb0>
```

Criação do GIF

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

display_image(EPOCHS)

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

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

```
<ipython-input-64-b87c8329cd77>:14: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
  image = imageio.imread(filename)
<ipython-input-64-b87c8329cd77>:16: DeprecationWarning: Starting with ImageIO v3 the behavior of this function will switch to that of iio.v3.imread. To keep the current behavior (and make this warning disappear) use `import imageio.v2 as imageio` or call `imageio.v2.imread` directly.
```



```
  image = imageio.imread(filename)
```

4 -

```
Transformer - Tradutor de Textos
https://www.tensorflow.org/text/tutorials/transformer?hl=pt-br

◉ Instalação e importação dos pacotes

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

import collections
import logging
import os
import pathlib
import re
import string
import sys
import time

import numpy as np
import matplotlib.pyplot as plt

import tensorflow_datasets as tfds
import tensorflow_text as text
import tensorflow as tf
logging.getLogger('tensorflow').setLevel(logging.ERROR) # suppress warnings
```

```
◉ Carregar a base de dados

# Carregar a base de dados

examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en',
with_info=True, as_supervised=True)

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

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

  print()
```

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

```
e quando melhoramos a procura , tiramos a única vantagem da impressão , que é
a serendipidade .
mas e se estes fatores fossem ativos ?
mas eles não tinham a curiosidade de me testar .

and when you improve searchability , you actually take away the one advantage
of print , which is serendipity .
but what if it were active ?
but they did n't test for curiosity .
```

⊙ Tokenização e Destokenização

```
# Tokenização e Destokenização do texto

model_name = "ted_hrlr_translate_pt_en_converter"

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

# Tem 2 tokenizers: um pt outro em en
# tokenizers.en tokeniza e detokeniza

tokenizers = tf.saved_model.load(model_name)
```

⊙ Pipeline de Entrada

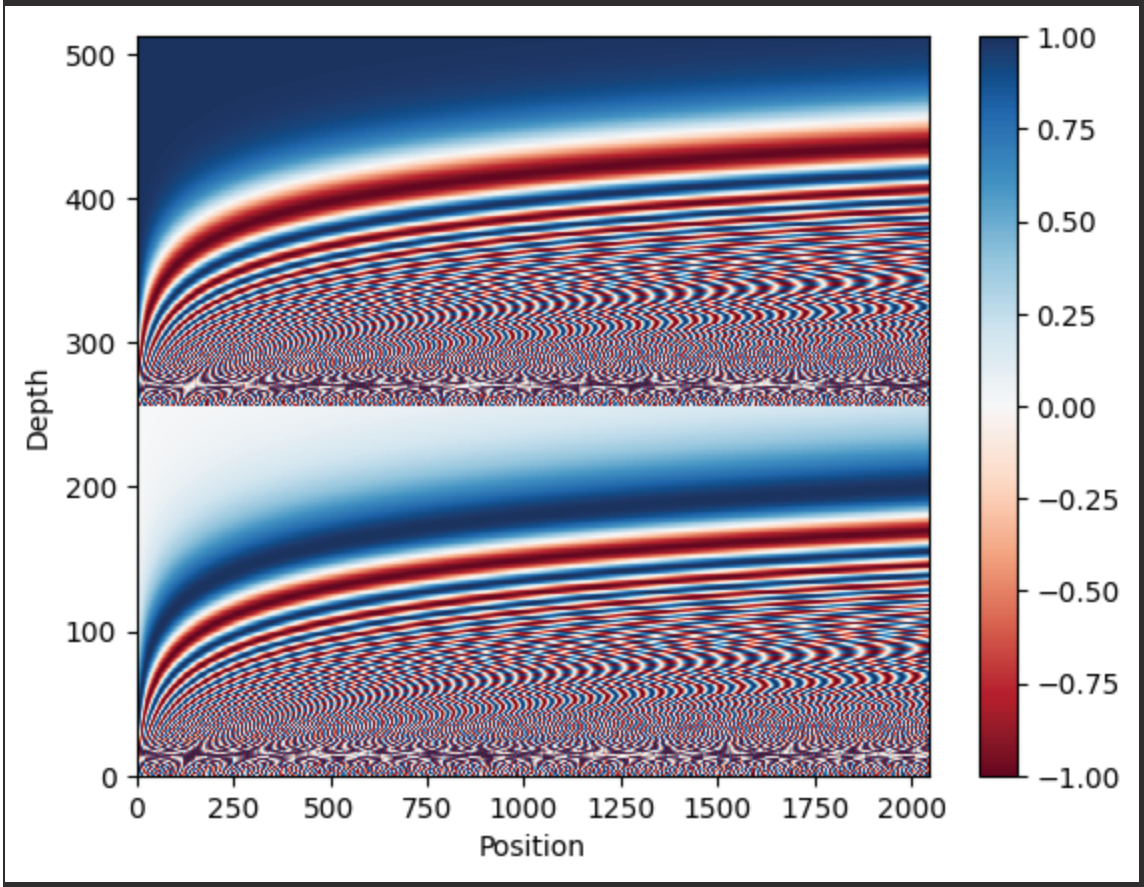
```
# PIPELINE DE ENTRADA
# Codificar/tokenizar lotes de texto puro

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

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

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

```
BUFFER_SIZE = 20000
BATCH_SIZE = 64

# Pipeline simpes: processa, embaralha, agrupa os dados, prefetch
# Datasets de entrada terminam com prefetch

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


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

```
◉ Codificação Posicional

# CODIFICAÇÃO POSICIONAL

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

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

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

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

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


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

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

plt.pcolormesh(pos_encoding, cmap='RdBu')
plt.ylabel('Depth')
plt.xlabel('Position')
plt.colorbar()
plt.show()
```

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



Criar Máscara de 0s e 1s

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

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

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

Função de Atenção

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

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

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

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

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

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

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

  return output, attention_weights
```

Atenção Multi-cabeças

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

    assert d_model % self.num_heads == 0

    self.depth = d_model // self.num_heads
    self.wq = tf.keras.layers.Dense(d_model)
    self.wk = tf.keras.layers.Dense(d_model)
    self.wv = tf.keras.layers.Dense(d_model)
```

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

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


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

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

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

        # 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)

        scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])  # (batch_size, seq_len_q, num_heads,
depth)

        concat_attention = tf.reshape(scaled_attention,
                                      (batch_size, -1, self.d_model))  # (batch_size, seq_len_q, 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')
```

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_states()
  train_accuracy.reset_states()

  # inp -> portuguese, tar -> english
  for (batch, (inp, tar)) in enumerate(train_batches):
    train_step(inp, tar)

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

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

  print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy {train_accuracy.result():.4f}')

  print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')
```

```
◉ 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'WONDERFUL'
```

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

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

Integração de Frameworks e Ferramentas de Segurança Cibernética com Hadoop Hortonworks Sandbox 2.1

Este projeto aborda a implementação de uma aplicação de Big Data que integra diversas ferramentas de segurança cibernética, utilizando o Hadoop Hortonworks Sandbox 2.1. O objetivo principal é processar, armazenar e analisar grandes volumes de dados provenientes de diferentes fontes de segurança cibernética, utilizando ferramentas NoSQL e NewSQL para lidar com a variabilidade e a velocidade dos dados.

Caracterização dos Dados e V’s

As fontes de dados incluem o Kaspersky Security Center, Firewall Checkpoint, Tenable Nessus, AWS WAF e CloudFront, MISP (Malware Information Sharing Platform) e Zabbix. Essas fontes geram alertas e relatórios sobre vulnerabilidades, acessos não autorizados e anomalias de rede. Os dados coletados apresentam grande variedade, pois cada ferramenta gera informações em diferentes formatos, como logs, arquivos CSV e JSON, além de grandes volumes de dados (volume), com alta velocidade de geração (velocidade) e veracidade questionável devido à possibilidade de falsos positivos (veracidade).

Volume: Os dados de firewall, gestão de ameaças e alertas de malware podem gerar terabytes de logs diariamente.

Velocidade: As ferramentas de segurança, especialmente as relacionadas ao firewall e ao sistema de monitoramento de ameaças, geram dados em tempo real.

Variedade: Formatos de dados estruturados (relatórios em CSV, logs) e semiestruturados (JSON) são comuns.

Veracidade: A consistência dos dados depende de políticas de segurança e detecção de falsos positivos.

Valor: A análise correta desses dados permite identificar rapidamente ameaças emergentes, mitigando potenciais ataques.

Integração e Alimentação de Dados

A implementação se inicia com a captura e ingestão dos dados das ferramentas no Sandbox 2.1. A ferramenta de orquestração Apache NiFi pode ser utilizada para automatizar e gerenciar o fluxo de dados. Ela facilita a conexão com APIs das ferramentas e a ingestão contínua dos dados para o sistema Hadoop.

Kaspersky Security Center: Alertas de e-mail podem ser extraídos e transformados em dados legíveis para o Hadoop utilizando NiFi, que se conecta ao servidor de e-mail via protocolo IMAP, extrai as informações relevantes e converte-as em formato JSON para posterior processamento.

Firewall Checkpoint e Tenable Nessus: Logs podem ser coletados diretamente por meio de conectores, como Flume ou NiFi, e enviados ao Hadoop, armazenados no HDFS (HadoopDistributed File System) e indexados pelo Apache HBase (NoSQL) para permitir consultas rápidas.

AWS WAF e CloudFront: Logs de segurança e acesso podem ser extraídos dos serviços AWS usando a API S3 para mover os dados para o Hadoop. Esses logs contém dados críticos sobre ataques de negação de serviço (DoS) e outras ameaças.

MISP: A integração com o MISP (Malware Information Sharing Platform) pode ser estabelecida para capturar dados sobre ameaças emergentes. As informações são transformadas e armazenadas no HDFS para análises preditivas e correlacionadas com outras fontes de dados.

Ferramentas e Modelagem dos Dados

Para gerenciar os dados heterogêneos, diferentes modelos de dados são utilizados:

NoSQL: O Apache HBase é utilizado para armazenar logs de eventos de firewall e alertas de segurança, que são dados de chave-valor. Tal abordagem facilita consultas de alta velocidade sobre logs massivos.

NewSQL: Utiliza-se o Apache Phoenix como camada SQL sobre o HBase para consultas mais complexas, facilitando a análise multidimensional, como a correlação de eventos entre diferentes ferramentas de segurança.

Além disso, o Zabbix é integrado ao Hadoop para monitorar a infraestrutura de rede e servidores, gerando dados sobre o desempenho do sistema e alertas de possíveis vulnerabilidades. A combinação do Zabbix com os dados de segurança permite uma análise mais profunda sobre o estado geral da segurança do ambiente.

Análise e Visualização

Com os dados coletados e armazenados no HDFS e HBase, ferramentas como Apache Spark são usadas para realizar análises em tempo real. As ameaças detectadas podem ser correlacionadas entre as diferentes fontes, identificando padrões ou eventos suspeitos. Para visualização, o Apache Zeppelin é integrado ao Hortonworks Sandbox, permitindo a criação de dashboards que mostram alertas e possíveis correlações entre ataques detectados por diferentes ferramentas.

Conclusão

A implementação mostra como o Sandbox 2.1 e suas ferramentas associadas podem operar a integração de dados de diferentes ferramentas de segurança cibernética. O uso de tecnologias NoSQL e NewSQL permite o armazenamento e análise em tempo real de grandes volumes de dados. A orquestração com Apache NiFi, a persistência com HBase e a análise com Spark oferecem uma solução para monitoramento de segurança em ambientes complexos e heterogêneos.

APÊNDICE 10 – VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

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

B – RESOLUÇÃO

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

1 -

a)

```
Importação de Bibliotecas


# Importando bibliotecas necessárias
import os
import numpy as np
import pandas as pd
import zipfile
import shutil
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report
import cv2
from tensorflow.keras.models import Model # Changed import path
from tensorflow.keras.applications import VGG16 # Changed import path
from tensorflow.keras.preprocessing.image import ImageDataGenerator # Changed import path
from tensorflow.keras.layers import Dense, Flatten # Changed import path
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping # Changed import path
import matplotlib.pyplot as plt
import seaborn as sns

!pip install opencv-python tensorflow numpy
from tensorflow.keras.applications.vgg16 import VGG16
!pip install scikit-image
from skimage.feature import local_binary_pattern
# Instalando livelossplot - Ensure this line is executed in the current session
!pip install livelossplot
# Importing after installation
from livelossplot import PlotLossesKeras # Import after installing the library
#!rm -rf /content/*
!find /content/ -mindepth 1 -maxdepth 1 ! -name 'sample_data' -exec rm -rf {} +
```

```
# Remover a pasta /content/Train_Warwick e seu conteúdo, se existir
!rm -rf /content/Train_Warwick

# Baixando e extraindo os dados de Treino
!gdown 1nh87sPfnrFkZvqcfX_4-tlFQiiHor8Qg
!unzip Train_Warwick.zip -d /content/Train_Warwick


Downloading...
From (original): https://drive.google.com/uc?id=1nh87sPfnrFkZvqcfX_4-tlFQiiHor8Qg
From (redirected): https://drive.google.com/uc?id=1nh87sPfnrFkZvqcfX_4-tlFQiiHor8Qg&confirm=t&uuid=72df37e0-2e84-4f55-822c-86c5e0f5fb91
To: /content/Train_Warwick.zip
100% 69.2M/69.2M [00:01<00:00, 39.9MB/s]
Archive:  Train_Warwick.zip
   creating: /content/Train_Warwick/Train_4cls_amostra/
   creating: /content/Train_Warwick/Train_4cls_amostra/0/
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/46_HER2_61709.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/46_HER2_64186.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/13_HER2_10243.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/46_HER2_58961.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/46_HER2_64180.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/46_HER2_55795.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/18_HER2_25188.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/13_HER2_11145.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/13_HER2_2910.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/01_HER2_4962.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/18_HER2_30785.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/13_HER2_11144.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/01_HER2_4153.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/57_HER2_6110.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/0/01_HER2_7841.png
  ...
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/14_HER2_7219.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/15_HER2_16417.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/14_HER2_7556.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/15_HER2_17849.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/15_HER2_16663.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/15_HER2_16661.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/15_HER2_16897.png
  inflating: /content/Train_Warwick/Train_4cls_amostra/1/14_HER2_8051.png

# Remover a pasta /content/Test_Warwick e seu conteúdo, se existir
!rm -rf /content/Test_Warwick

# Baixando e extraindo os dados de Teste
!gdown 1svpG23vIVWRK43HNdY65Khw-keh-Fs3K
!unzip Test_Warwick.zip -d /content/Test_Warwick

Downloading...
```

```
From (original): https://drive.google.com/uc?id=1svpG23vIVWRK43HNdY65Khw-keh-Fs3K
From (redirected): https://drive.google.com/uc?id=1svpG23vIVWRK43HNdY65Khw-keh-Fs3K&confirm=t&uuid=79af1902-81e1-4cae-ba27-f7c6eb6b4ee7
To: /content/Test_Warwick.zip
100% 42.3M/42.3M [00:01<00:00, 31.0MB/s]
Archive:  Test_Warwick.zip
   creating: /content/Test_Warwick/Test_4cl_amostra/
   creating: /content/Test_Warwick/Test_4cl_amostra/0/
  inflating: /content/Test_Warwick/Test_4cl_amostra/0/70_HER2_23828.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/0/70_HER2_23827.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/0/68_HER2_13324.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/0/73_HER2_7612.png
...
  inflating: /content/Test_Warwick/Test_4cl_amostra/1/34_HER2_11632.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/1/34_HER2_9127.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/1/34_HER2_18120.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/1/34_HER2_17287.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/1/34_HER2_18288.png
  inflating: /content/Test_Warwick/Test_4cl_amostra/1/34_HER2_6302.png
```

b)

```python
import os
import shutil
import json
from pathlib import Path

# Função para criar os diretórios /80 e /20 dentro de Train_4cls_amostra
def create_dirs(base_path):
    for dir_name in ['80', '20']:  # Diretórios 80 e 20
        for i in range(4):  # Subdiretórios 0, 1, 2, 3 dentro de 80 e 20
            os.makedirs(base_path / dir_name / str(i), exist_ok=True)

# Função para mover as imagens do primeiro paciente de cada diretório para /20 e o resto para /80
def split_images(base_path):
    for i in range(4):  # Diretórios 0, 1, 2, 3
        img_dir = base_path / str(i)
        images = sorted([img for img in os.listdir(img_dir) if img.endswith('.png')])

        if images:  # Verifica se há imagens no diretório
            # Pega as imagens do primeiro paciente
            first_patient = images[0].split('_')[0]
            for img in images:
                if img.startswith(first_patient):
                    shutil.move(img_dir / img, base_path / '20' / str(i) / img)
                else:
                    shutil.move(img_dir / img, base_path / '80' / str(i) / img)
```

```python
# Função para contar as imagens nos diretórios /80 e /20
def count_images(base_path):
    counts = {}
    for i in range(4):  # Diretórios 0, 1, 2, 3
        count_80 = len(os.listdir(base_path / '80' / str(i)))
        count_20 = len(os.listdir(base_path / '20' / str(i)))
        counts[i] = {"80": count_80, "20": count_20}
    return counts

# Função para gerar os arquivos de metadados
def create_meta_files(base_path, counts, meta_dir):
    os.makedirs(meta_dir, exist_ok=True)

    # Cria o arquivo classes.txt
    with open(meta_dir / 'classes.txt', 'w') as f:
        f.write('\n'.join([str(i) for i in range(4)]))

    # Cria o arquivo labels.txt
    with open(meta_dir / 'labels.txt', 'w') as f:
        f.write('\n'.join([str(i) for i in range(4)]))

    # Cria o arquivo totais.txt
    with open(meta_dir / 'totais.txt', 'w') as f:
        for i in range(4):
            f.write(f"Classe {i}: 80 -> {counts[i]['80']} imagens, 20 -> {counts[i]['20']} imagens\n")

# Função para listar arquivos e gerar arquivos .txt e .json corrigido
def list_files_and_generate_metadata(base_path, split, meta_dir):
    files_train = []
    files_train_80 = []
    files_train_20 = []
    files_test = []

    train_json = {str(i): [] for i in range(4)}
    train_80_json = {str(i): [] for i in range(4)}
    train_20_json = {str(i): [] for i in range(4)}
    test_json = {str(i): [] for i in range(4)}

    # Listar arquivos de treino e separá-los entre /80 e /20
    for i in range(4):
        for root, dirs, files in os.walk(base_path / f"Train_Warwick/Train_4cls_amostra/{i}"):
            for file in files:
                if file.endswith('.png'):
                    rel_path = os.path.join(str(i), file)
                    files_train.append(f"{i}/{file}")

                    if '80' in root:
                        files_train_80.append(f"{i}/{file}")
                        train_80_json[str(i)].append(rel_path)
                    elif '20' in root:
                        files_train_20.append(f"{i}/{file}")
                        train_20_json[str(i)].append(rel_path)
```

```
    # Listar arquivos de teste
    for i in range(4):
        for root, dirs, files in os.walk(base_path / f"Test_Warwick/Test_4cl_amostra/{i}"):
            for file in files:
                if file.endswith('.png'):
                    rel_path = os.path.join(str(i), file)
                    files_test.append(f"{i}/{file}")
                    test_json[str(i)].append(rel_path)

    # Escrever arquivos .txt
    with open(meta_dir / 'train.txt', 'w') as f:
        f.write('\n'.join(files_train))

    with open(meta_dir / 'train_80.txt', 'w') as f:
        f.write('\n'.join(files_train_80))

    with open(meta_dir / 'train_20.txt', 'w') as f:
        f.write('\n'.join(files_train_20))

    with open(meta_dir / 'test.txt', 'w') as f:
        f.write('\n'.join(files_test))

    # Escrever arquivos .json
    with open(meta_dir / 'train.json', 'w') as f:
        json.dump(train_json, f, indent=4)

    with open(meta_dir / 'train_80.json', 'w') as f:
        json.dump(train_80_json, f, indent=4)

    with open(meta_dir / 'train_20.json', 'w') as f:
        json.dump(train_20_json, f, indent=4)

    with open(meta_dir / 'test.json', 'w') as f:
        json.dump(test_json, f, indent=4)

# Caminhos base
train_path = Path('/content/Train_Warwick/Train_4cls_amostra')
test_path = Path('/content/Test_Warwick/Test_4cl_amostra')
meta_path = Path('/content/meta')

# Criar diretórios
create_dirs(train_path)
create_dirs(test_path)

# Mover as imagens
split_images(train_path)
split_images(test_path)

# Contar as imagens e gerar os arquivos de metadados
counts = count_images(train_path)
create_meta_files(train_path, counts, meta_path)
```

```
# Gerar listagens e arquivos .json corrigidos
list_files_and_generate_metadata(train_path, split="80", meta_dir=meta_path)
list_files_and_generate_metadata(test_path, split="20", meta_dir=meta_path)

# Remover as pastas vazias numeradas de 0 a 3 dentro de /content/Train_Warwick/Train_4cls_amostra/
for i in range(4):
    !rm -rf /content/Train_Warwick/Train_4cls_amostra/{i}

for i in range(4):
    !rm -rf /content/Test_Warwick/Test_4cl_amostra/{i}
```

c)

```
!pip install opencv-python-headless tensorflow pandas scikit-image

import os
import numpy as np
import pandas as pd
import cv2
from skimage.feature import local_binary_pattern
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from pathlib import Path
from tensorflow.keras.models import Model

# Função para extrair características usando LBP
def extract_lbp_features(img_path):
    img = cv2.imread(str(img_path), cv2.IMREAD_GRAYSCALE)  # Ler a imagem em escala de cinza
    img = cv2.resize(img, (128, 128))  # Redimensionar a imagem
    lbp = local_binary_pattern(img, P=8, R=1, method='uniform')  # Calcular LBP
    hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, 11), density=True)  # Histograma normalizado
    return hist
# Carregar o modelo VGG16 fora da função para evitar recarga repetida
base_model = VGG16(weights='imagenet', include_top=True)
vgg16_model = Model(inputs=base_model.input, outputs=base_model.layers[-2].output)

# Função para extrair características usando VGG16 (modificada)
def extract_vgg16_features(img_path):
    img = image.load_img(img_path, target_size=(224, 224))  # Redimensionar a imagem para 224x224
    img_array = image.img_to_array(img)  # Converter a imagem em array
    img_array = np.expand_dims(img_array, axis=0)  # Adicionar dimensão extra
    img_array = preprocess_input(img_array)  # Pré-processar a imagem
    features = vgg16_model.predict(img_array)  # Extrair características
    return features.flatten()  # Retornar como vetor 1D
```

```python
# Função para processar as imagens nos diretórios /80 e /20 e salvar os CSVs separados e unificados
def process_images(base_path, output_dir):
    subsets = ['80', '20']
    lbp_combined_data = []
    vgg16_combined_data = []

    for subset in subsets:
        lbp_data = []
        vgg16_data = []
        for i in range(4):  # Para cada classe (0 a 3)
            class_path = base_path / subset / str(i)  # Caminho da classe
            if class_path.exists():  # Verificar se o diretório existe
                for img_name in os.listdir(class_path):
                    if img_name.endswith('.png'):  # Filtrar apenas imagens .png
                        img_path = class_path / img_name

                        # Extrair características
                        lbp_features = extract_lbp_features(img_path)
                        vgg16_features = extract_vgg16_features(img_path)

                        # Adicionar dados à lista
                        lbp_data.append([i] + list(lbp_features))
                        vgg16_data.append([i] + list(vgg16_features))

                        # Adicionar aos dados combinados (80 + 20)
                        lbp_combined_data.append([i] + list(lbp_features))
                        vgg16_combined_data.append([i] + list(vgg16_features))

        # Criar DataFrames e salvar como CSV para cada extrator e conjunto (80 ou 20)
        lbp_df = pd.DataFrame(lbp_data)
        lbp_csv_path = output_dir / f'lbp_features_train_{subset}.csv'
        lbp_df.to_csv(lbp_csv_path, index=False, header=False)

        vgg16_df = pd.DataFrame(vgg16_data)
        vgg16_csv_path = output_dir / f'vgg16_features_train_{subset}.csv'
        vgg16_df.to_csv(vgg16_csv_path, index=False, header=False)

    # Criar DataFrames combinados e salvar como CSV unificado (80 + 20)
    lbp_combined_df = pd.DataFrame(lbp_combined_data)
    lbp_combined_csv_path = output_dir / 'lbp_features_train_combined.csv'
    lbp_combined_df.to_csv(lbp_combined_csv_path, index=False, header=False)

    vgg16_combined_df = pd.DataFrame(vgg16_combined_data)
    vgg16_combined_csv_path = output_dir / 'vgg16_features_train_combined.csv'
    vgg16_combined_df.to_csv(vgg16_combined_csv_path, index=False, header=False)

# Caminhos base
train_path = Path('/content/Train_Warwick/Train_4cls_amostra')
output_path = Path('/content/output_features')
output_path.mkdir(parents=True, exist_ok=True)  # Criar diretório de saída, se não existir

# Processar imagens e extrair características
```

```
process_images(train_path, output_path)

print("Extração de características concluída!")

Downloading data from
https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 6s 0us/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 7s 7s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 83ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 132ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 110ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 92ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 109ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 199ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Extração de características concluída!
```

d)

```
!pip install scikit-learn tensorflow

import pandas as pd
import numpy as np
import joblib
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from pathlib import Path

# Função para carregar os dados do CSV
def load_data(file_path):
    data = pd.read_csv(file_path, header=None)
    X = data.iloc[:, 1:].values  # Características
    y = data.iloc[:, 0].values   # Classes
    return X, y
```

```
# Função para carregar dados de treino e validação
def load_train_val_data(train_file_path, val_file_path):
    X_train, y_train = load_data(train_file_path)
    X_val, y_val = load_data(val_file_path)
    return X_train, X_val, y_train, y_val

# Função para treinar, avaliar e salvar um modelo
def train_and_evaluate_model(X_train, X_val, y_train, y_val, model_name, scaler_filename, model_filename):
    # Normalizar os dados
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_val = scaler.transform(X_val)

    # Salvar o scaler
    joblib.dump(scaler, f'/content/{scaler_filename}.pkl')

    # Definir o modelo com base no nome
    if model_name == 'Random Forest':
        model = RandomForestClassifier(n_estimators=100, random_state=42)
        model.fit(X_train, y_train)  # Treinar modelo Random Forest

        # Salvar o modelo Random Forest
        joblib.dump(model, f'/content/{model_filename}.pkl')

    elif model_name == 'SVM':
        model = SVC(kernel='linear', random_state=42)
        model.fit(X_train, y_train)  # Treinar modelo SVM

        # Salvar o modelo SVM
        joblib.dump(model, f'/content/{model_filename}.pkl')

    elif model_name == 'RNA':

        model = Sequential()
        model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
        model.add(Dense(64, activation='relu'))
        model.add(Dense(4, activation='softmax'))  # 4 classes

        model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
        model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0)

        # Salvar o modelo RNA
        model.save(f'/content/{model_filename}.h5')

        # Avaliação do modelo RNA
        y_pred = model.predict(X_val)
        y_pred = np.argmax(y_pred, axis=1)  # Converter probabilidades em classes
    else:
        raise ValueError("Modelo não suportado.")

    if model_name != 'RNA':
        y_pred = model.predict(X_val)
```

```
    # Avaliar o modelo
    accuracy = accuracy_score(y_val, y_pred)
    report = classification_report(y_val, y_pred)
    print(f"Modelo: {model_name}")
    print(f"Acurácia: {accuracy:.4f}")
    print(f"Relatório de Classificação:\n{report}")

# Caminhos base
output_path = Path('/content/output_features')
train_lbp_file_path = output_path / 'lbp_features_train_80.csv'
val_lbp_file_path = output_path / 'lbp_features_train_20.csv'
train_vgg16_file_path = output_path / 'vgg16_features_train_80.csv'
val_vgg16_file_path = output_path / 'vgg16_features_train_20.csv'

# Carregar dados LBP e VGG16
X_train_lbp, X_val_lbp, y_train_lbp, y_val_lbp = load_train_val_data(train_lbp_file_path, val_lbp_file_path)
X_train_vgg16, X_val_vgg16, y_train_vgg16, y_val_vgg16 = load_train_val_data(train_vgg16_file_path,
val_vgg16_file_path)

# Treinar e avaliar modelos usando características LBP
print("Resultados para LBP:")
train_and_evaluate_model(X_train_lbp, X_val_lbp, y_train_lbp, y_val_lbp, 'Random Forest', 'scaler_lbp_rf',
'Random_Forest_lbp_model')
train_and_evaluate_model(X_train_lbp, X_val_lbp, y_train_lbp, y_val_lbp, 'SVM', 'scaler_lbp_svm',
'SVM_lbp_model')
train_and_evaluate_model(X_train_lbp, X_val_lbp, y_train_lbp, y_val_lbp, 'RNA', None, 'RNA_lbp_model')

# Treinar e avaliar modelos usando características VGG16
print("Resultados para VGG16:")
train_and_evaluate_model(X_train_vgg16, X_val_vgg16, y_train_vgg16, y_val_vgg16, 'Random Forest',
'scaler_vgg16_rf', 'Random_Forest_vgg16_model')


train_and_evaluate_model(X_train_vgg16, X_val_vgg16, y_train_vgg16, y_val_vgg16, 'SVM', 'scaler_vgg16_svm',
'SVM_vgg16_model')
train_and_evaluate_model(X_train_vgg16, X_val_vgg16, y_train_vgg16, y_val_vgg16, 'RNA', None,
'RNA_vgg16_model')


Resultados para LBP:
Modelo: Random Forest
Acurácia: 0.8103
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.87      0.69      0.77        29
           1       0.70      0.70      0.70        27
           2       0.79      0.87      0.83        30
           3       0.88      0.97      0.92        30

    accuracy                           0.81       116
   macro avg       0.81      0.81      0.80       116
```

```
weighted avg       0.81      0.81      0.81       116

Modelo: SVM
Acurácia: 0.7414
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.88      0.79      0.84        29
           1       0.67      0.22      0.33        27
           2       0.56      0.93      0.70        30
           3       0.94      0.97      0.95        30

    accuracy                           0.74       116
   macro avg       0.76      0.73      0.71       116
weighted avg       0.76      0.74      0.71       116

/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
4/4 ━━━━━━━━━━━━━━━━━━━━ 2s 370ms/step
Modelo: RNA
Acurácia: 0.8276
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.91      0.72      0.81        29
           1       0.70      0.59      0.64        27
           2       0.72      0.97      0.83        30
           3       1.00      1.00      1.00        30

    accuracy                           0.83       116
   macro avg       0.83      0.82      0.82       116
weighted avg       0.84      0.83      0.82       116

Resultados para VGG16:
Modelo: Random Forest
Acurácia: 0.6983
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.58      0.86      0.69        29
           1       0.56      0.19      0.28        27
           2       0.70      1.00      0.82        30
           3       1.00      0.70      0.82        30

    accuracy                           0.70       116
   macro avg       0.71      0.69      0.65       116
weighted avg       0.71      0.70      0.66       116
```

```
Modelo: SVM
Acurácia: 0.7069
Relatório de Classificação:
                precision    recall  f1-score   support

           0       0.62      0.97      0.76        29
           1       0.92      0.41      0.56        27
           2       0.63      0.90      0.74        30
           3       1.00      0.53      0.70        30

    accuracy                           0.71       116
   macro avg       0.79      0.70      0.69       116
weighted avg       0.79      0.71      0.69       116

/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
1/4 ━━━━━━━━━━━━━━━━━━━━ 0s 171ms/stepWARNING:tensorflow:5 out of the last 601 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x7c6197d38670> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
Modelo: RNA
Acurácia: 0.6466
Relatório de Classificação:
                precision    recall  f1-score   support

           0       0.63      0.90      0.74        29
           1       0.47      0.30      0.36        27
           2       0.60      0.80      0.69        30
           3       0.94      0.57      0.71        30

    accuracy                           0.65       116
   macro avg       0.66      0.64      0.63       116
weighted avg       0.67      0.65      0.63       116
```

e)

```
import os
import numpy as np
import pandas as pd
import cv2
```

```python
from skimage.feature import local_binary_pattern
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import preprocess_input
from pathlib import Path
from tensorflow.keras.models import Model

# Função para extrair características usando LBP
def extract_lbp_features(img_path):
    img = cv2.imread(str(img_path), cv2.IMREAD_GRAYSCALE)  # Ler a imagem em escala de cinza
    img = cv2.resize(img, (128, 128))  # Redimensionar a imagem
    lbp = local_binary_pattern(img, P=8, R=1, method='uniform')  # Calcular LBP
    hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, 11), density=True)  # Histograma normalizado
    return hist

# Carregar o modelo VGG16 fora da função para evitar recarga repetida
base_model = VGG16(weights='imagenet', include_top=True)
vgg16_model = Model(inputs=base_model.input, outputs=base_model.layers[-2].output)

# Função para extrair características usando VGG16 (modificada)
def extract_vgg16_features(img_path):
    img = image.load_img(img_path, target_size=(224, 224))  # Redimensionar a imagem para 224x224
    img_array = image.img_to_array(img)  # Converter a imagem em array
    img_array = np.expand_dims(img_array, axis=0)  # Adicionar dimensão extra
    img_array = preprocess_input(img_array)  # Pré-processar a imagem
    features = vgg16_model.predict(img_array)  # Extrair características
    return features.flatten()  # Retornar como vetor 1D

# Função para processar as imagens nos diretórios /80 e /20 e salvar os CSVs separados e unificados
def process_images(base_path, output_dir):
    subsets = ['80', '20']
    lbp_combined_data = []
    vgg16_combined_data = []

    for subset in subsets:
        lbp_data = []
        vgg16_data = []
        for i in range(4):  # Para cada classe (0 a 3)
            class_path = base_path / subset / str(i)  # Caminho da classe
            if class_path.exists():  # Verificar se o diretório existe
                for img_name in os.listdir(class_path):
                    if img_name.endswith('.png'):  # Filtrar apenas imagens .png
                        img_path = class_path / img_name

                        # Extrair características
                        lbp_features = extract_lbp_features(img_path)
                        vgg16_features = extract_vgg16_features(img_path)

                        # Adicionar dados à lista
                        lbp_data.append([i] + list(lbp_features))
                        vgg16_data.append([i] + list(vgg16_features))
```

```
                            # Adicionar aos dados combinados (80 + 20)
                            lbp_combined_data.append([i] + list(lbp_features))
                            vgg16_combined_data.append([i] + list(vgg16_features))

        # Criar DataFrames e salvar como CSV para cada extrator e conjunto (80 ou 20)
        lbp_df = pd.DataFrame(lbp_data)
        lbp_csv_path = output_dir / f'lbp_features_test_{subset}.csv'
        lbp_df.to_csv(lbp_csv_path, index=False, header=False)

        vgg16_df = pd.DataFrame(vgg16_data)
        vgg16_csv_path = output_dir / f'vgg16_features_test_{subset}.csv'
        vgg16_df.to_csv(vgg16_csv_path, index=False, header=False)

    # Criar DataFrames combinados e salvar como CSV unificado (80 + 20)
    lbp_combined_df = pd.DataFrame(lbp_combined_data)
    lbp_combined_csv_path = output_dir / 'lbp_features_test_combined.csv'
    lbp_combined_df.to_csv(lbp_combined_csv_path, index=False, header=False)

    vgg16_combined_df = pd.DataFrame(vgg16_combined_data)
    vgg16_combined_csv_path = output_dir / 'vgg16_features_test_combined.csv'
    vgg16_combined_df.to_csv(vgg16_combined_csv_path, index=False, header=False)

# Caminhos base
test_path = Path('/content/Test_Warwick/Test_4cl_amostra')
output_path = Path('/content/output_features')
output_path.mkdir(parents=True, exist_ok=True)  # Criar diretório de saída, se não existir

# Processar imagens e extrair características
process_images(train_path, output_path)

print("Extração de características concluída!")

WARNING:tensorflow:6 out of the last 602 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x7c6197a5c310> triggered
tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating
@tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of
tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has
reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 504ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
...
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
Extração de características concluída!
```

f)

```
import pandas as pd
import numpy as np
import joblib
from sklearn.metrics import classification_report, accuracy_score
from tensorflow.keras.models import load_model
from pathlib import Path

# Função para carregar os dados de teste do CSV
def load_test_data(file_path):
    data = pd.read_csv(file_path, header=None)
    X = data.iloc[:, 1:].values  # Características
    y = data.iloc[:, 0].values   # Classes
    return X, y

# Função para carregar e avaliar o modelo nos dados de teste
def evaluate_model(X_test, y_test, model_name, model_file, scaler_file=None):
    # Normalizar os dados se for um modelo de Machine Learning (Random Forest ou SVM)
    if scaler_file:
        scaler = joblib.load(scaler_file)
        X_test = scaler.transform(X_test)
    # Carregar o modelo
    if model_name in ['Random Forest', 'SVM']:
        model = joblib.load(model_file)
        y_pred = model.predict(X_test)
    elif model_name == 'RNA':
        model = load_model(model_file)
        y_pred = model.predict(X_test)
        y_pred = np.argmax(y_pred, axis=1)  # Converter probabilidades em classes
    else:
        raise ValueError("Modelo não suportado.")
    # Avaliar o modelo
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    print(f"Modelo: {model_name}")
    print(f"Acurácia: {accuracy:.4f}")
    print(f"Relatório de Classificação:\n{report}")
# Caminhos dos arquivos gerados na tarefa 5 (corrigidos)
test_lbp_file_path = Path('/content/output_features/lbp_features_test_combined.csv')
test_vgg16_file_path = Path('/content/output_features/vgg16_features_test_combined.csv')
```

```
# Caminhos dos modelos treinados (Random Forest e SVM separados para LBP e VGG16)
rf_lbp_model_path = Path('/content/Random_Forest_lbp_model.pkl')
svm_lbp_model_path = Path('/content/SVM_lbp_model.pkl')
rf_vgg16_model_path = Path('/content/Random_Forest_vgg16_model.pkl')
svm_vgg16_model_path = Path('/content/SVM_vgg16_model.pkl')

# Caminhos para os modelos RNA separados
rna_lbp_model_path = Path('/content/RNA_lbp_model.h5')  # RNA para LBP
rna_vgg16_model_path = Path('/content/RNA_vgg16_model.h5')  # RNA para VGG16

# Caminhos dos scalers usados para normalizar os dados de treino
scaler_lbp_rf_path = Path('/content/scaler_lbp_rf.pkl')
scaler_vgg16_rf_svm_path = Path('/content/scaler_vgg16_rf.pkl')

# Carregar dados de teste LBP e VGG16
X_test_lbp, y_test_lbp = load_test_data(test_lbp_file_path)
X_test_vgg16, y_test_vgg16 = load_test_data(test_vgg16_file_path)

# Avaliar modelos nos dados de teste usando LBP
print("Resultados para LBP:")
evaluate_model(X_test_lbp, y_test_lbp, 'Random Forest', rf_lbp_model_path, scaler_lbp_rf_path)
evaluate_model(X_test_lbp, y_test_lbp, 'SVM', svm_lbp_model_path, scaler_lbp_rf_path)
evaluate_model(X_test_lbp, y_test_lbp, 'RNA', rna_lbp_model_path)  # Usando o modelo correto para LBP

# Avaliar modelos nos dados de teste usando VGG16
print("\nResultados para VGG16:")
evaluate_model(X_test_vgg16, y_test_vgg16, 'Random Forest', rf_vgg16_model_path, scaler_vgg16_rf_svm_path)
evaluate_model(X_test_vgg16, y_test_vgg16, 'SVM', svm_vgg16_model_path, scaler_vgg16_rf_svm_path)
evaluate_model(X_test_vgg16, y_test_vgg16, 'RNA', rna_vgg16_model_path)  # Usando o modelo correto para VGG16

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Resultados para LBP:
Modelo: Random Forest
Acurácia: 0.9629
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       146
           1       0.95      0.95      0.95       147
           2       0.95      0.97      0.96       150
           3       0.97      0.99      0.98       150

    accuracy                           0.96       593
   macro avg       0.96      0.96      0.96       593
weighted avg       0.96      0.96      0.96       593

Modelo: SVM
Acurácia: 0.7926
Relatório de Classificação:
              precision    recall  f1-score   support
```

```
           0       0.83      0.80      0.82       146
           1       0.66      0.62      0.64       147
           2       0.72      0.81      0.76       150
           3       0.97      0.94      0.95       150

    accuracy                           0.79       593
   macro avg       0.79      0.79      0.79       593
weighted avg       0.79      0.79      0.79       593

19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
Modelo: RNA
Acurácia: 0.2530
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       146
           1       0.00      0.00      0.00       147
           2       0.25      1.00      0.40       150
           3       0.00      0.00      0.00       150

    accuracy                           0.25       593
   macro avg       0.06      0.25      0.10       593
weighted avg       0.06      0.25      0.10       593


Resultados para VGG16:
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: 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))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: 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))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: 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))
Modelo: Random Forest
Acurácia: 0.9410
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.89      0.97      0.93       146
           1       0.97      0.85      0.91       147
           2       0.92      1.00      0.96       150
           3       1.00      0.94      0.97       150

    accuracy                           0.94       593
   macro avg       0.94      0.94      0.94       593
weighted avg       0.94      0.94      0.94       593
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Modelo: SVM
Acurácia: 0.9427
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94       146
           1       0.99      0.89      0.94       147
           2       0.90      0.98      0.94       150
           3       1.00      0.91      0.95       150

    accuracy                           0.94       593
   macro avg       0.95      0.94      0.94       593
weighted avg       0.95      0.94      0.94       593

19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
Modelo: RNA
Acurácia: 0.9309
Relatório de Classificação:
              precision    recall  f1-score   support

           0       0.86      0.98      0.92       146
           1       0.92      0.84      0.88       147
           2       0.96      0.94      0.95       150
           3       0.99      0.97      0.98       150

    accuracy                           0.93       593
   macro avg       0.93      0.93      0.93       593
weighted avg       0.93      0.93      0.93       593
```

g)

```python
import pandas as pd
import numpy as np
import joblib
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score
from tensorflow.keras.models import load_model
from pathlib import Path

# Função para carregar os dados de teste do CSV
def load_test_data(file_path):
    data = pd.read_csv(file_path, header=None)
    X = data.iloc[:, 1:].values  # Características
    y = data.iloc[:, 0].values    # Classes
    return X, y
# Função para calcular e exibir métricas a partir da matriz de confusão
def calculate_metrics(y_true, y_pred):
    # Calcular matriz de confusão
```

```
    conf_matrix = confusion_matrix(y_true, y_pred)
    print(f"Matriz de Confusão:\n{conf_matrix}")

    # Calcular sensibilidade (recall)
    sensitivity = recall_score(y_true, y_pred, average='weighted')  # Sensibilidade média ponderada
    print(f"Sensibilidade (Recall): {sensitivity:.4f}")

    # Calcular especificidade manualmente
    specificity = []
    for i in range(len(conf_matrix)):
        true_negatives = conf_matrix.sum() - (conf_matrix[i, :].sum() + conf_matrix[:, i].sum() -
conf_matrix[i, i])
        false_positives = conf_matrix[:, i].sum() - conf_matrix[i, i]
        specificity.append(true_negatives / (true_negatives + false_positives))
    specificity = np.mean(specificity)  # Média das especificidades
    print(f"Especificidade: {specificity:.4f}")

    # Calcular F1-Score
    f1 = f1_score(y_true, y_pred, average='weighted')  # F1-Score ponderado
    print(f"F1-Score: {f1:.4f}")

# Função para carregar e avaliar o modelo nos dados de teste
def evaluate_model(X_test, y_test, model_name, model_file, scaler_file=None):
    # Normalizar os dados se for um modelo de Machine Learning (Random Forest ou SVM)
    if scaler_file:
        scaler = joblib.load(scaler_file)
        X_test = scaler.transform(X_test)

    # Carregar o modelo
    if model_name in ['Random Forest', 'SVM']:
        model = joblib.load(model_file)
        y_pred = model.predict(X_test)
    elif model_name == 'RNA':
        model = load_model(model_file)
        y_pred = model.predict(X_test)
        y_pred = np.argmax(y_pred, axis=1)  # Converter probabilidades em classes
    else:
        raise ValueError("Modelo não suportado.")

    # Avaliar o modelo
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Modelo: {model_name}")
    print(f"Acurácia: {accuracy:.4f}")

    # Calcular métricas
    calculate_metrics(y_test, y_pred)

# Caminhos dos arquivos gerados na tarefa 5 (corrigidos)
test_lbp_file_path = Path('/content/output_features/lbp_features_test_combined.csv')
test_vgg16_file_path = Path('/content/output_features/vgg16_features_test_combined.csv')

# Caminhos dos modelos treinados (Random Forest e SVM separados para LBP e VGG16)
```

```
rf_lbp_model_path = Path('/content/Random_Forest_lbp_model.pkl')
svm_lbp_model_path = Path('/content/SVM_lbp_model.pkl')
rf_vgg16_model_path = Path('/content/Random_Forest_vgg16_model.pkl')
svm_vgg16_model_path = Path('/content/SVM_vgg16_model.pkl')

# Caminhos para os modelos RNA separados
rna_lbp_model_path = Path('/content/RNA_lbp_model.h5')  # RNA para LBP
rna_vgg16_model_path = Path('/content/RNA_vgg16_model.h5')  # RNA para VGG16

# Caminhos dos scalers usados para normalizar os dados de treino
scaler_lbp_rf_path = Path('/content/scaler_lbp_rf.pkl')
scaler_vgg16_rf_svm_path = Path('/content/scaler_vgg16_rf.pkl')

# Carregar dados de teste LBP e VGG16
X_test_lbp, y_test_lbp = load_test_data(test_lbp_file_path)
X_test_vgg16, y_test_vgg16 = load_test_data(test_vgg16_file_path)

# Avaliar modelos nos dados de teste usando LBP
print("Resultados para LBP:")
evaluate_model(X_test_lbp, y_test_lbp, 'Random Forest', rf_lbp_model_path, scaler_lbp_rf_path)
evaluate_model(X_test_lbp, y_test_lbp, 'SVM', svm_lbp_model_path, scaler_lbp_rf_path)
evaluate_model(X_test_lbp, y_test_lbp, 'RNA', rna_lbp_model_path)

# Avaliar modelos nos dados de teste usando VGG16
print("\nResultados para VGG16:")
evaluate_model(X_test_vgg16, y_test_vgg16, 'Random Forest', rf_vgg16_model_path, scaler_vgg16_rf_svm_path)
evaluate_model(X_test_vgg16, y_test_vgg16, 'SVM', svm_vgg16_model_path, scaler_vgg16_rf_svm_path)
evaluate_model(X_test_vgg16, y_test_vgg16, 'RNA', rna_vgg16_model_path)

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will
be empty until you train or evaluate the model.
Resultados para LBP:
Modelo: Random Forest
Acurácia: 0.9629
Matriz de Confusão:
[[137   7   1   1]
 [  2 139   6   0]
 [  1   0 146   3]
 [  0   1   0 149]]
Sensibilidade (Recall): 0.9629
Especificidade: 0.9876
F1-Score: 0.9628
Modelo: SVM
Acurácia: 0.7926
Matriz de Confusão:
[[117  24   3   2]
 [ 18  91  38   0]
 [  5  21 121   3]
 [  1   2   6 141]]
Sensibilidade (Recall): 0.7926
Especificidade: 0.9309
F1-Score: 0.7925
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```

```
Modelo: RNA
Acurácia: 0.2530
Matriz de Confusão:
[[  0   0 146   0]
 [  0   0 147   0]
 [  0   0 150   0]
 [  0   0 150   0]]
Sensibilidade (Recall): 0.2530
Especificidade: 0.7500
F1-Score: 0.1021

Resultados para VGG16:
Modelo: Random Forest
Acurácia: 0.9410
Matriz de Confusão:
[[142   4   0   0]
 [ 18 125   4   0]
 [  0   0 150   0]
 [  0   0   9 141]]
Sensibilidade (Recall): 0.9410
Especificidade: 0.9804
F1-Score: 0.9406
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
Modelo: SVM
Acurácia: 0.9427
Matriz de Confusão:
[[145   1   0   0]
 [ 14 131   2   0]
 [  3   0 147   0]
 [  0   0  14 136]]
Sensibilidade (Recall): 0.9427
Especificidade: 0.9809
F1-Score: 0.9428
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Modelo: RNA
Acurácia: 0.9309
Matriz de Confusão:
[[143   3   0   0]
 [ 23 123   1   0]
 [  0   8 141   1]
 [  0   0   5 145]]
Sensibilidade (Recall): 0.9309
Especificidade: 0.9770
F1-Score: 0.9307
```

h)

```
Resultados para LBP: Melhor Modelo: Random Forest Acurácia: 0.9494 F1-Score: 0.9492
```

```
Resultados para LBP: Melhor Modelo: Random Forest Acurácia: 0.9494 F1-Score: 0.9492
Resultados para VGG16: Melhor Modelo: SVM Acurácia: 0.9427 F1-Score: 0.9428
Conclusão: O Random Forest com características LBP apresentou o melhor desempenho geral (94.94% de acurácia).
O SVM com VGG16 também teve bom desempenho, mas inferior (94.27% de acurácia).
```

A análise dos resultados dos modelos de aprendizado de máquina foi realizada para duas abordagens distintas: LBP e VGG16. No contexto do LBP, o modelo Random Forest destacou-se como o mais eficaz, apresentando uma acurácia de 0.9494, além de resultados consistentes em sensibilidade e especificidade. Essa escolha é justificada pela capacidade do Random Forest em generalizar e evitar o overfitting, aspecto crucial ao trabalhar com conjuntos de dados mais reduzidos.

Por outro lado, ao avaliar o VGG16, o modelo SVM demonstrou um desempenho superior, alcançando uma acurácia de 0.9427. O SVM sobressaiu-se pela sua eficácia em lidar com dados de alta dimensão, uma característica fundamental ao lidar com características extraídas de redes neurais profundas. Sua aptidão para separar classes complexas torna-o uma escolha ideal para essa abordagem.

Em síntese, o Random Forest é o modelo recomendado para LBP devido à capacidade de generalização, enquanto o SVM é preferido para VGG16 pela sua habilidade em manipular dados complexos. Ambas as escolhas equilibram acurácia, sensibilidade e especificidade, elementos essenciais em aplicações de diagnóstico médico.

---

2 -

a)

```
Unificar diretórios teste

import os
import shutil
from pathlib import Path

# Caminhos base para os diretórios de origem e destino
base_dir = Path('/content/Test_Warwick/Test_4cl_amostra')
source_dirs = [base_dir / '20', base_dir / '80']
target_dirs = [base_dir / '0', base_dir / '1', base_dir / '2', base_dir / '3']
```

```
# Criação dos diretórios de destino, se não existirem
for target_dir in target_dirs:
    target_dir.mkdir(parents=True, exist_ok=True)

# Função para copiar arquivos das pastas de origem para as pastas de destino
def copy_files(src_dirs, dest_dirs):
    for src_dir in src_dirs:
        for i, dest_dir in enumerate(dest_dirs):
            src_class_dir = src_dir / str(i)
            if src_class_dir.exists():
                for file in src_class_dir.iterdir():
                    if file.is_file():
                        shutil.copy(file, dest_dir)
                        print(f"Copiado: {file} para {dest_dir}")

# Copiando arquivos das pastas /20 e /80 para as pastas combinadas /0, /1, /2, /3
copy_files(source_dirs, target_dirs)

print("Cópia concluída.")

# Caminhos para as pastas de treino, validação e teste
train_dir = '/content/Train_Warwick/Train_4cls_amostra/80'  # Treino
val_dir = '/content/Train_Warwick/Train_4cls_amostra/20'    # Validação
test_dir = '/content/Test_Warwick/Test_4cl_amostra/'        # Teste

# Parâmetros
IMG_SIZE = 224
BATCH_SIZE = 32
NUM_CLASSES = 4
EPOCHS = 10
```

***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.

```
# SEM Data Augmentation (apenas rescale)

train_generator = ImageDataGenerator(rescale=1./255)
test_generator = ImageDataGenerator(preprocessing_function=preprocess_input)
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,
                                        validation_split=0.2,
                                        preprocessing_function=preprocess_input)
```

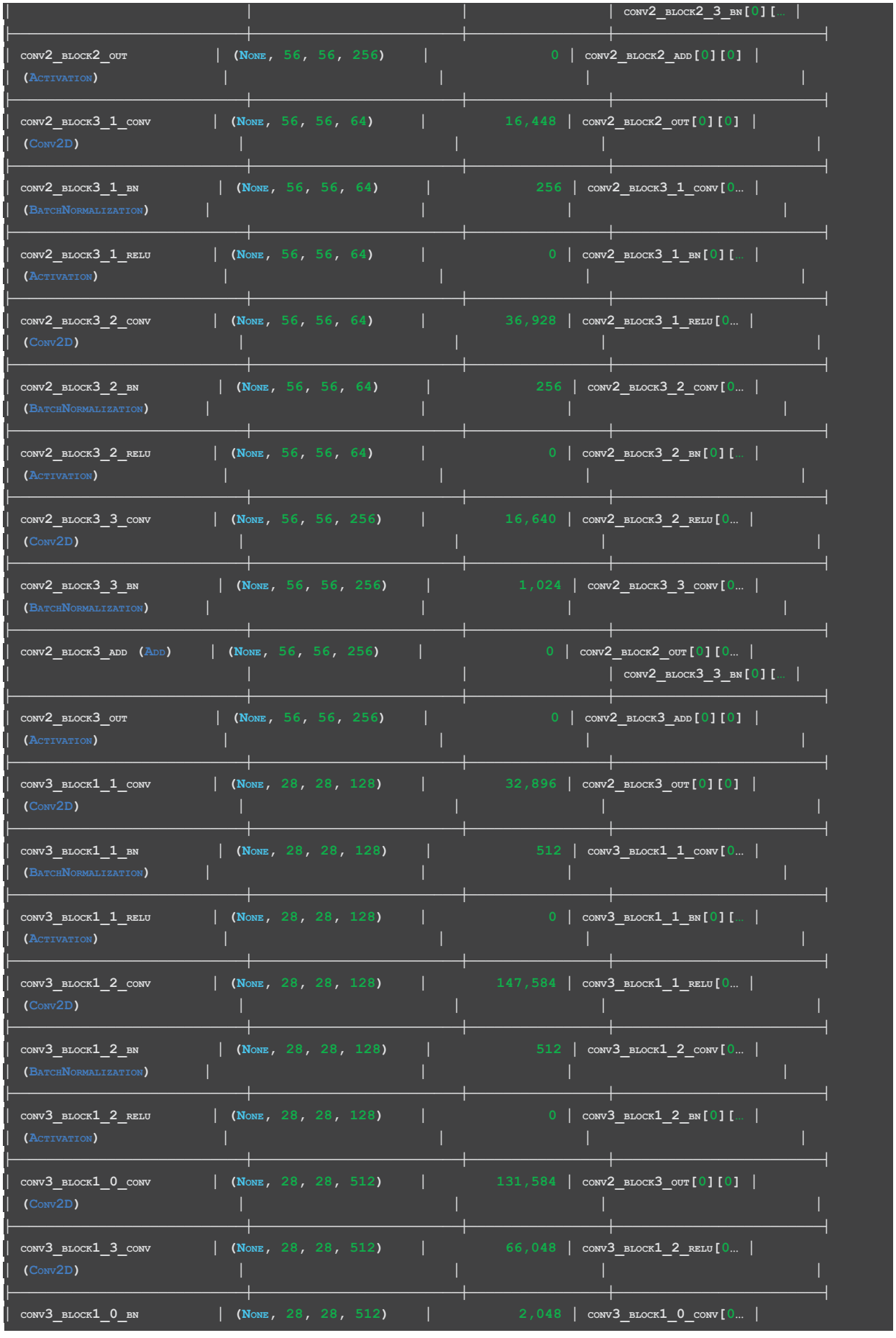
```
test_generator = ImageDataGenerator(preprocessing_function=preprocess_input)



import os

# Leitura do arquivo classes.txt e definição do class_subset
with open('/content/meta/classes.txt', 'r') as file:
    class_subset = [line.strip() for line in file.readlines()]
    class_subset_test = [line.strip() for line in file.readlines()]

# Impressão do class_subset
print(class_subset)

['0', '1', '2', '3']


traingen = train_generator.flow_from_directory(
    train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=class_subset,
    shuffle=True,
    seed=42
)

validgen = train_generator.flow_from_directory(
    val_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=class_subset,
    shuffle=True,
    seed=42
)

Found 477 images belonging to 4 classes.
Found 116 images belonging to 4 classes.



#testgen = test_generator.flow_from_directory(
#     test_dir,
#     target_size=(IMG_SIZE, IMG_SIZE),
#     batch_size=BATCH_SIZE,
#     class_mode='categorical',
#     classes=class_subset,
#     shuffle=True,
#     seed=42
#)
Found 0 images belonging to 4 classes.
```

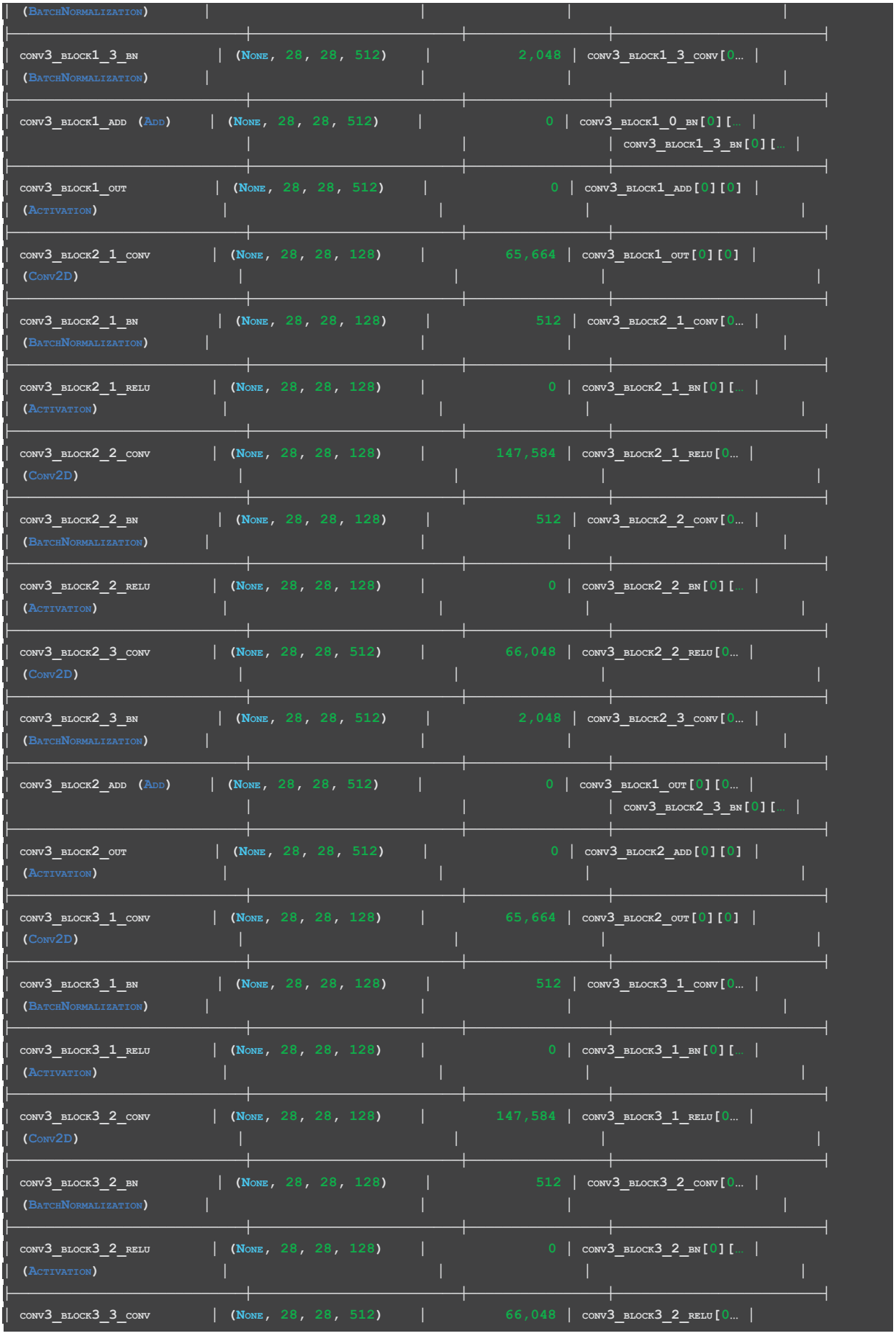
```
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from pathlib import Path

# Caminho do diretório de teste
test_dir = Path('/content/Test_Warwick/Test_4cl_amostra/' )

# Função para obter as classes de subdiretórios (no caso, das pastas /20 e /80)
def get_classes_from_subdirs(base_dir):
    subdirs_20 = os.listdir(base_dir / '20')
    subdirs_80 = os.listdir(base_dir / '80')

    # Garantir que ambos os diretórios têm as mesmas classes
    classes = set(subdirs_20).intersection(subdirs_80)
    return list(classes)

# Carregar as classes presentes em /20 e /80
class_subset = get_classes_from_subdirs(test_dir)

# Definir parâmetros
IMG_SIZE = 224  # Tamanho da imagem
BATCH_SIZE = 64  # Batch size

# Inicializar o ImageDataGenerator
test_generator = ImageDataGenerator()

# Gerar dados de teste a partir dos subdiretórios /20 e /80
testgen_20 = test_generator.flow_from_directory(
    test_dir / '20',
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=class_subset,  # Definir classes encontradas
    shuffle=True,
    seed=42
)

testgen = test_generator.flow_from_directory(
    test_dir / '80',
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=class_subset,  # Definir classes encontradas
    shuffle=True,
    seed=42
)

# Se necessário, você pode combinar os geradores ou usar um por vez

Found 119 images belonging to 4 classes.
Found 252 images belonging to 4 classes.
```

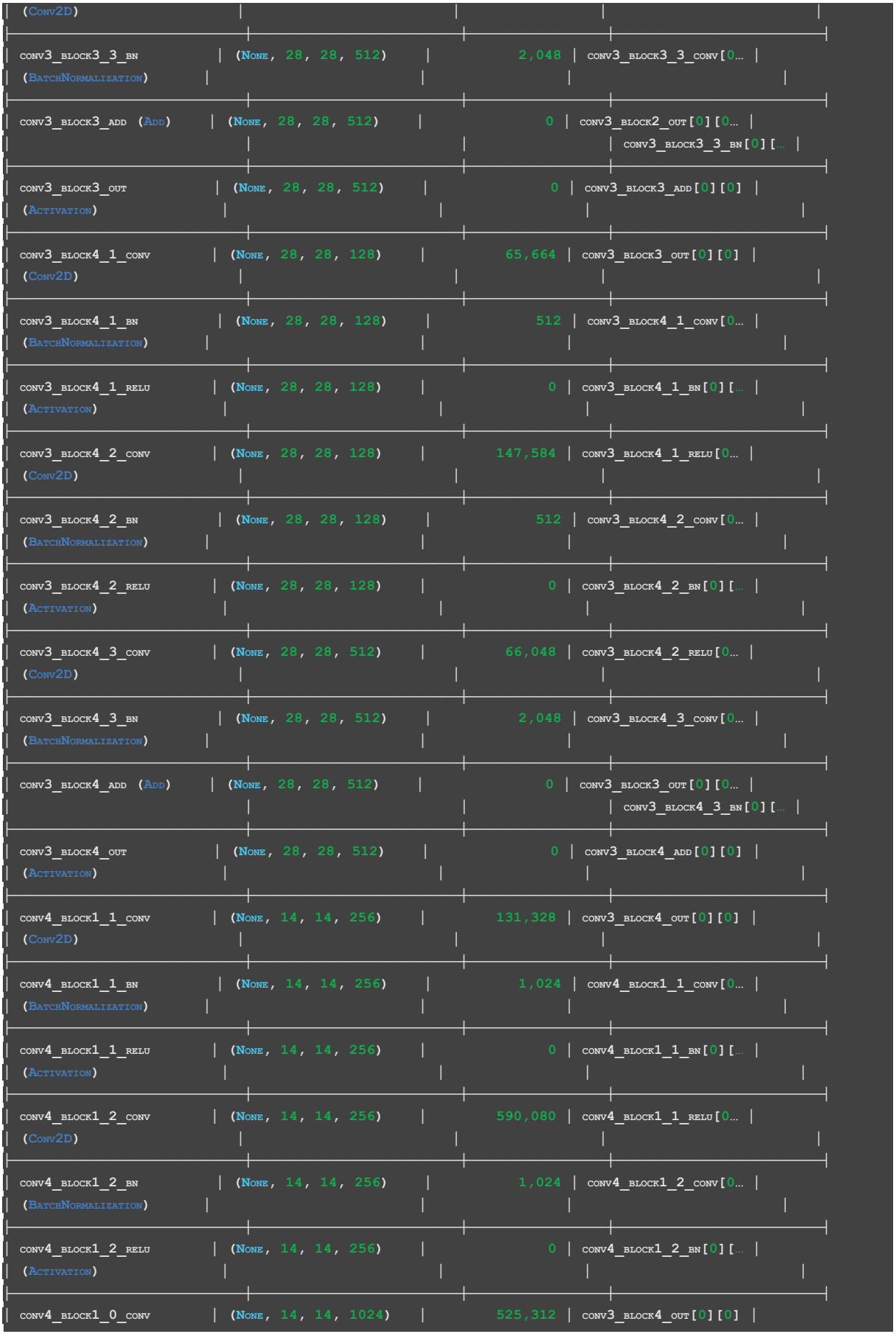
b)

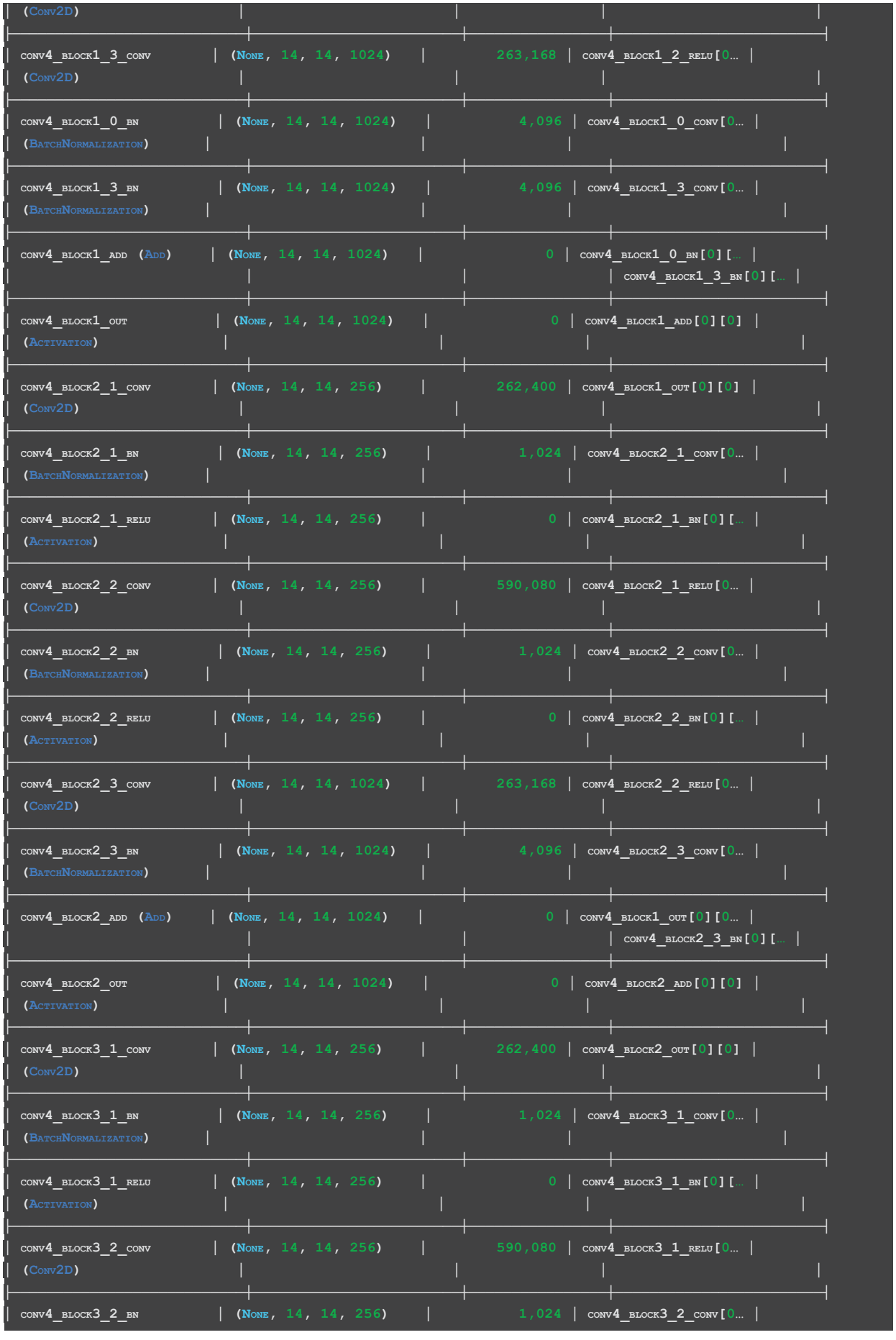
```
ResNet 50

from tensorflow.keras.applications.resnet50 import ResNet50

ResNet - do Zero - Não inclui as camadas de aprendizado da rede original

# A opção include_top=False não inclui as camadas de aprendizado da rede original
resnet_0 = ResNet50(input_shape=(224,224,3), weights=None,  include_top=False)

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

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 10 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_20"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                 ┃ Output Shape         ┃    Param # ┃ Connected to        ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━┩
│ input_layer_4                │ (None, 224, 224, 3)  │          0 │ -                   │
│ (InputLayer)                 │                      │            │                     │
├──────────────────────────────┼──────────────────────┼────────────┼─────────────────────┤
│ conv1_pad (ZeroPadding2D)    │ (None, 230, 230, 3)  │          0 │ input_layer_4[0][0] │
├──────────────────────────────┼──────────────────────┼────────────┼─────────────────────┤
│ conv1_conv (Conv2D)          │ (None, 112, 112, 64) │      9,472 │ conv1_pad[0][0]     │
├──────────────────────────────┼──────────────────────┼────────────┼─────────────────────┤
│ conv1_bn                     │ (None, 112, 112, 64) │        256 │ conv1_conv[0][0]    │
│ (BatchNormalization)         │                      │            │                     │
├──────────────────────────────┼──────────────────────┼────────────┼─────────────────────┤
│ conv1_relu (Activation)      │ (None, 112, 112, 64) │          0 │ conv1_bn[0][0]      │
├──────────────────────────────┼──────────────────────┼────────────┼─────────────────────┤
│ pool1_pad (ZeroPadding2D)    │ (None, 114, 114, 64) │          0 │ conv1_relu[0][0]    │
├──────────────────────────────┼──────────────────────┼────────────┼─────────────────────┤
│ pool1_pool (MaxPooling2D)    │ (None, 56, 56, 64)   │          0 │ pool1_pad[0][0]     │
├──────────────────────────────┼──────────────────────┼────────────┼─────────────────────┤
│ conv2_block1_1_conv          │ (None, 56, 56, 64)   │      4,160 │ pool1_pool[0][0]    │
│ (Conv2D)                     │                      │            │                     │
```

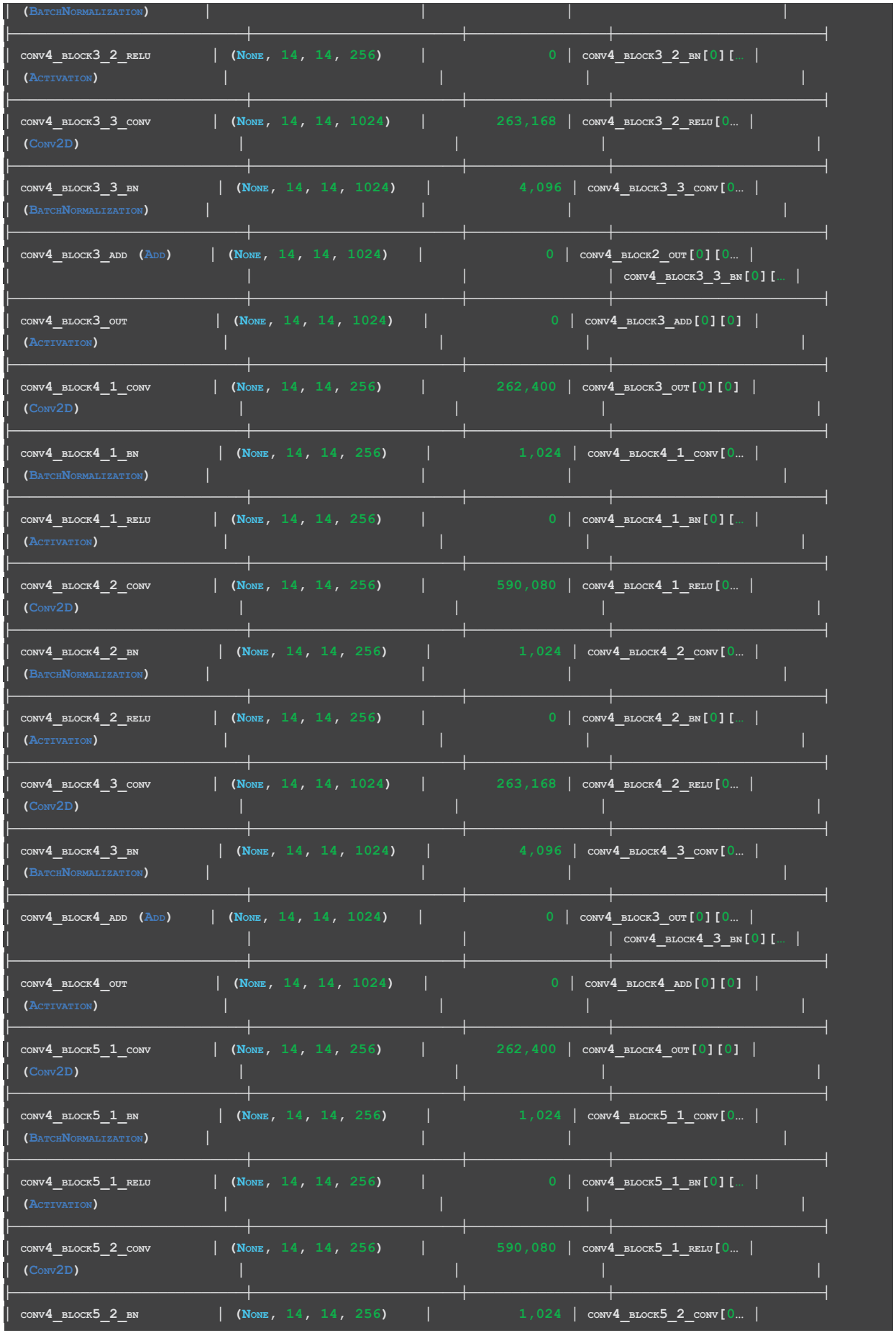
```
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_1_bn        │ (None, 56, 56, 64)       │              256 │ conv2_block1_1_conv[0…   │
│ (BatchNormalization)     │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_1_relu      │ (None, 56, 56, 64)       │                0 │ conv2_block1_1_bn[0][…   │
│ (Activation)             │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_2_conv      │ (None, 56, 56, 64)       │           36,928 │ conv2_block1_1_relu[0…   │
│ (Conv2D)                 │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_2_bn        │ (None, 56, 56, 64)       │              256 │ conv2_block1_2_conv[0…   │
│ (BatchNormalization)     │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_2_relu      │ (None, 56, 56, 64)       │                0 │ conv2_block1_2_bn[0][…   │
│ (Activation)             │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_0_conv      │ (None, 56, 56, 256)      │           16,640 │ pool1_pool[0][0]         │
│ (Conv2D)                 │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_3_conv      │ (None, 56, 56, 256)      │           16,640 │ conv2_block1_2_relu[0…   │
│ (Conv2D)                 │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_0_bn        │ (None, 56, 56, 256)      │            1,024 │ conv2_block1_0_conv[0…   │
│ (BatchNormalization)     │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_3_bn        │ (None, 56, 56, 256)      │            1,024 │ conv2_block1_3_conv[0…   │
│ (BatchNormalization)     │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_add (Add)   │ (None, 56, 56, 256)      │                0 │ conv2_block1_0_bn[0][…   │
│                          │                          │                  │ conv2_block1_3_bn[0][…   │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block1_out         │ (None, 56, 56, 256)      │                0 │ conv2_block1_add[0][0]   │
│ (Activation)             │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_1_conv      │ (None, 56, 56, 64)       │           16,448 │ conv2_block1_out[0][0]   │
│ (Conv2D)                 │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_1_bn        │ (None, 56, 56, 64)       │              256 │ conv2_block2_1_conv[0…   │
│ (BatchNormalization)     │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_1_relu      │ (None, 56, 56, 64)       │                0 │ conv2_block2_1_bn[0][…   │
│ (Activation)             │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_2_conv      │ (None, 56, 56, 64)       │           36,928 │ conv2_block2_1_relu[0…   │
│ (Conv2D)                 │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_2_bn        │ (None, 56, 56, 64)       │              256 │ conv2_block2_2_conv[0…   │
│ (BatchNormalization)     │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_2_relu      │ (None, 56, 56, 64)       │                0 │ conv2_block2_2_bn[0][…   │
│ (Activation)             │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_3_conv      │ (None, 56, 56, 256)      │           16,640 │ conv2_block2_2_relu[0…   │
│ (Conv2D)                 │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_3_bn        │ (None, 56, 56, 256)      │            1,024 │ conv2_block2_3_conv[0…   │
│ (BatchNormalization)     │                          │                  │                          │
├──────────────────────────┼──────────────────────────┼──────────────────┼──────────────────────────┤
│ conv2_block2_add (Add)   │ (None, 56, 56, 256)      │                0 │ conv2_block1_out[0][0…   │
```

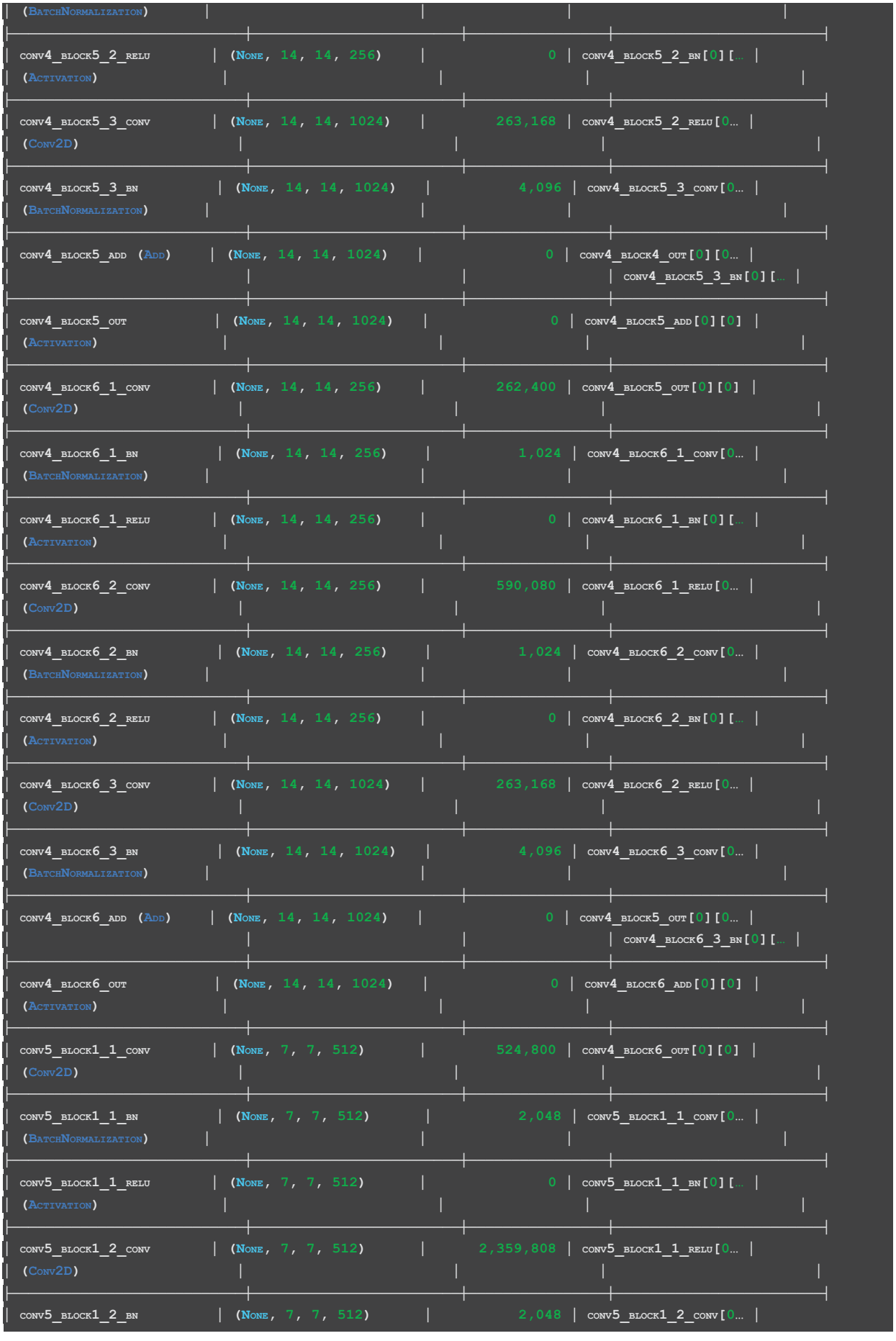

conv2_block2_3_bn[0][...
conv2_block2_out (Activation) (None, 56, 56, 256) 0 conv2_block2_add[0][0]
conv2_block3_1_conv (Conv2D) (None, 56, 56, 64) 16,448 conv2_block2_out[0][0]
conv2_block3_1_bn (BatchNormalization) (None, 56, 56, 64) 256 conv2_block3_1_conv[0...
conv2_block3_1_relu (Activation) (None, 56, 56, 64) 0 conv2_block3_1_bn[0][...
conv2_block3_2_conv (Conv2D) (None, 56, 56, 64) 36,928 conv2_block3_1_relu[0...
conv2_block3_2_bn (BatchNormalization) (None, 56, 56, 64) 256 conv2_block3_2_conv[0...
conv2_block3_2_relu (Activation) (None, 56, 56, 64) 0 conv2_block3_2_bn[0][...
conv2_block3_3_conv (Conv2D) (None, 56, 56, 256) 16,640 conv2_block3_2_relu[0...
conv2_block3_3_bn (BatchNormalization) (None, 56, 56, 256) 1,024 conv2_block3_3_conv[0...
conv2_block3_add (Add) (None, 56, 56, 256) 0 conv2_block2_out[0][0...
conv2_block3_3_bn[0][...
conv2_block3_out (Activation) (None, 56, 56, 256) 0 conv2_block3_add[0][0]
conv3_block1_1_conv (Conv2D) (None, 28, 28, 128) 32,896 conv2_block3_out[0][0]
conv3_block1_1_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block1_1_conv[0...
conv3_block1_1_relu (Activation) (None, 28, 28, 128) 0 conv3_block1_1_bn[0][...
conv3_block1_2_conv (Conv2D) (None, 28, 28, 128) 147,584 conv3_block1_1_relu[0...
conv3_block1_2_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block1_2_conv[0...
conv3_block1_2_relu (Activation) (None, 28, 28, 128) 0 conv3_block1_2_bn[0][...
conv3_block1_0_conv (Conv2D) (None, 28, 28, 512) 131,584 conv2_block3_out[0][0]
conv3_block1_3_conv (Conv2D) (None, 28, 28, 512) 66,048 conv3_block1_2_relu[0...
conv3_block1_0_bn (None, 28, 28, 512) 2,048 conv3_block1_0_conv[0...

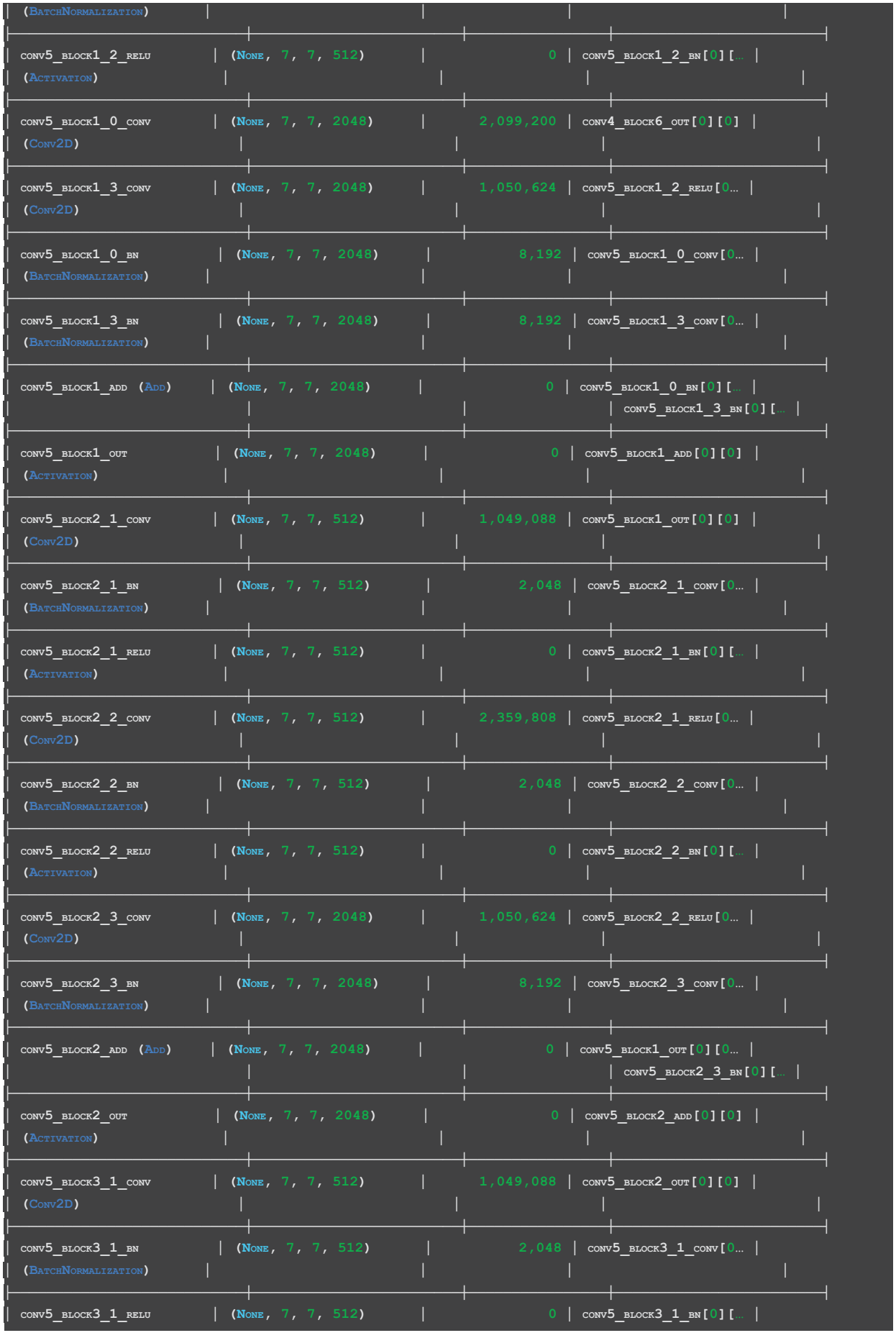

(BatchNormalization)
conv3_block1_3_bn (BatchNormalization) (None, 28, 28, 512) 2,048 conv3_block1_3_conv[0...
conv3_block1_add (Add) (None, 28, 28, 512) 0 conv3_block1_0_bn[0][... conv3_block1_3_bn[0][...
conv3_block1_out (Activation) (None, 28, 28, 512) 0 conv3_block1_add[0][0]
conv3_block2_1_conv (Conv2D) (None, 28, 28, 128) 65,664 conv3_block1_out[0][0]
conv3_block2_1_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block2_1_conv[0...
conv3_block2_1_relu (Activation) (None, 28, 28, 128) 0 conv3_block2_1_bn[0][...
conv3_block2_2_conv (Conv2D) (None, 28, 28, 128) 147,584 conv3_block2_1_relu[0...
conv3_block2_2_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block2_2_conv[0...
conv3_block2_2_relu (Activation) (None, 28, 28, 128) 0 conv3_block2_2_bn[0][...
conv3_block2_3_conv (Conv2D) (None, 28, 28, 512) 66,048 conv3_block2_2_relu[0...
conv3_block2_3_bn (BatchNormalization) (None, 28, 28, 512) 2,048 conv3_block2_3_conv[0...
conv3_block2_add (Add) (None, 28, 28, 512) 0 conv3_block1_out[0][0... conv3_block2_3_bn[0][...
conv3_block2_out (Activation) (None, 28, 28, 512) 0 conv3_block2_add[0][0]
conv3_block3_1_conv (Conv2D) (None, 28, 28, 128) 65,664 conv3_block2_out[0][0]
conv3_block3_1_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block3_1_conv[0...
conv3_block3_1_relu (Activation) (None, 28, 28, 128) 0 conv3_block3_1_bn[0][...
conv3_block3_2_conv (Conv2D) (None, 28, 28, 128) 147,584 conv3_block3_1_relu[0...
conv3_block3_2_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block3_2_conv[0...
conv3_block3_2_relu (Activation) (None, 28, 28, 128) 0 conv3_block3_2_bn[0][...
conv3_block3_3_conv (None, 28, 28, 512) 66,048 conv3_block3_2_relu[0...


(Conv2D)
conv3_block3_3_bn (BatchNormalization) (None, 28, 28, 512) 2,048 conv3_block3_3_conv[0...
conv3_block3_add (Add) (None, 28, 28, 512) 0 conv3_block2_out[0][0... conv3_block3_3_bn[0][...
conv3_block3_out (Activation) (None, 28, 28, 512) 0 conv3_block3_add[0][0]
conv3_block4_1_conv (Conv2D) (None, 28, 28, 128) 65,664 conv3_block3_out[0][0]
conv3_block4_1_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block4_1_conv[0...
conv3_block4_1_relu (Activation) (None, 28, 28, 128) 0 conv3_block4_1_bn[0][...
conv3_block4_2_conv (Conv2D) (None, 28, 28, 128) 147,584 conv3_block4_1_relu[0...
conv3_block4_2_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block4_2_conv[0...
conv3_block4_2_relu (Activation) (None, 28, 28, 128) 0 conv3_block4_2_bn[0][...
conv3_block4_3_conv (Conv2D) (None, 28, 28, 512) 66,048 conv3_block4_2_relu[0...
conv3_block4_3_bn (BatchNormalization) (None, 28, 28, 512) 2,048 conv3_block4_3_conv[0...
conv3_block4_add (Add) (None, 28, 28, 512) 0 conv3_block3_out[0][0... conv3_block4_3_bn[0][...
conv3_block4_out (Activation) (None, 28, 28, 512) 0 conv3_block4_add[0][0]
conv4_block1_1_conv (Conv2D) (None, 14, 14, 256) 131,328 conv3_block4_out[0][0]
conv4_block1_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block1_1_conv[0...
conv4_block1_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block1_1_bn[0][...
conv4_block1_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block1_1_relu[0...
conv4_block1_2_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block1_2_conv[0...
conv4_block1_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block1_2_bn[0][...
conv4_block1_0_conv (None, 14, 14, 1024) 525,312 conv3_block4_out[0][0]


(Conv2D)
conv4_block1_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block1_2_relu[0...
conv4_block1_0_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block1_0_conv[0...
conv4_block1_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block1_3_conv[0...
conv4_block1_add (Add) (None, 14, 14, 1024) 0 conv4_block1_0_bn[0][...
conv4_block1_3_bn[0][...
conv4_block1_out (Activation) (None, 14, 14, 1024) 0 conv4_block1_add[0][0]
conv4_block2_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block1_out[0][0]
conv4_block2_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block2_1_conv[0...
conv4_block2_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block2_1_bn[0][...
conv4_block2_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block2_1_relu[0...
conv4_block2_2_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block2_2_conv[0...
conv4_block2_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block2_2_bn[0][...
conv4_block2_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block2_2_relu[0...
conv4_block2_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block2_3_conv[0...
conv4_block2_add (Add) (None, 14, 14, 1024) 0 conv4_block1_out[0][0...
conv4_block2_3_bn[0][...
conv4_block2_out (Activation) (None, 14, 14, 1024) 0 conv4_block2_add[0][0]
conv4_block3_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block2_out[0][0]
conv4_block3_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block3_1_conv[0...
conv4_block3_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block3_1_bn[0][...
conv4_block3_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block3_1_relu[0...
conv4_block3_2_bn (None, 14, 14, 256) 1,024 conv4_block3_2_conv[0...


(BatchNormalization)
conv4_block3_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block3_2_bn[0][...
conv4_block3_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block3_2_relu[0...
conv4_block3_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block3_3_conv[0...
conv4_block3_add (Add) (None, 14, 14, 1024) 0 conv4_block2_out[0][0... conv4_block3_3_bn[0][...
conv4_block3_out (Activation) (None, 14, 14, 1024) 0 conv4_block3_add[0][0]
conv4_block4_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block3_out[0][0]
conv4_block4_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block4_1_conv[0...
conv4_block4_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block4_1_bn[0][...
conv4_block4_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block4_1_relu[0...
conv4_block4_2_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block4_2_conv[0...
conv4_block4_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block4_2_bn[0][...
conv4_block4_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block4_2_relu[0...
conv4_block4_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block4_3_conv[0...
conv4_block4_add (Add) (None, 14, 14, 1024) 0 conv4_block3_out[0][0... conv4_block4_3_bn[0][...
conv4_block4_out (Activation) (None, 14, 14, 1024) 0 conv4_block4_add[0][0]
conv4_block5_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block4_out[0][0]
conv4_block5_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block5_1_conv[0...
conv4_block5_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block5_1_bn[0][...
conv4_block5_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block5_1_relu[0...
conv4_block5_2_bn (None, 14, 14, 256) 1,024 conv4_block5_2_conv[0...


(BatchNormalization)
conv4_block5_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block5_2_bn[0][...
conv4_block5_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block5_2_relu[0...
conv4_block5_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block5_3_conv[0...
conv4_block5_add (Add) (None, 14, 14, 1024) 0 conv4_block4_out[0][0... conv4_block5_3_bn[0][...
conv4_block5_out (Activation) (None, 14, 14, 1024) 0 conv4_block5_add[0][0]
conv4_block6_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block5_out[0][0]
conv4_block6_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block6_1_conv[0...
conv4_block6_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block6_1_bn[0][...
conv4_block6_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block6_1_relu[0...
conv4_block6_2_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block6_2_conv[0...
conv4_block6_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block6_2_bn[0][...
conv4_block6_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block6_2_relu[0...
conv4_block6_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block6_3_conv[0...
conv4_block6_add (Add) (None, 14, 14, 1024) 0 conv4_block5_out[0][0... conv4_block6_3_bn[0][...
conv4_block6_out (Activation) (None, 14, 14, 1024) 0 conv4_block6_add[0][0]
conv5_block1_1_conv (Conv2D) (None, 7, 7, 512) 524,800 conv4_block6_out[0][0]
conv5_block1_1_bn (BatchNormalization) (None, 7, 7, 512) 2,048 conv5_block1_1_conv[0...
conv5_block1_1_relu (Activation) (None, 7, 7, 512) 0 conv5_block1_1_bn[0][...
conv5_block1_2_conv (Conv2D) (None, 7, 7, 512) 2,359,808 conv5_block1_1_relu[0...
conv5_block1_2_bn (None, 7, 7, 512) 2,048 conv5_block1_2_conv[0...


(BatchNormalization)
conv5_block1_2_relu (Activation) (None, 7, 7, 512) 0 conv5_block1_2_bn[0][...
conv5_block1_0_conv (Conv2D) (None, 7, 7, 2048) 2,099,200 conv4_block6_out[0][0]
conv5_block1_3_conv (Conv2D) (None, 7, 7, 2048) 1,050,624 conv5_block1_2_relu[0...
conv5_block1_0_bn (BatchNormalization) (None, 7, 7, 2048) 8,192 conv5_block1_0_conv[0...
conv5_block1_3_bn (BatchNormalization) (None, 7, 7, 2048) 8,192 conv5_block1_3_conv[0...
conv5_block1_add (Add) (None, 7, 7, 2048) 0 conv5_block1_0_bn[0][...
conv5_block1_3_bn[0][...
conv5_block1_out (Activation) (None, 7, 7, 2048) 0 conv5_block1_add[0][0]
conv5_block2_1_conv (Conv2D) (None, 7, 7, 512) 1,049,088 conv5_block1_out[0][0]
conv5_block2_1_bn (BatchNormalization) (None, 7, 7, 512) 2,048 conv5_block2_1_conv[0...
conv5_block2_1_relu (Activation) (None, 7, 7, 512) 0 conv5_block2_1_bn[0][...
conv5_block2_2_conv (Conv2D) (None, 7, 7, 512) 2,359,808 conv5_block2_1_relu[0...
conv5_block2_2_bn (BatchNormalization) (None, 7, 7, 512) 2,048 conv5_block2_2_conv[0...
conv5_block2_2_relu (Activation) (None, 7, 7, 512) 0 conv5_block2_2_bn[0][...
conv5_block2_3_conv (Conv2D) (None, 7, 7, 2048) 1,050,624 conv5_block2_2_relu[0...
conv5_block2_3_bn (BatchNormalization) (None, 7, 7, 2048) 8,192 conv5_block2_3_conv[0...
conv5_block2_add (Add) (None, 7, 7, 2048) 0 conv5_block1_out[0][0...
conv5_block2_3_bn[0][...
conv5_block2_out (Activation) (None, 7, 7, 2048) 0 conv5_block2_add[0][0]
conv5_block3_1_conv (Conv2D) (None, 7, 7, 512) 1,049,088 conv5_block2_out[0][0]
conv5_block3_1_bn (BatchNormalization) (None, 7, 7, 512) 2,048 conv5_block3_1_conv[0...
conv5_block3_1_relu (None, 7, 7, 512) 0 conv5_block3_1_bn[0][...

```
| (Activation)                 |                           |                 |                              |
|------------------------------|---------------------------|-----------------|------------------------------|
| conv5_block3_2_conv          | (None, 7, 7, 512)         |       2,359,808 | conv5_block3_1_relu[0…       |
| (Conv2D)                     |                           |                 |                              |
|------------------------------|---------------------------|-----------------|------------------------------|
| conv5_block3_2_bn            | (None, 7, 7, 512)         |           2,048 | conv5_block3_2_conv[0…       |
| (BatchNormalization)         |                           |                 |                              |
|------------------------------|---------------------------|-----------------|------------------------------|
| conv5_block3_2_relu          | (None, 7, 7, 512)         |               0 | conv5_block3_2_bn[0][…       |
| (Activation)                 |                           |                 |                              |
|------------------------------|---------------------------|-----------------|------------------------------|
| conv5_block3_3_conv          | (None, 7, 7, 2048)        |       1,050,624 | conv5_block3_2_relu[0…       |
| (Conv2D)                     |                           |                 |                              |
|------------------------------|---------------------------|-----------------|------------------------------|
| conv5_block3_3_bn            | (None, 7, 7, 2048)        |           8,192 | conv5_block3_3_conv[0…       |
| (BatchNormalization)         |                           |                 |                              |
|------------------------------|---------------------------|-----------------|------------------------------|
| conv5_block3_add (Add)       | (None, 7, 7, 2048)        |               0 | conv5_block2_out[0][0…       |
|                              |                           |                 | conv5_block3_3_bn[0][…       |
|------------------------------|---------------------------|-----------------|------------------------------|
| conv5_block3_out             | (None, 7, 7, 2048)        |               0 | conv5_block3_add[0][0]       |
| (Activation)                 |                           |                 |                              |
|------------------------------|---------------------------|-----------------|------------------------------|
| flatten (Flatten)            | (None, 100352)            |               0 | conv5_block3_out[0][0]       |
|------------------------------|---------------------------|-----------------|------------------------------|
| dense_6 (Dense)              | (None, 4)                 |         401,412 | flatten[0][0]                |
|______________________________|___________________________|_________________|______________________________|

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

Trainable params: 23,936,004 (91.31 MB)

Non-trainable params: 53,120 (207.50 KB)
```

Treinamento

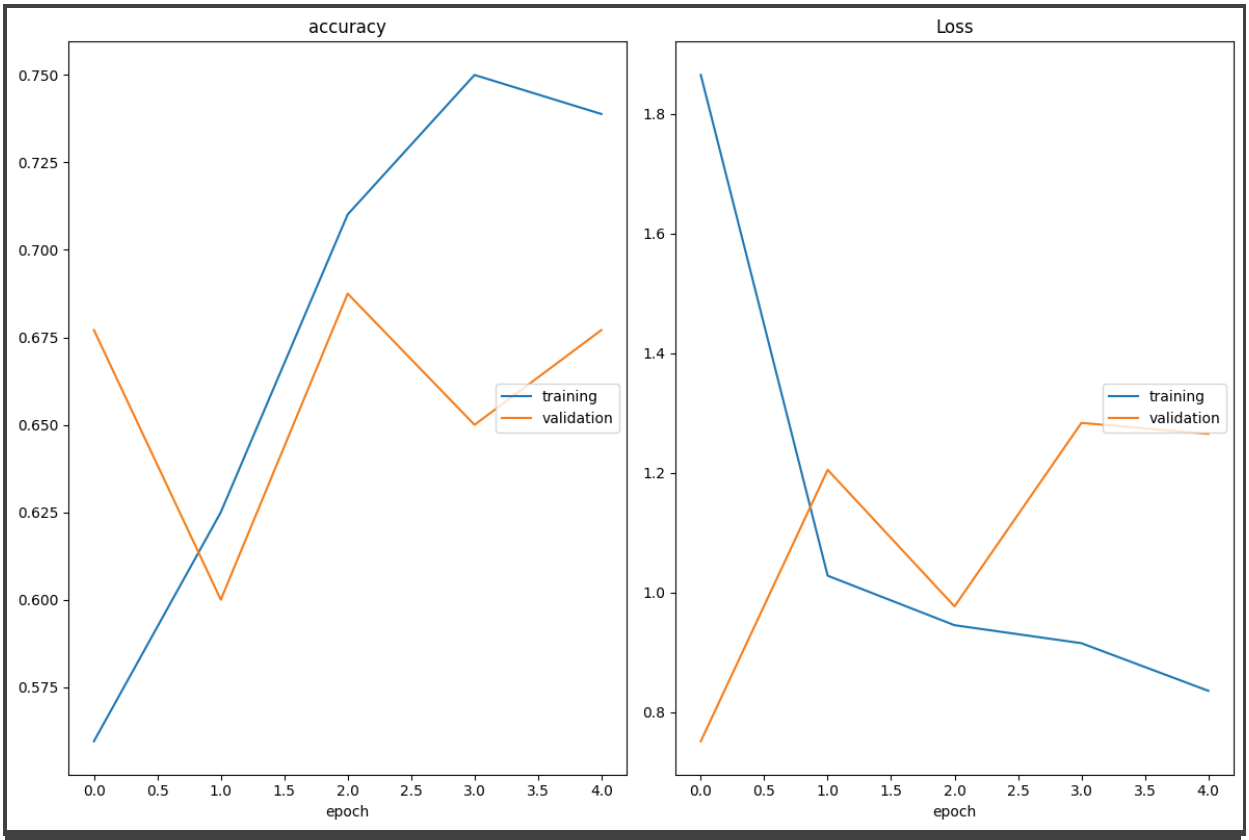
```
%%time
# Otimizador propagação da Raiz quadrada da média ao quadrado (Root Mean Squared Propagation)
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

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'])

# 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
# 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')

# Treinamento do Modelo
history_0 = model_0.fit(traingen,
                        epochs=n_epochs,
                        steps_per_epoch=steps_per_epoch,
                        validation_data=validgen,
                        validation_steps=val_steps,
                        callbacks=[checkpointer, PlotLossesKeras()],
                        #callbacks=[early_stop, checkpointer, PlotLossesKeras()],
                        verbose=False)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<timed exec> in <module>

NameError: name 'traingen' is not defined
```

```
Resnet com Transfer Learning

# A opção include_top=False não inclui as camadas de aprendizado da rede original
# Utiliza os pesos treinados na base imagenet
resnet_tl = ResNet50(input_shape=(224,224,3), weights='imagenet', include_top=False)

# não treinar os pesos existentes
for layer in resnet_tl.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 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step


Criação das camadas Fully Connected para o treinamento
Rede igual ao treinamento do zero, porém utilizando os pesos da Imagenet.

# camadas próprias - você pode colocar mais se quiser
# A saída da resnet será a entrada da camada criada
x_tl = Flatten()(resnet_tl.output)

# camada de classificação com as 10 classes utilizadas
prediction = Dense(len(class_subset), activation='softmax')(x_tl)
```

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



Treinamento

%%time

from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

steps_per_epoch = traingen.samples // BATCH_SIZE
val_steps = validgen.samples // BATCH_SIZE

n_epochs = 10

optimizer = RMSprop(learning_rate=0.0001)

model_tl.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_tl.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')

# Treinamento do Modelo
history_tl = model_tl.fit(traingen,
                          epochs=n_epochs,
                          steps_per_epoch=steps_per_epoch,
                          validation_data=validgen,
                          validation_steps=val_steps,
                          callbacks=[checkpointer, PlotLossesKeras()],
                          #callbacks=[early_stop, checkpointer, PlotLossesKeras()],
                          verbose=False)
```



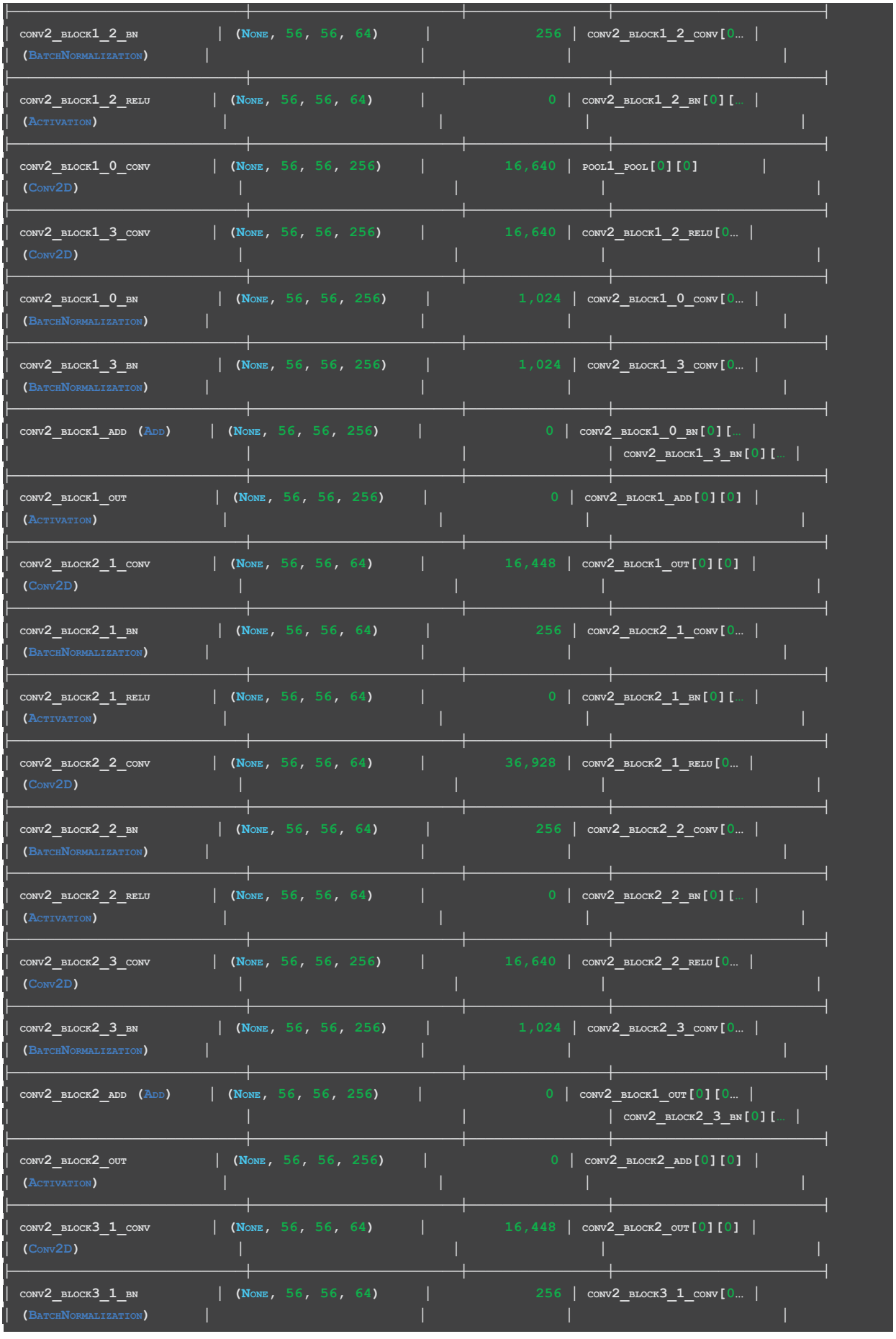

```
accuracy
        training                (min:    0.560, max:    0.750, cur:    0.739)
        validation              (min:    0.600, max:    0.688, cur:    0.677)
Loss
        training                (min:    0.836, max:    1.865, cur:    0.836)
        validation              (min:    0.752, max:    1.283, cur:    1.265)
CPU times: user 8min 15s, sys: 27.6 s, total: 8min 42s
Wall time: 6min 2s
```

```
Fine Tunning

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

Ajuste do Fine Tunning.
A camada interna da REsnet50 chamada 'conv_block3_out' da será treinada. Por ser uma das últimas camadas é responsável pelo parte de mais alto nível. Além disso uma camada de AveragePooling será conectada a saída da rede, junto com uma Flatten, uma densa, um Drop e a de predição.

#Indica que a rede vai ser treinável. Porém, na parte de baixo são indicadas as
# camadas que irão ser treinadas.
resnet_ft.trainable = True

# treinar somente a última camada antes das Totalmente conecetadas - Fine Tunning
for layer in resnet_ft.layers:
   if layer.name == 'conv5_block3_out':
       layer.trainable = True
```

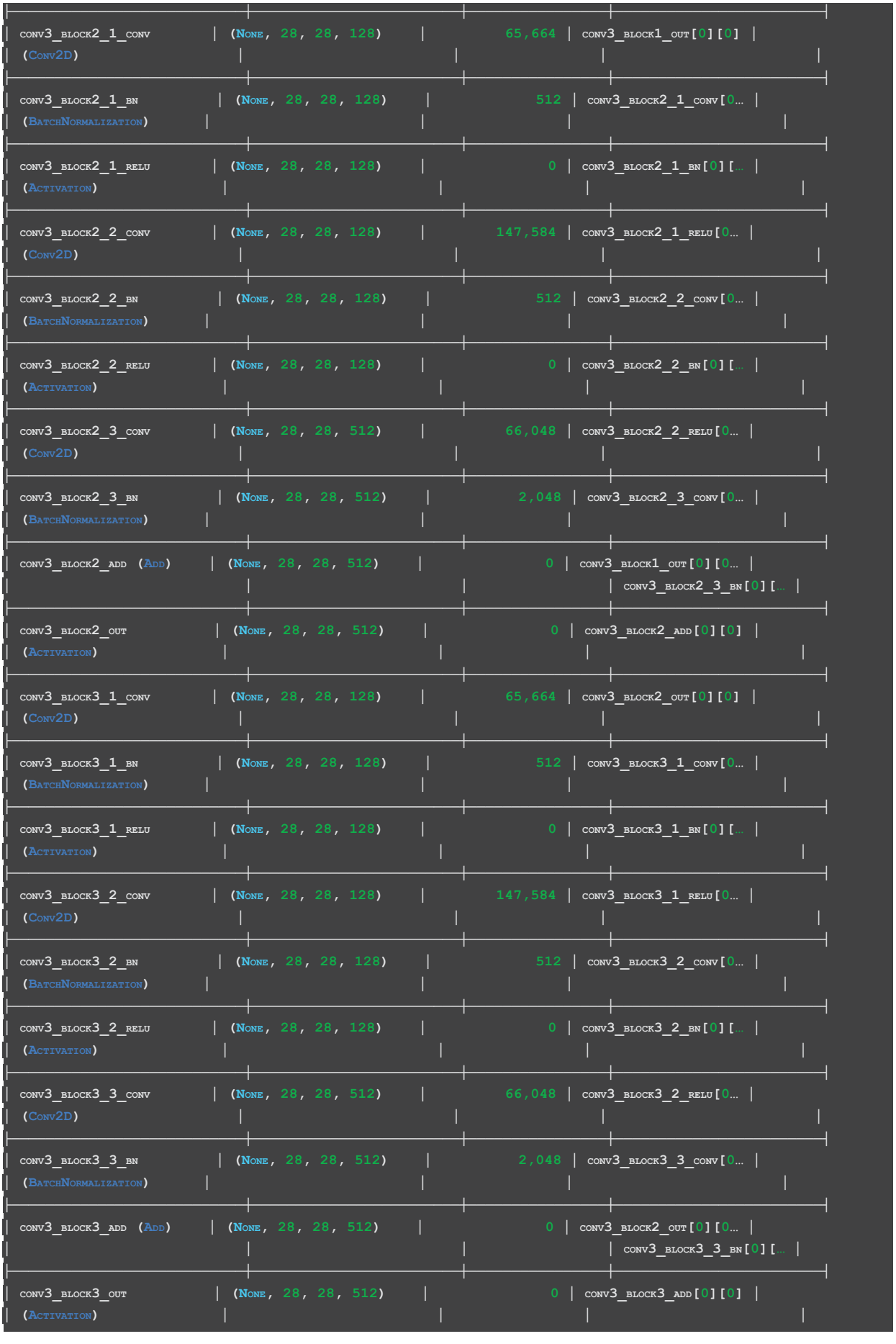
```
    else:
        layer.trainable = False

from tensorflow.keras.layers import AveragePooling2D

from tensorflow.keras.layers import Dropout

# camadas próprias - você pode colocar mais se quiser
# A saída da resnet será a entrada da camada criada
pool_ft = AveragePooling2D(pool_size=(7,7))(resnet_ft.output)
x_ft = Flatten()(pool_ft)
x2_ft = Dense(1024, activation='relu')(x_ft)
drop_ft = Dropout(0.2)(x2_ft)

# camada de classificação com as 10 classes utilizadas
prediction = Dense(len(class_subset), activation='softmax')(drop_ft)

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

model_ft.summary()
```

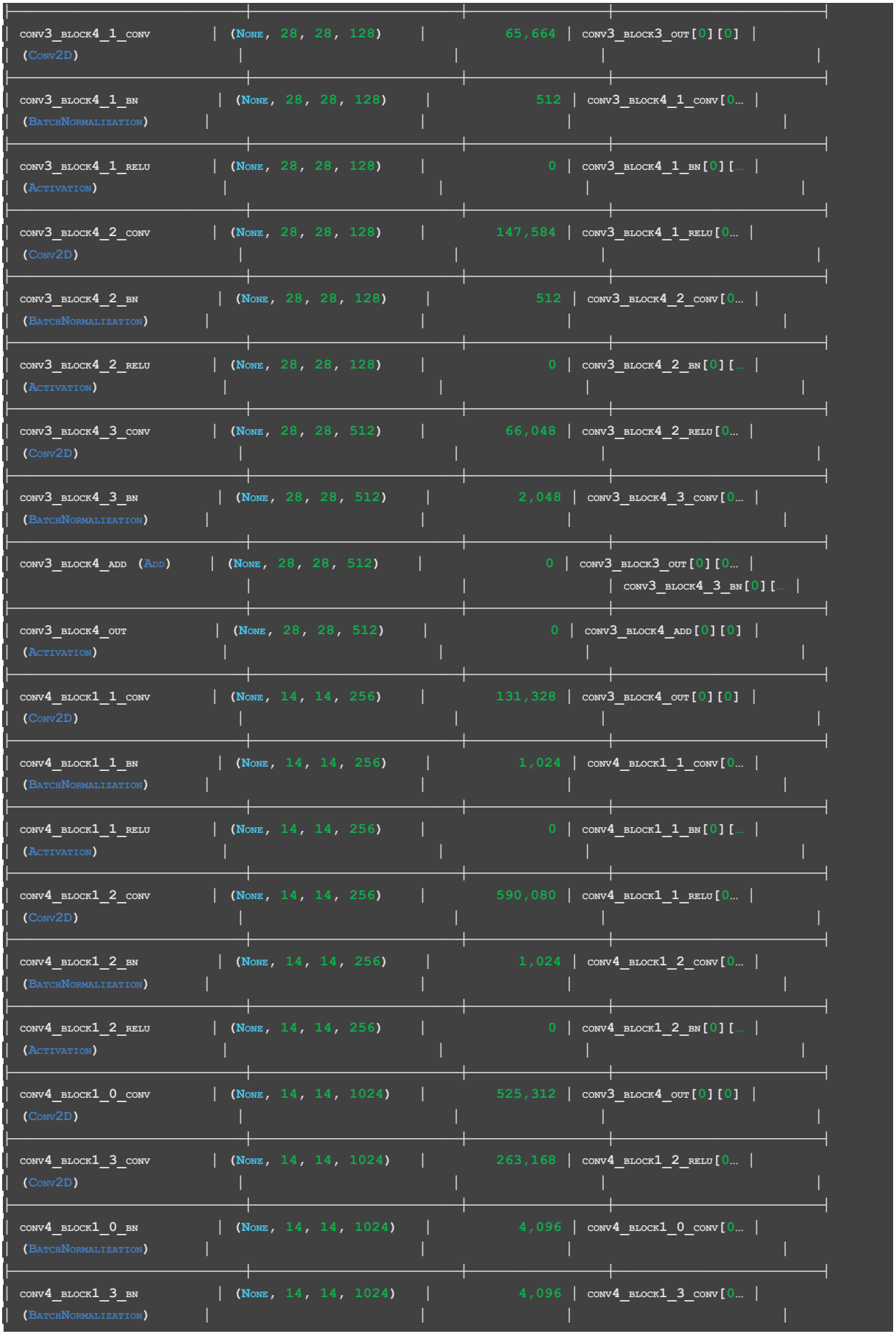
```
Model: "functional_37"
```

Layer (type)	Output Shape	Param #	Connected to
input_layer_11 (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer_11[0][0]
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalization)	(None, 56, 56, 64)	256	conv2_block1_1_conv[0...
conv2_block1_1_relu (Activation)	(None, 56, 56, 64)	0	conv2_block1_1_bn[0][...
conv2_block1_2_conv (Conv2D)	(None, 56, 56, 64)	36,928	conv2_block1_1_relu[0...


conv2_block1_2_bn (BatchNormalization) (None, 56, 56, 64) 256 conv2_block1_2_conv[0...
conv2_block1_2_relu (Activation) (None, 56, 56, 64) 0 conv2_block1_2_bn[0][...
conv2_block1_0_conv (Conv2D) (None, 56, 56, 256) 16,640 pool1_pool[0][0]
conv2_block1_3_conv (Conv2D) (None, 56, 56, 256) 16,640 conv2_block1_2_relu[0...
conv2_block1_0_bn (BatchNormalization) (None, 56, 56, 256) 1,024 conv2_block1_0_conv[0...
conv2_block1_3_bn (BatchNormalization) (None, 56, 56, 256) 1,024 conv2_block1_3_conv[0...
conv2_block1_add (Add) (None, 56, 56, 256) 0 conv2_block1_0_bn[0][...
conv2_block1_3_bn[0][...
conv2_block1_out (Activation) (None, 56, 56, 256) 0 conv2_block1_add[0][0]
conv2_block2_1_conv (Conv2D) (None, 56, 56, 64) 16,448 conv2_block1_out[0][0]
conv2_block2_1_bn (BatchNormalization) (None, 56, 56, 64) 256 conv2_block2_1_conv[0...
conv2_block2_1_relu (Activation) (None, 56, 56, 64) 0 conv2_block2_1_bn[0][...
conv2_block2_2_conv (Conv2D) (None, 56, 56, 64) 36,928 conv2_block2_1_relu[0...
conv2_block2_2_bn (BatchNormalization) (None, 56, 56, 64) 256 conv2_block2_2_conv[0...
conv2_block2_2_relu (Activation) (None, 56, 56, 64) 0 conv2_block2_2_bn[0][...
conv2_block2_3_conv (Conv2D) (None, 56, 56, 256) 16,640 conv2_block2_2_relu[0...
conv2_block2_3_bn (BatchNormalization) (None, 56, 56, 256) 1,024 conv2_block2_3_conv[0...
conv2_block2_add (Add) (None, 56, 56, 256) 0 conv2_block1_out[0][0...
conv2_block2_3_bn[0][...
conv2_block2_out (Activation) (None, 56, 56, 256) 0 conv2_block2_add[0][0]
conv2_block3_1_conv (Conv2D) (None, 56, 56, 64) 16,448 conv2_block2_out[0][0]
conv2_block3_1_bn (BatchNormalization) (None, 56, 56, 64) 256 conv2_block3_1_conv[0...

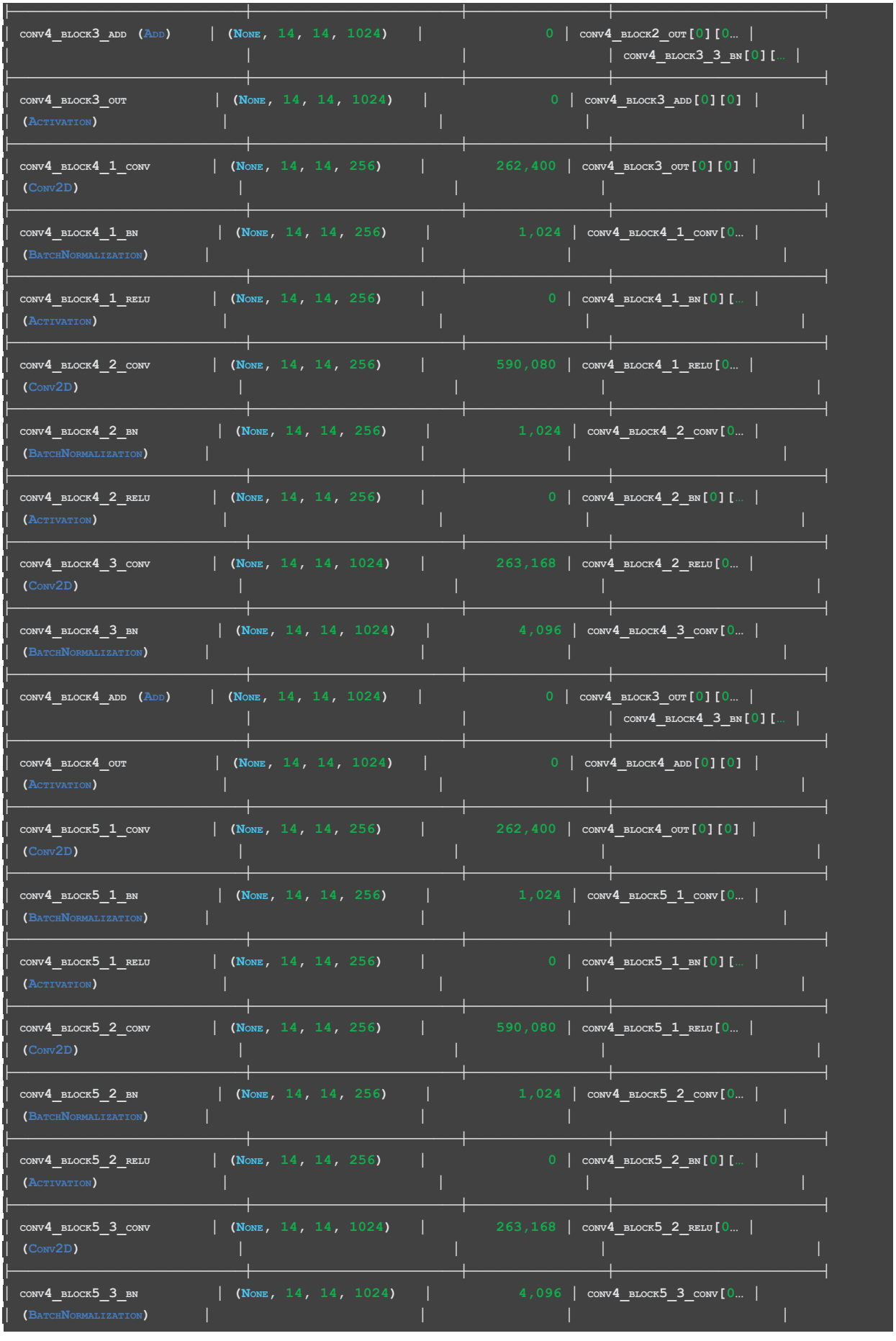
```
| conv2_block3_1_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block3_1_bn[0][... |
| conv2_block3_2_conv (Conv2D) | (None, 56, 56, 64) | 36,928 | conv2_block3_1_relu[0... |
| conv2_block3_2_bn (BatchNormalization) | (None, 56, 56, 64) | 256 | conv2_block3_2_conv[0... |
| conv2_block3_2_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block3_2_bn[0][... |
| conv2_block3_3_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | conv2_block3_2_relu[0... |
| conv2_block3_3_bn (BatchNormalization) | (None, 56, 56, 256) | 1,024 | conv2_block3_3_conv[0... |
| conv2_block3_add (Add) | (None, 56, 56, 256) | 0 | conv2_block2_out[0][0... |
| | | | conv2_block3_3_bn[0][... |
| conv2_block3_out (Activation) | (None, 56, 56, 256) | 0 | conv2_block3_add[0][0] |
| conv3_block1_1_conv (Conv2D) | (None, 28, 28, 128) | 32,896 | conv2_block3_out[0][0] |
| conv3_block1_1_bn (BatchNormalization) | (None, 28, 28, 128) | 512 | conv3_block1_1_conv[0... |
| conv3_block1_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block1_1_bn[0][... |
| conv3_block1_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block1_1_relu[0... |
| conv3_block1_2_bn (BatchNormalization) | (None, 28, 28, 128) | 512 | conv3_block1_2_conv[0... |
| conv3_block1_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block1_2_bn[0][... |
| conv3_block1_0_conv (Conv2D) | (None, 28, 28, 512) | 131,584 | conv2_block3_out[0][0] |
| conv3_block1_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block1_2_relu[0... |
| conv3_block1_0_bn (BatchNormalization) | (None, 28, 28, 512) | 2,048 | conv3_block1_0_conv[0... |
| conv3_block1_3_bn (BatchNormalization) | (None, 28, 28, 512) | 2,048 | conv3_block1_3_conv[0... |
| conv3_block1_add (Add) | (None, 28, 28, 512) | 0 | conv3_block1_0_bn[0][... |
| | | | conv3_block1_3_bn[0][... |
| conv3_block1_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block1_add[0][0] |
```

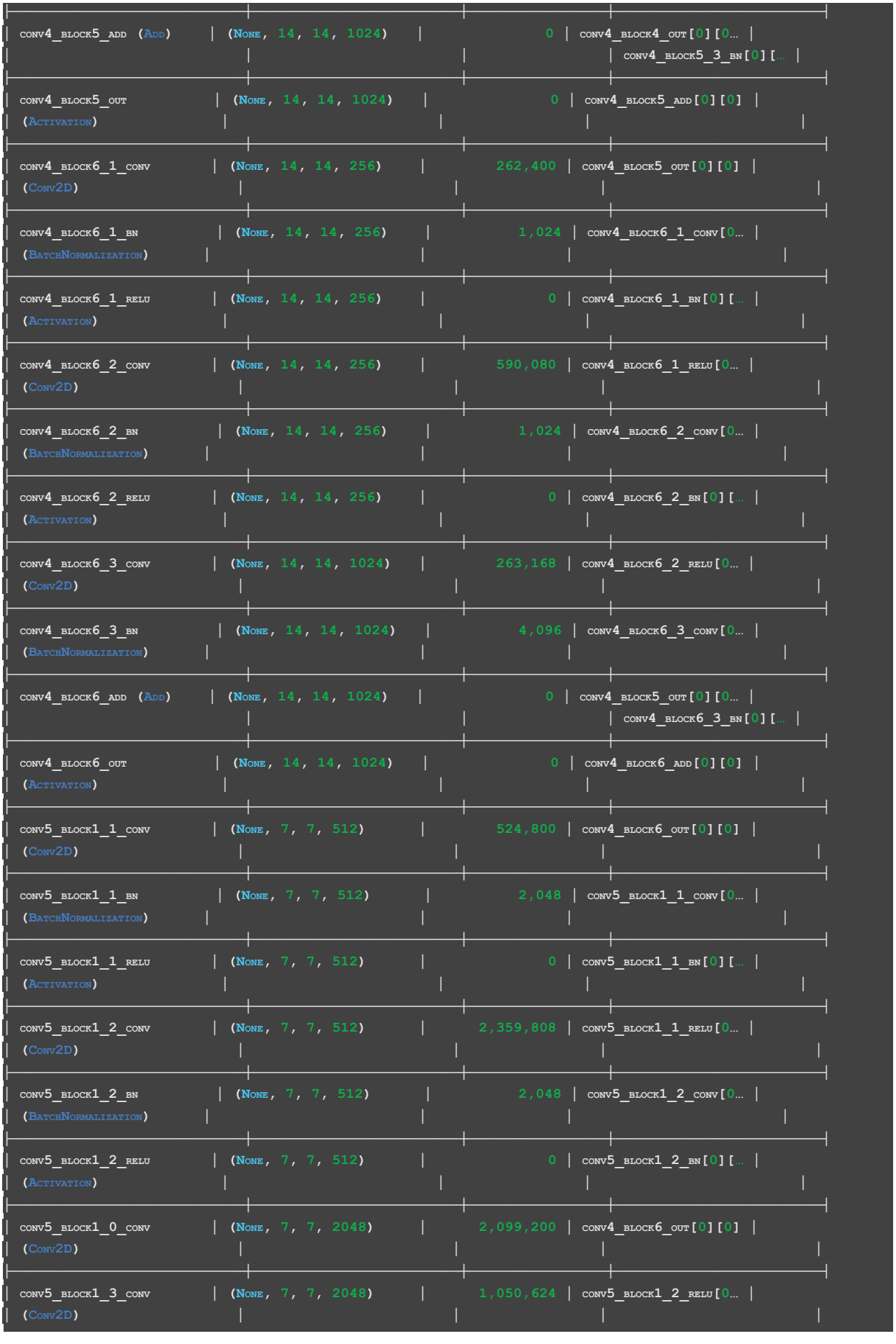

conv3_block2_1_conv (Conv2D) (None, 28, 28, 128) 65,664 conv3_block1_out[0][0]
conv3_block2_1_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block2_1_conv[0...
conv3_block2_1_relu (Activation) (None, 28, 28, 128) 0 conv3_block2_1_bn[0][...
conv3_block2_2_conv (Conv2D) (None, 28, 28, 128) 147,584 conv3_block2_1_relu[0...
conv3_block2_2_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block2_2_conv[0...
conv3_block2_2_relu (Activation) (None, 28, 28, 128) 0 conv3_block2_2_bn[0][...
conv3_block2_3_conv (Conv2D) (None, 28, 28, 512) 66,048 conv3_block2_2_relu[0...
conv3_block2_3_bn (BatchNormalization) (None, 28, 28, 512) 2,048 conv3_block2_3_conv[0...
conv3_block2_add (Add) (None, 28, 28, 512) 0 conv3_block1_out[0][0...
conv3_block2_3_bn[0][...
conv3_block2_out (Activation) (None, 28, 28, 512) 0 conv3_block2_add[0][0]
conv3_block3_1_conv (Conv2D) (None, 28, 28, 128) 65,664 conv3_block2_out[0][0]
conv3_block3_1_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block3_1_conv[0...
conv3_block3_1_relu (Activation) (None, 28, 28, 128) 0 conv3_block3_1_bn[0][...
conv3_block3_2_conv (Conv2D) (None, 28, 28, 128) 147,584 conv3_block3_1_relu[0...
conv3_block3_2_bn (BatchNormalization) (None, 28, 28, 128) 512 conv3_block3_2_conv[0...
conv3_block3_2_relu (Activation) (None, 28, 28, 128) 0 conv3_block3_2_bn[0][...
conv3_block3_3_conv (Conv2D) (None, 28, 28, 512) 66,048 conv3_block3_2_relu[0...
conv3_block3_3_bn (BatchNormalization) (None, 28, 28, 512) 2,048 conv3_block3_3_conv[0...
conv3_block3_add (Add) (None, 28, 28, 512) 0 conv3_block2_out[0][0...
conv3_block3_3_bn[0][...
conv3_block3_out (Activation) (None, 28, 28, 512) 0 conv3_block3_add[0][0]



conv3_block4_1_conv (Conv2D)	(None, 28, 28, 128)	65,664	conv3_block3_out[0][0]
conv3_block4_1_bn (BatchNormalization)	(None, 28, 28, 128)	512	conv3_block4_1_conv[0...
conv3_block4_1_relu (Activation)	(None, 28, 28, 128)	0	conv3_block4_1_bn[0][...
conv3_block4_2_conv (Conv2D)	(None, 28, 28, 128)	147,584	conv3_block4_1_relu[0...
conv3_block4_2_bn (BatchNormalization)	(None, 28, 28, 128)	512	conv3_block4_2_conv[0...
conv3_block4_2_relu (Activation)	(None, 28, 28, 128)	0	conv3_block4_2_bn[0][...
conv3_block4_3_conv (Conv2D)	(None, 28, 28, 512)	66,048	conv3_block4_2_relu[0...
conv3_block4_3_bn (BatchNormalization)	(None, 28, 28, 512)	2,048	conv3_block4_3_conv[0...
conv3_block4_add (Add)	(None, 28, 28, 512)	0	conv3_block3_out[0][0... conv3_block4_3_bn[0][...
conv3_block4_out (Activation)	(None, 28, 28, 512)	0	conv3_block4_add[0][0]
conv4_block1_1_conv (Conv2D)	(None, 14, 14, 256)	131,328	conv3_block4_out[0][0]
conv4_block1_1_bn (BatchNormalization)	(None, 14, 14, 256)	1,024	conv4_block1_1_conv[0...
conv4_block1_1_relu (Activation)	(None, 14, 14, 256)	0	conv4_block1_1_bn[0][...
conv4_block1_2_conv (Conv2D)	(None, 14, 14, 256)	590,080	conv4_block1_1_relu[0...
conv4_block1_2_bn (BatchNormalization)	(None, 14, 14, 256)	1,024	conv4_block1_2_conv[0...
conv4_block1_2_relu (Activation)	(None, 14, 14, 256)	0	conv4_block1_2_bn[0][...
conv4_block1_0_conv (Conv2D)	(None, 14, 14, 1024)	525,312	conv3_block4_out[0][0]
conv4_block1_3_conv (Conv2D)	(None, 14, 14, 1024)	263,168	conv4_block1_2_relu[0...
conv4_block1_0_bn (BatchNormalization)	(None, 14, 14, 1024)	4,096	conv4_block1_0_conv[0...
conv4_block1_3_bn (BatchNormalization)	(None, 14, 14, 1024)	4,096	conv4_block1_3_conv[0...

```
┃ conv4_block1_add (Add)      ┃ (None, 14, 14, 1024)   ┃             0 ┃ conv4_block1_0_bn[0][… ┃
┃                             ┃                        ┃               ┃ conv4_block1_3_bn[0][… ┃
┃ conv4_block1_out            ┃ (None, 14, 14, 1024)   ┃             0 ┃ conv4_block1_add[0][0] ┃
┃ (Activation)                ┃                        ┃               ┃                        ┃
┃ conv4_block2_1_conv         ┃ (None, 14, 14, 256)    ┃       262,400 ┃ conv4_block1_out[0][0] ┃
┃ (Conv2D)                    ┃                        ┃               ┃                        ┃
┃ conv4_block2_1_bn           ┃ (None, 14, 14, 256)    ┃         1,024 ┃ conv4_block2_1_conv[0… ┃
┃ (BatchNormalization)        ┃                        ┃               ┃                        ┃
┃ conv4_block2_1_relu         ┃ (None, 14, 14, 256)    ┃             0 ┃ conv4_block2_1_bn[0][… ┃
┃ (Activation)                ┃                        ┃               ┃                        ┃
┃ conv4_block2_2_conv         ┃ (None, 14, 14, 256)    ┃       590,080 ┃ conv4_block2_1_relu[0… ┃
┃ (Conv2D)                    ┃                        ┃               ┃                        ┃
┃ conv4_block2_2_bn           ┃ (None, 14, 14, 256)    ┃         1,024 ┃ conv4_block2_2_conv[0… ┃
┃ (BatchNormalization)        ┃                        ┃               ┃                        ┃
┃ conv4_block2_2_relu         ┃ (None, 14, 14, 256)    ┃             0 ┃ conv4_block2_2_bn[0][… ┃
┃ (Activation)                ┃                        ┃               ┃                        ┃
┃ conv4_block2_3_conv         ┃ (None, 14, 14, 1024)   ┃       263,168 ┃ conv4_block2_2_relu[0… ┃
┃ (Conv2D)                    ┃                        ┃               ┃                        ┃
┃ conv4_block2_3_bn           ┃ (None, 14, 14, 1024)   ┃         4,096 ┃ conv4_block2_3_conv[0… ┃
┃ (BatchNormalization)        ┃                        ┃               ┃                        ┃
┃ conv4_block2_add (Add)      ┃ (None, 14, 14, 1024)   ┃             0 ┃ conv4_block1_out[0][0… ┃
┃                             ┃                        ┃               ┃ conv4_block2_3_bn[0][… ┃
┃ conv4_block2_out            ┃ (None, 14, 14, 1024)   ┃             0 ┃ conv4_block2_add[0][0] ┃
┃ (Activation)                ┃                        ┃               ┃                        ┃
┃ conv4_block3_1_conv         ┃ (None, 14, 14, 256)    ┃       262,400 ┃ conv4_block2_out[0][0] ┃
┃ (Conv2D)                    ┃                        ┃               ┃                        ┃
┃ conv4_block3_1_bn           ┃ (None, 14, 14, 256)    ┃         1,024 ┃ conv4_block3_1_conv[0… ┃
┃ (BatchNormalization)        ┃                        ┃               ┃                        ┃
┃ conv4_block3_1_relu         ┃ (None, 14, 14, 256)    ┃             0 ┃ conv4_block3_1_bn[0][… ┃
┃ (Activation)                ┃                        ┃               ┃                        ┃
┃ conv4_block3_2_conv         ┃ (None, 14, 14, 256)    ┃       590,080 ┃ conv4_block3_1_relu[0… ┃
┃ (Conv2D)                    ┃                        ┃               ┃                        ┃
┃ conv4_block3_2_bn           ┃ (None, 14, 14, 256)    ┃         1,024 ┃ conv4_block3_2_conv[0… ┃
┃ (BatchNormalization)        ┃                        ┃               ┃                        ┃
┃ conv4_block3_2_relu         ┃ (None, 14, 14, 256)    ┃             0 ┃ conv4_block3_2_bn[0][… ┃
┃ (Activation)                ┃                        ┃               ┃                        ┃
┃ conv4_block3_3_conv         ┃ (None, 14, 14, 1024)   ┃       263,168 ┃ conv4_block3_2_relu[0… ┃
┃ (Conv2D)                    ┃                        ┃               ┃                        ┃
┃ conv4_block3_3_bn           ┃ (None, 14, 14, 1024)   ┃         4,096 ┃ conv4_block3_3_conv[0… ┃
┃ (BatchNormalization)        ┃                        ┃               ┃                        ┃
```


conv4_block3_add (Add) (None, 14, 14, 1024) 0 conv4_block2_out[0][0]... conv4_block3_3_bn[0][...
conv4_block3_out (Activation) (None, 14, 14, 1024) 0 conv4_block3_add[0][0]
conv4_block4_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block3_out[0][0]
conv4_block4_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block4_1_conv[0...
conv4_block4_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block4_1_bn[0][...
conv4_block4_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block4_1_relu[0...
conv4_block4_2_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block4_2_conv[0...
conv4_block4_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block4_2_bn[0][...
conv4_block4_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block4_2_relu[0...
conv4_block4_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block4_3_conv[0...
conv4_block4_add (Add) (None, 14, 14, 1024) 0 conv4_block3_out[0][0]... conv4_block4_3_bn[0][...
conv4_block4_out (Activation) (None, 14, 14, 1024) 0 conv4_block4_add[0][0]
conv4_block5_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block4_out[0][0]
conv4_block5_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block5_1_conv[0...
conv4_block5_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block5_1_bn[0][...
conv4_block5_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block5_1_relu[0...
conv4_block5_2_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block5_2_conv[0...
conv4_block5_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block5_2_bn[0][...
conv4_block5_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block5_2_relu[0...
conv4_block5_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block5_3_conv[0...


conv4_block5_add (Add) (None, 14, 14, 1024) 0 conv4_block4_out[0][0]... conv4_block5_3_bn[0][...
conv4_block5_out (Activation) (None, 14, 14, 1024) 0 conv4_block5_add[0][0]
conv4_block6_1_conv (Conv2D) (None, 14, 14, 256) 262,400 conv4_block5_out[0][0]
conv4_block6_1_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block6_1_conv[0...
conv4_block6_1_relu (Activation) (None, 14, 14, 256) 0 conv4_block6_1_bn[0][...
conv4_block6_2_conv (Conv2D) (None, 14, 14, 256) 590,080 conv4_block6_1_relu[0...
conv4_block6_2_bn (BatchNormalization) (None, 14, 14, 256) 1,024 conv4_block6_2_conv[0...
conv4_block6_2_relu (Activation) (None, 14, 14, 256) 0 conv4_block6_2_bn[0][...
conv4_block6_3_conv (Conv2D) (None, 14, 14, 1024) 263,168 conv4_block6_2_relu[0...
conv4_block6_3_bn (BatchNormalization) (None, 14, 14, 1024) 4,096 conv4_block6_3_conv[0...
conv4_block6_add (Add) (None, 14, 14, 1024) 0 conv4_block5_out[0][0]... conv4_block6_3_bn[0][...
conv4_block6_out (Activation) (None, 14, 14, 1024) 0 conv4_block6_add[0][0]
conv5_block1_1_conv (Conv2D) (None, 7, 7, 512) 524,800 conv4_block6_out[0][0]
conv5_block1_1_bn (BatchNormalization) (None, 7, 7, 512) 2,048 conv5_block1_1_conv[0...
conv5_block1_1_relu (Activation) (None, 7, 7, 512) 0 conv5_block1_1_bn[0][...
conv5_block1_2_conv (Conv2D) (None, 7, 7, 512) 2,359,808 conv5_block1_1_relu[0...
conv5_block1_2_bn (BatchNormalization) (None, 7, 7, 512) 2,048 conv5_block1_2_conv[0...
conv5_block1_2_relu (Activation) (None, 7, 7, 512) 0 conv5_block1_2_bn[0][...
conv5_block1_0_conv (Conv2D) (None, 7, 7, 2048) 2,099,200 conv4_block6_out[0][0]
conv5_block1_3_conv (Conv2D) (None, 7, 7, 2048) 1,050,624 conv5_block1_2_relu[0...

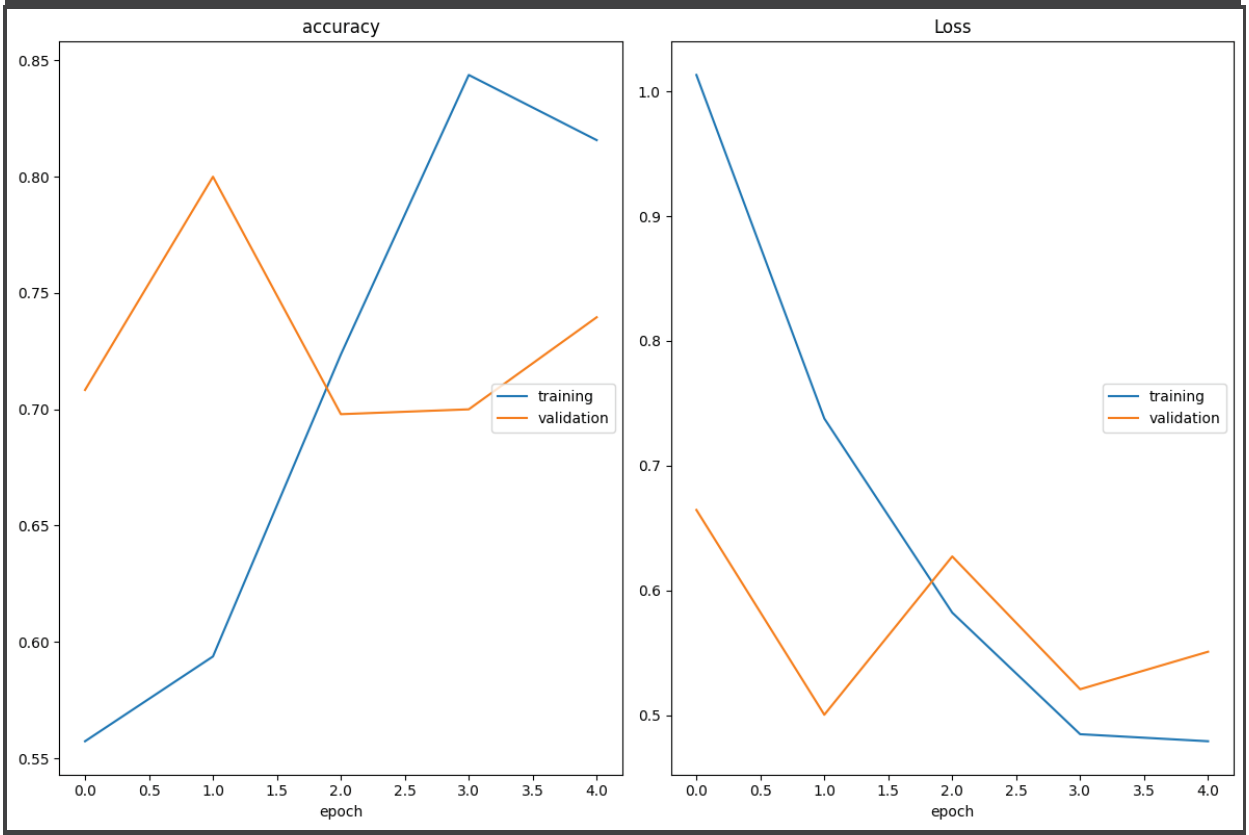
```
| conv5_block1_0_bn (BatchNormalization) | (None, 7, 7, 2048) | 8,192 | conv5_block1_0_conv[0... |
| conv5_block1_3_bn (BatchNormalization) | (None, 7, 7, 2048) | 8,192 | conv5_block1_3_conv[0... |
| conv5_block1_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block1_0_bn[0][... |
|  |  |  | conv5_block1_3_bn[0][... |
| conv5_block1_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block1_add[0][0] |
| conv5_block2_1_conv (Conv2D) | (None, 7, 7, 512) | 1,049,088 | conv5_block1_out[0][0] |
| conv5_block2_1_bn (BatchNormalization) | (None, 7, 7, 512) | 2,048 | conv5_block2_1_conv[0... |
| conv5_block2_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block2_1_bn[0][... |
| conv5_block2_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block2_1_relu[0... |
| conv5_block2_2_bn (BatchNormalization) | (None, 7, 7, 512) | 2,048 | conv5_block2_2_conv[0... |
| conv5_block2_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block2_2_bn[0][... |
| conv5_block2_3_conv (Conv2D) | (None, 7, 7, 2048) | 1,050,624 | conv5_block2_2_relu[0... |
| conv5_block2_3_bn (BatchNormalization) | (None, 7, 7, 2048) | 8,192 | conv5_block2_3_conv[0... |
| conv5_block2_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block1_out[0][0... |
|  |  |  | conv5_block2_3_bn[0][... |
| conv5_block2_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block2_add[0][0] |
| conv5_block3_1_conv (Conv2D) | (None, 7, 7, 512) | 1,049,088 | conv5_block2_out[0][0] |
| conv5_block3_1_bn (BatchNormalization) | (None, 7, 7, 512) | 2,048 | conv5_block3_1_conv[0... |
| conv5_block3_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block3_1_bn[0][... |
| conv5_block3_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block3_1_relu[0... |
| conv5_block3_2_bn (BatchNormalization) | (None, 7, 7, 512) | 2,048 | conv5_block3_2_conv[0... |
| conv5_block3_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block3_2_bn[0][... |
```

```
| conv5_block3_3_conv (Conv2D)        | (None, 7, 7, 2048)        |       1,050,624 | conv5_block3_2_relu[0... |
| conv5_block3_3_bn (BatchNormalization) | (None, 7, 7, 2048)     |           8,192 | conv5_block3_3_conv[0... |
| conv5_block3_add (Add)              | (None, 7, 7, 2048)        |               0 | conv5_block2_out[0][0... |
|                                     |                           |                 | conv5_block3_3_bn[0][... |
| conv5_block3_out (Activation)       | (None, 7, 7, 2048)        |               0 | conv5_block3_add[0][0] |
| average_pooling2d_1 (AveragePooling2D) | (None, 1, 1, 2048)     |               0 | conv5_block3_out[0][0] |
| flatten_4 (Flatten)                 | (None, 2048)              |               0 | average_pooling2d_1[0... |
| dense_16 (Dense)                    | (None, 1024)              |       2,098,176 | flatten_4[0][0]          |
| dropout (Dropout)                   | (None, 1024)              |               0 | dense_16[0][0]           |
| dense_17 (Dense)                    | (None, 4)                 |           4,100 | dropout[0][0]            |

Total params: 25,689,988 (98.00 MB)

Trainable params: 2,102,276 (8.02 MB)

Non-trainable params: 23,587,712 (89.98 MB)
```

Treinamento

```
%%time

from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from livelossplot import PlotLossesKeras

steps_per_epoch = traingen.samples // BATCH_SIZE
val_steps = validgen.samples // BATCH_SIZE

n_epochs = 10

optimizer = RMSprop(learning_rate=0.0001)

model_ft.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_ft.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',
```

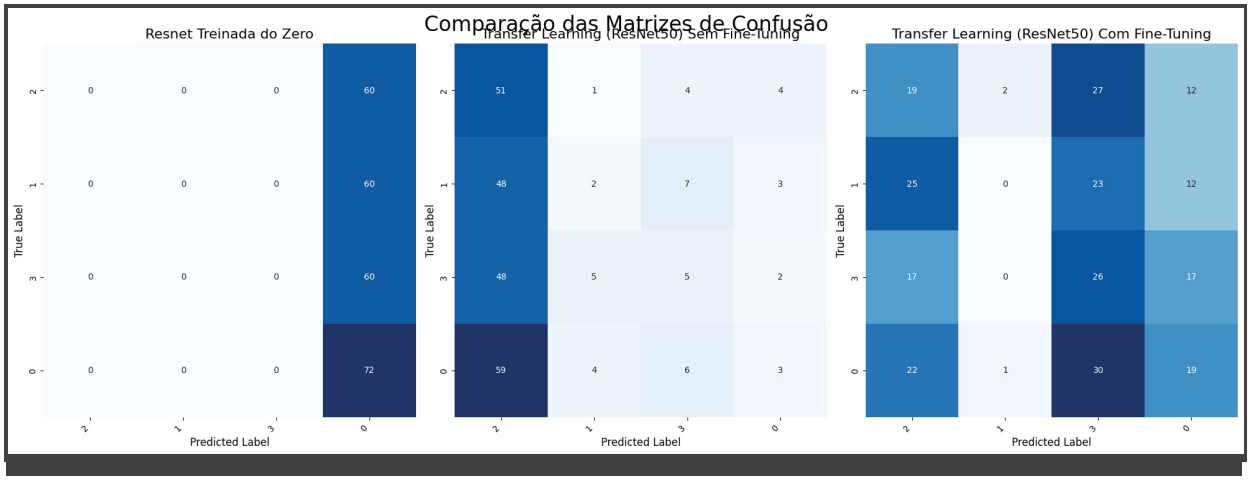
```
                                    patience=10,
                                    restore_best_weights=True,
                                    mode='min')

# Treinamento do Modelo
history_ft = model_ft.fit(traingen,
                          epochs=n_epochs,
                          steps_per_epoch=steps_per_epoch,
                          validation_data=validgen,
                          validation_steps=val_steps,
                          callbacks=[checkpointer, PlotLossesKeras()],
                          #callbacks=[early_stop, checkpointer, PlotLossesKeras()],
                          verbose=False)
```




```
accuracy
        training                (min:    0.557, max:    0.844, cur:    0.816)
        validation              (min:    0.698, max:    0.800, cur:    0.740)
Loss
        training                (min:    0.479, max:    1.013, cur:    0.479)
        validation              (min:    0.500, max:    0.665, cur:    0.551)
CPU times: user 8min 15s, sys: 28.5 s, total: 8min 44s
Wall time: 6min 19s
```

```
- Predições dos modelos


from sklearn.metrics import accuracy_score

# Generate predictions
model_0.load_weights('img_model_0.weights.best.keras') # initialize the best trained weights

model_tl.load_weights('img_model_tl.weights.best.keras') # initialize the best trained weights

model_ft.load_weights('img_model_ft.weights.best.keras') # initialize the best trained weights

true_classes = testgen.classes
class_indices = traingen.class_indices
class_indices = dict((v,k) for k,v in class_indices.items())

preds_0 = model_0.predict(testgen)
pred_classes_0 = np.argmax(preds_0, axis=1)

preds_tl = model_tl.predict(testgen)
pred_classes_tl = np.argmax(preds_tl, axis=1)

preds_ft = model_ft.predict(testgen)
pred_classes_ft = np.argmax(preds_ft, axis=1)

acc_0 = accuracy_score(true_classes, pred_classes_0)
print("Acurácia Modelo ResNet50 treinado do zero: {:.2f}%".format(acc_0 * 100))

acc_tl = accuracy_score(true_classes, pred_classes_tl)
print("Acurácia Modelo ResNet50 com Transfer Learning sem Fine-Tuning: {:.2f}%".format(acc_tl * 100))

acc_ft = accuracy_score(true_classes, pred_classes_ft)
print("Acurácia Modelo ResNet50 com com Transfer Learning e Fine-Tuning: {:.2f}%".format(acc_ft * 100))



/usr/local/lib/python3.10/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()
4/4 ━━━━━━━━━━━━━━━━━━━━ 47s 11s/step
4/4 ━━━━━━━━━━━━━━━━━━━━ 47s 11s/step
4/4 ━━━━━━━━━━━━━━━━━━━━ 46s 11s/step
Acurácia Modelo ResNet50 treinado do zero: 28.57%
Acurácia Modelo ResNet50 com Transfer Learning sem Fine-Tuning: 24.21%
Acurácia Modelo ResNet50 com com Transfer Learning e Fine-Tuning: 25.40%

Resultados

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

```
# Get the names of the ten classes
class_names = testgen.class_indices.keys()

print(f'Classes de Teste: {class_names}')



Classes de Teste: dict_keys(['2', '1', '3', '0'])



import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score



# Get the names of the ten classes
class_names = testgen.class_indices.keys()

def plot_heatmap(y_true, y_pred, class_names, ax, title):
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(
        cm,
        annot=True,
        square=True,
        xticklabels=class_names,
        yticklabels=class_names,
        fmt='d',
        cmap=plt.cm.Blues,
        cbar=False,
        ax=ax
    )
    ax.set_title(title, fontsize=16)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha="right")
    ax.set_ylabel('True Label', fontsize=12)
    ax.set_xlabel('Predicted Label', fontsize=12)

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(20, 10))

plot_heatmap(true_classes, pred_classes_0, class_names, ax1, title="Resnet Treinada do Zero")
plot_heatmap(true_classes, pred_classes_tl, class_names, ax2, title="Transfer Learning (ResNet50) Sem
Fine-Tuning")
plot_heatmap(true_classes, pred_classes_ft, class_names, ax3, title="Transfer Learning (ResNet50) Com
Fine-Tuning")

fig.suptitle("Comparação das Matrizes de Confusão", fontsize=24)
fig.tight_layout()
fig.subplots_adjust(top=1.25)
plt.show()
```




```
VGG 16

1. Importação das bibliotecas necessárias

import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Flatten, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
import pathlib



2. Configuração dos caminhos e parâmetros

# Caminhos para as pastas de treino, validação e teste
train_dir = '/content/Train_Warwick/Train_4cls_amostra/80'   # Treino
val_dir = '/content/Train_Warwick/Train_4cls_amostra/20'     # Validação
test_dir = '/content/Test_Warwick/Test_4cl_amostra/'         # Teste

# Parâmetros
IMG_SIZE = 224
BATCH_SIZE = 32
NUM_CLASSES = 4
EPOCHS = 10

3. DESNECESSÁRIO - Criação do Data Augmentation e geradores de imagem

# Data Augmentation para treino
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
```

```
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Data Augmentation simples para validação (apenas rescale)
val_datagen = ImageDataGenerator(rescale=1./255)

# Gerador de treino
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    seed=42
)

# Gerador de validação
val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,
    seed=42
)


4. Criação do modelo VGG16 com Transfer Learning

# Carregando a VGG16 pré-treinada no ImageNet, sem a última camada
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))

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

# Adicionando novas camadas Fully Connected
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
output = Dense(NUM_CLASSES, activation='softmax')(x)

# Criando o modelo final
model = Model(inputs=base_model.input, outputs=output)

Downloading data from
https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
```

```
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step


5. Compilação do modelo

model.compile(optimizer=Adam(learning_rate=0.0001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])



6. Configuração de Early Stopping para evitar overfitting

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)



7. Treinamento do modelo com Data Augmentation
history_with_aug = model.fit(
    traingen,
    validation_data=validgen,
    epochs=EPOCHS,
    callbacks=[early_stopping]
)



8. Treinamento do modelo sem Data Augmentation
# Criando um novo gerador de treino sem Data Augmentation, apenas rescale
traingen_no_aug = ImageDataGenerator(rescale=1./255)

traingen_no_aug = traingen_no_aug.flow_from_directory(
    train_dir,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    seed=42
)

# Treinando o modelo
history_without_aug = model.fit(
    traingen_no_aug,
    validation_data=validgen,
    epochs=EPOCHS,
    callbacks=[early_stopping]
)



Found 477 images belonging to 4 classes.
Epoch 1/5
/usr/local/lib/python3.10/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()
15/15 ━━━━━━━━━━━━━━━━━━━━ 329s 22s/step - accuracy: 0.4807 - loss: 1.3862 -
val_accuracy: 0.5690 - val_loss: 0.8669
```

```
Epoch 2/5
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 327s 22s/step - accuracy: 0.6081 - loss: 1.2499 -
val_accuracy: 0.5259 - val_loss: 1.0271
Epoch 3/5
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 328s 22s/step - accuracy: 0.7621 - loss: 1.1225 -
val_accuracy: 0.5603 - val_loss: 1.0977
Epoch 4/5
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 327s 22s/step - accuracy: 0.7776 - loss: 1.0144 -
val_accuracy: 0.4914 - val_loss: 1.2848
Epoch 5/5
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 382s 22s/step - accuracy: 0.8496 - loss: 0.9454 -
val_accuracy: 0.4914 - val_loss: 1.3202


9. DESNECESSÁRIO Configuração do gerador de imagens para a base de teste

# Caminho para a base de teste
test_dir_20 = '/content/Test_Warwick/Test_4cl_amostra/20'
test_dir_80 = '/content/Test_Warwick/Test_4cl_amostra/80'

# Gerador de teste com apenas rescale (sem Data Augmentation)
testgen_no_aug = ImageDataGenerator(rescale=1./255)

# Gerador de teste para as pastas 20 e 80
testgen_20 = testgen.flow_from_directory(
    test_dir_20,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)

testgen_no_aug = testgen.flow_from_directory(
    test_dir_80,
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)
```

c)

```
10 - Aplicação dos modelos treinados nas imagens de teste

# Avaliando o modelo com Data Augmentation na base de teste 20
#test_loss_20, test_accuracy_20 = model.evaluate(testgen_20)
#print(f"Acurácia na base de teste (20): {test_accuracy_20 * 100:.2f}%")

# Avaliando o modelo com Data Augmentation na base de teste 80
```

```
test_loss_80, test_accuracy_80 = model.evaluate(testgen)
print(f"Acurácia na base de teste (80): {test_accuracy_80 * 100:.2f}%")

4/4 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 141s 35s/step - accuracy: 0.1467 - loss: 3.9405
Acurácia na base de teste (80): 13.89%

11. Previsões nas imagens da base de teste

# Previsões para a base de teste 20
#predictions_20 = model.predict(testgen_20)
#predicted_classes_20 = predictions_20.argmax(axis=1)

# Previsões para a base de teste 80
predictions_80 = model.predict(testgen)
predicted_classes_80 = predictions_80.argmax(axis=1)

4/4 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 167s 44s/step
```

d)

```
- 12. Cálculo das métricas de desempenho

from sklearn.metrics import classification_report, confusion_matrix

# Classes verdadeiras para a base de teste 20
#true_classes_20 = testgen_20.classes

# Classes verdadeiras para a base de teste 80
true_classes_80 = testgen.classes

# Relatório de classificação e matriz de confusão para a base 20
#print("Métricas para a base de teste 20:")
#print(classification_report(true_classes_20, predicted_classes_20, target_names=testgen_20.class_indices.keys()))
#print("Matriz de confusão:")
#print(confusion_matrix(true_classes_20, predicted_classes_20))

# Relatório de classificação e matriz de confusão para a base 80
print("\nMétricas para a base de teste 80:")
print(classification_report(true_classes_80, predicted_classes_80, target_names=testgen.class_indices.keys()))
print("Matriz de confusão:")
print(confusion_matrix(true_classes_80, predicted_classes_80))



Métricas para a base de teste 80:
              precision    recall  f1-score   support

           2       0.18      0.40      0.25        60
           1       0.31      0.15      0.20        60
           3       0.27      0.07      0.11        60
```

```
              0       0.32      0.35      0.33        72

    accuracy                           0.25       252
   macro avg       0.27      0.24      0.22       252
weighted avg       0.27      0.25      0.23       252

Matriz de confusão:
[[24  8  3 25]
 [31  9  5 15]
 [36  7  4 13]
 [39  5  3 25]]
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report, confusion_matrix, f1_score

# Calcular as métricas Sensibilidade e Especificidade
def calculate_metrics(y_true, y_pred, class_names):
    cm = confusion_matrix(y_true, y_pred)

    # Sensibilidade (ou Recall)
    sensitivity = cm.diagonal() / cm.sum(axis=1)  # True Positive / (True Positive + False Negative)

    # Especificidade
    specificity = []
    for i in range(len(class_names)):
        tn = cm.sum() - (cm[i, :].sum() + cm[:, i].sum() - cm[i, i])  # True Negative
        specificity.append(tn / (tn + cm[:, i].sum() - cm[i, i]))  # True Negative / (True Negative + False
Positive)

    return cm, sensitivity, specificity

# Obter os nomes das classes
class_names = sorted(testgen.class_indices.keys())  # Garantir que os nomes das classes estejam ordenados
num_classes = len(class_names)

# Classes verdadeiras e previstas para a base de teste 80 para VGG16
true_classes_80 = testgen.classes
predicted_classes_vgg16_80 = predicted_classes_80  # Certifique-se de que predicted_classes_80 contenha as previsões
da VGG16 para a base 80

# Calcular e exibir as métricas
def display_metrics(y_true, y_pred, model_name):
    # Calcular matriz de confusão, sensibilidade e especificidade
    cm, sensitivity, specificity = calculate_metrics(y_true, y_pred, class_names)

    # Gerar o relatório de classificação com zero_division ajustado para evitar avisos
    report = classification_report(y_true, y_pred, target_names=class_names, digits=2, zero_division=0)

    print(f"\nMétricas para {model_name}:")
    print(report)

    # Exibir a matriz de confusão
```

```
    print("Matriz de confusão:")
    print(cm)

# Exibir os resultados para ResNet50 e VGG16
display_metrics(true_classes, pred_classes_0, "ResNet Treinada do Zero")
display_metrics(true_classes, pred_classes_tl, "Transfer Learning (ResNet50) Sem Fine-Tuning")
display_metrics(true_classes, pred_classes_ft, "Transfer Learning (ResNet50) Com Fine-Tuning")
display_metrics(true_classes_80, predicted_classes_vgg16_80, "Transfer Learning (VGG16) na Base 80")

Métricas para ResNet Treinada do Zero:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        60
           1       0.00      0.00      0.00        60
           2       0.00      0.00      0.00        60
           3       0.29      1.00      0.44        72

    accuracy                           0.29       252
   macro avg       0.07      0.25      0.11       252
weighted avg       0.08      0.29      0.13       252

Matriz de confusão:
[[ 0  0  0 60]
 [ 0  0  0 60]
 [ 0  0  0 60]
 [ 0  0  0 72]]

Métricas para Transfer Learning (ResNet50) Sem Fine-Tuning:
              precision    recall  f1-score   support

           0       0.25      0.85      0.38        60
           1       0.17      0.03      0.06        60
           2       0.23      0.08      0.12        60
           3       0.25      0.04      0.07        72

    accuracy                           0.24       252
   macro avg       0.22      0.25      0.16       252
weighted avg       0.22      0.24      0.15       252

Matriz de confusão:
[[51  1  4  4]
 [48  2  7  3]
 [48  5  5  2]
 [59  4  6  3]]

Métricas para Transfer Learning (ResNet50) Com Fine-Tuning:
              precision    recall  f1-score   support

           0       0.23      0.32      0.27        60
           1       0.00      0.00      0.00        60
           2       0.25      0.43      0.31        60
           3       0.32      0.26      0.29        72
```

```
    accuracy                           0.25       252
   macro avg       0.20      0.25      0.22       252
weighted avg       0.20      0.25      0.22       252

Matriz de confusão:
[[19  2 27 12]
 [25  0 23 12]
 [17  0 26 17]
 [22  1 30 19]]

Métricas para Transfer Learning (VGG16) na Base 80:
              precision    recall  f1-score   support

           0       0.18      0.40      0.25        60
           1       0.31      0.15      0.20        60
           2       0.27      0.07      0.11        60
           3       0.32      0.35      0.33        72

    accuracy                           0.25       252
   macro avg       0.27      0.24      0.22       252
weighted avg       0.27      0.25      0.23       252

Matriz de confusão:
[[24  8  3 25]
 [31  9  5 15]
 [36  7  4 13]
 [39  5  3 25]]
```

e)

Para determinar o melhor modelo, foi utilizado o F1-Score ponderado, que equilibra precisão e recall, sendo adequado para cenários com classes desbalanceadas.

Resultados

1- ResNet Treinada do Zero: F1-Score ponderado: 0.127

2- Transfer Learning (ResNet50) Sem Fine-Tuning: F1-Score ponderado: 0.154

3- Transfer Learning (ResNet50) Com Fine-Tuning: F1-Score ponderado: 0.220

4- Transfer Learning (VGG16): F1-Score ponderado: 0.229

Conclusão

O modelo VGG16 apresentou o maior F1-Score ponderado (0.229), indicando o melhor desempenho geral dentre os avaliados.

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

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

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

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

B – RESOLUÇÃO

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

ESTUDO DE CASO: IMPLICAÇÕES ÉTICAS DO USO DO CHATGPT

Artigo apresentado como requisito parcial à conclusão da Especialização em Inteligência Artificial Aplicada, disciplina de Aspectos Filosóficos e Éticos da IA, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná.

Professor: Dr. Marcos Wachowicz

“Assim que Pandora abriu a caixa que deveria manter fechada, males incontáveis saíram dela e começaram a se espalhar descontroladamente pelo mundo.”

Mitologia Grega

RESUMO

O ChatGPT, um chatbot de inteligência artificial desenvolvido pela OpenAI, apresenta implicações éticas diversas que precisam ser consideradas. Este trabalho explora essas implicações em diferentes contextos, como a disseminação de informações, a privacidade dos usuários e a tomada de decisões éticas. Através da análise crítica de estudos de caso específicos, o trabalho identifica dilemas éticos relevantes e propõe soluções responsáveis para lidar com esses desafios.

Palavras-chave: chatGPT; chatbot; inteligência artificial; implicações; éticas; direitos; privacidade.

ABSTRACT

ChatGPT, an artificial intelligence chatbot developed by OpenAI, has several ethical implications that need to be considered. This work explores these implications in different contexts, such as information dissemination, user privacy, and ethical decision-making. Through a critical analysis of specific case studies, the work identifies relevant ethical dilemmas and proposes responsible solutions to deal with these challenges.

Keywords: chatGPT; chatbot; artificial intelligence; implications; ethics; rights;

1 INTRODUÇÃO

Toda nova tecnologia traz consigo potenciais benefícios e seus respectivos riscos, tomando-se como exemplo a comodidade trazida pelos automóveis, eletricidade e aviões, note-se que não havia acidentes de carro até a invenção do automóvel, eletrocussões antes da utilização da eletricidade e nem acidentes de avião antes destes serem desenvolvidos, não esquecendo, é claro a energia nuclear, o exemplo mais pungente desta dicotomia tecnológica; com a Inteligência Artificial não poderia ser diferente.

As principais implicações éticas a serem consideradas no tocante à utilização de modelos de Inteligência Artificial (IA), mais especificamente o modelo chatGPT, são aquelas referentes aos direitos fundamentais, em particular, à privacidade e intimidade, julgamento justo pelo sistema de justiça, educação, comunicação e trabalho.

2 DESENVOLVIMENTO

2.1 IMPLICAÇOES ÉTICAS

Privacidade

A invasão de Privacidade dos usuários por uma tecnologia que tende a permear muitas de suas atividades, considerando-se a tendência atual de incorporação da IA como um diferencial competitivo de negócio, afigura-se como a principal causa de preocupação. É inquietante imaginar o nível de conhecimento que os programas terão de seus usuários quando estiverem incorporados aos sistemas operacionais de todos os dispositivos, no caso do Microsoft Windows, esse processo já se encontra em curso com a funcionalidade “recall” disponível em algumas máquinas com coplilot PC, que possuem integração com o chatgpt: “A ferramenta permitirá que os usuários revejam tudo o que foi feito naquele computador, incluindo sites e programas acessados. “(TECMUNDO, 2024).

Nesse sentido, a incorporação do GPT a outros sistemas e dispositivos que podem torna-lo ubíquo e permitir acessos a inúmeras bases de dados, como as de hospitais e laboratórios médicos, dados genéticos, governamentais (dos três poderes), fiscais e bancários, levanta mais algumas questões éticas relevantes: Falta de transparência sobre quem possui os dados, falta de consentimento para coleta e uso contínuo de dados, falta de acesso dos consumidores aos dados brutos, risco de hackeamento de dados.

Outra preocupação relevante diz respeito à incorporação de dados sensíveis às bases das IA’s ininterruptamente, pois ao interagir os usuários

fornecem mais informações do que pretendem e os termos e condições não costumam ser claros a esse respeito, muitas vezes deliberadamente. Acrescente-se a isso a captura de dados em tempo real via celular, carros, câmeras, dispositivos “vestíveis” - wearables (NIH, 2024) e Iot - Internet das Coisas.

Sistema de Justiça

No âmbito do Poder Judiciário, a utilização de modelos de IA suscita alguns questionamentos sobre suas implicações éticas, a saber:

O potencial desaparecimento do juiz natural a partir da utilização dos sistemas para fundamentação das decisões, tendo em vista que não será possível determinar se o magistrado analisou profundamente o caso em questão ou apenas chancelou o trabalho da máquina. Conflito de Interesses quando a empresa que desenvolve o modelo estiver envolvida em demandas a serem julgadas pelo seu próprio produto. Comprometimento da credibilidade do sistema de justiça por decisões alucinadas geradas por LLM’s e assinadas por magistrados (MIGALHAS, 2024) ou por petições geradas por modelos e impetradas por advogados (CNN Brasil, 2024). A IA utilizando suas próprias decisões como jurisprudência: considerando-se que o aumento da fundamentação das sentenças por máquinas será progressivo, logo suas decisões serão adicionadas às bases de treino e teste e, em algum ponto no futuro, haverá mais decisões geradas por robôs do que por juízes e eles se alimentarão da própria jurisprudência. A esse respeito, observe-se o parecer da área técnica (Comissão Permanente de Tecnologia da Informação e Inovação) do CNJ sobre o Procedimento de Controle Administrativo PCA 0000416-89.2023.2.00.0000, que “postula a adoção de providências para proibir a utilização do ChatGPT na elaboração de atos processuais.”:

“(a) o uso do ChatGTP por juízes pode ser prejudicial, pois a ferramenta não consegue avaliar adequadamente parâmetros fáticos e suas respostas são falaciosas;

(b) juízes devem realizar suas próprias pesquisas em textos legais, doutrina e jurisprudência, sem transferir seu poder e dever de julgamento para um recurso tecnológico;

(c) todo cidadão tem direito ao julgamento por um juiz humano, investido do poder e dever de julgar processos;

(d) o CNJ deve frear a invasão da inteligência artificial no sistema de justiça, em razão dos riscos de privatização e desnacionalização tecnológica, com danos semelhantes aos causados pelos algoritmos de redes sociais.” (BRASIL. CNJ, 2023).

Educação

Na seara da educação é possível elencar diversos riscos, tais como, o ensino de informações incorretas, imprecisas, não verificáveis ou alucinações simplesmente, sobre estas é importante verificar a declaração do CEO do google sobre não existir solução até o momento (GOOGLE Discovery, 2023). Há também o risco de ocorrência de vieses oriundos dos dados de treinamento ou preconceitos dos profissionais envolvidos na criação e desenvolvimento do sistema. É possível ainda o comprometimento da capacidade cognitiva pela falta dos estímulos adequados à construção do raciocínio, possível condicionamento causado pela obtenção de respostas prontas sem a necessidade de esforço. Acrescentem-se a isso implicações no desenvolvimento social dos alunos, crianças principalmente, pela falta de contato com outros humanos, na possibilidade distópica de um home schooling automatizado. Ainda é importante questionar qual seria o propósito da introdução da IA na educação; barateamento de profissionais nos quais se investe pouco ou a precarização de um serviço para pessoas nas quais se investe menos ainda.

Comunicação

No campo da comunicação é possível vislumbrar não só riscos éticos potenciais, mas também reais, tendo em vista casos concretos já ocorridos. A formação de bolhas informacionais, fenômeno que isola pessoas em um ambiente virtual onde não há visões divergentes, apenas reforço daquilo que o sujeito já pensa, é um exemplo de problema que pode ser agravado pela Inteligência Artificial devido à quantidade de informações sobre as pessoas que os modelos tendem a acumular, o que tornará mais fácil alimentar os indivíduos apenas com aquilo que desejam ver, ouvir e saber; isso configura uma manipulação do sistema de recompensa do cérebro, circuito que processa a informação relacionada à sensação de prazer ou de satisfação.

A partir do ponto em que a IA tiver capacidade de gerar conteúdo totalmente realístico em tempo real, será cada vez mais difícil discernir os fatos da realidade artificial, aqui tem-se um risco gigantesco de desinformação e manipulação, considerando-se que, com muito menos poder, apenas semeando desinformação, foi possível influenciar referendos como no brexit (Facebook / Cambridge Analytica) e eleições presidenciais (Whatsapp / Telegram / Facebook / Youtube), como a norte americana em 2016 e a brasileira em 2018. Tais precedentes de ataques às instituições e ao próprio regime democrático devem ser tratados com a devida gravidade.

A mídia também será profundamente afetada, com risco de substituição das fontes tradicionais de informações pelas big techs, além da perda geral de credibilidade a incapacidade das pessoas de saber o que é verdade.

Não menos importante, o potencial destrutivo dos deep fakes é preocupante,

criando-se falas, imagens e vídeos indistinguíveis dos reais, reputações podem ser destruídas e todo tipo de bestialidade pode ser criado e inundar as redes.

Golpes e fraudes via internet, por sua vez, podem ganhar nova dimensão com a possibilidade de simulação de imagens, áudios e vídeos.

Outros

Cabe registrar também o problema do trabalho semi-escravo dos trabalhadores fantasmas, "Modern Day Slaves" (REAL Stories, 2024) responsáveis por moderação de conteúdo em redes sociais e treinamento - fine tunning, de modelos de IA mediante pagamento vil - Amazon Mechanical Turk, Facebook (Via Figure 8 e Accenture). Aqui se vislumbra o que seria um modelo de “uberização” total do mercado de trabalho, com pessoas disputando subempregos em tarefas por demanda, sob a pressão de uma reserva de mercado mundial.

Outros riscos relevantes são o uso militar da tecnologia, incorporação da IA em armas autônomas, inclusive robôs - Boston Dynamics Atlas, (FORBES Tech, 2023).

2.2 SOLUÇÕES

Regular as tecnologias é um imperativo para a preservação dos direitos fundamentais das pessoas, desde eletricidade a radioatividade, passando por televisão, rádio e medicamentos. De maneira que, para tratar as implicações éticas elencadas anteriormente, o quesito regulação deve se aplicar a todos os tópicos. Importante ressaltar que auto regulação não seria apropriada nesse caso pois não haveria garantia de transparência e sim o risco de formação de cartéis pelas big techs legislando em causa própria.

Privacidade

Mais especificamente, no que tange a privacidade, alguns países já dispõem de legislação específica, no caso do Brasil, tem-se a Lei Geral de Proteção de Dados Pessoais - LGPD Lei nº 13.709/2018 que “dispõe sobre o tratamento de dados pessoais, inclusive nos meios digitais, por pessoa natural ou por pessoa jurídica de direito público ou privado, com o objetivo de proteger os direitos fundamentais de liberdade e de privacidade e o livre desenvolvimento da personalidade da pessoa natural.” Tem-se também a ANPD, órgão central de interpretação da Lei Geral de Proteção de Dados, ao qual cabe estabelecer normas e diretrizes para a sua implementação, buscando zelar pela garantia do direito de

todos os brasileiros terem seus dados pessoais devidamente protegidos, segundo estabelece a Lei nº 13.853/2019.

Espera-se dessas autoridades que executem auditorias regulares das bases de dados utilizadas para IA e que busquem assegurar transparência no uso desses dados. As empresas, é importante que incorporem os princípios de propriedade de dados e consentimento, projetem notificações nas plataformas para alertar os consumidores quando os dados forem acessados por outras pessoas, forneçam acesso a dados brutos aos seus titulares, definam e apliquem padrões de segurança cibernética, considerem a privacidade no design.

Sistema de Justiça

Até que se tenha dimensão mais precisa dos impactos dos modelos no judiciário, a providência mais prudente é a proibição do uso de modelos de IA para fundamentar e resolver lides, conforme sugerido no supracitado parecer do CNJ:

"… os riscos mapeados não podem ser simplesmente aceitos e monitorados; é necessária a implementação de medidas de mitigação e compartilhamento de responsabilidades. Nesse sentido, sugerem-se as seguintes medidas de tratamento de riscos:

(a) proibição do uso de soluções de inteligência artificial baseadas em large language models (LLMs) que importem em automação de atividades decisórias;

(b) determinação para que implementações de uso de LLMs assegurem ao usuário humano a revisão e opção de escolha quanto ao aproveitamento de insumo fornecido pelo modelo;

(c) determinação para que, em cenários de uso corporativo, os órgãos do Poder Judiciário adotem providências para preservação de dados pessoais e informações sensíveis;

(d) determinação para que, em cenários de uso corporativo ou individual, os usuários sejam submetidos à capacitação formal, a fim de serem esclarecidos quanto aos riscos e estratégias de prevenção, incluindo construção de contextos e engenharia de prompt." (BRASIL. CNJ, 2023)

Também é essencial que se aprimorem as ferramentas de detecção de conteúdo gerado por IA, tal como um sistema antivírus.

Educação

No contexto da educação, pela importância e potencial impacto nas gerações futuras, é preciso bastante cautela ao se adotar inovações metodológicas, de forma que, a adoção da IA como ferramenta auxiliar e sob supervisão parece ser a opção mais adequada para o uso responsável. De forma complementar, a

responsabilização das plataformas por inadequações, fatos incorretos ou alucinações deve fazer parte das exigências para se permitir tal tecnologia.

Curiosamente, nota-se em alguns países, cujos sistemas educacionais servem como referência para o restante do mundo, a tendência de retorno aos métodos tradicionais de ensino – desinformatização:

“Em uma mudança surpreendente, a Suécia volta atrás em sua abordagem de educação digital e investe milhões em livros didáticos impressos. Decisão anunciada pela ministra da Educação, Lotta Edholm. A mudança de direção ocorre após décadas de esforços para incorporar tecnologia nas salas de aula suecas, incluindo a distribuição de tablets e laptops para os alunos. No entanto, a ministra Edholm destacou em um artigo publicado no jornal "Expressen" que houve uma postura acrítica em relação à tecnologia, negligenciando o conteúdo em favor da forma. Ela ressaltou que, apesar das vantagens dos recursos didáticos digitais, os livros físicos oferecem benefícios que as telas não podem substituir. (FORUM, 2023)

Comunicação

No âmbito da comunicação, mais uma vez a regulação é o passo mais importante no tratamento dos dilemas éticos causados pela IA. Destaque-se que isso deverá ser implementado em nível mundial, tendo em vista o alcance de tais empresas e a possibilidade de descumprimento de determinações judiciais, como já se viu em casos envolvendo whatsapp e twitter. Além da legislação específica, é necessário supervisão, auditoria, responsabilização de plataformas e pessoas e, principalmente, punição aos abusos.

3 CONCLUSÃO

Diante do exposto, conclui-se que a Inteligência Artificial, assim como todas as grandes inovações tecnológicas, apresenta uma dualidade inerente aos benefícios e riscos. Todos os avanços tecnológicos trouxeram consigo novos desafios e perigos que a humanidade precisaria aprender a gerenciar. Da mesma forma, o uso de modelos de IA, como o chatGPT, impõe seus próprios dilemas éticos, especialmente no que tange aos direitos fundamentais. A privacidade e a intimidade dos indivíduos, a imparcialidade no sistema de justiça, a integridade na educação, a veracidade na comunicação e a equidade no ambiente de trabalho são áreas que exigem regulação e urgente e supervisão constante. Assim, o desenvolvimento e a implementação dessas tecnologias devem ser conduzidos sob o olhar atento da sociedade e do Estado, garantindo que os avanços tecnológicos não comprometam valores humanos.

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

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

Investigar o uso prático de ferramentas de Machine Learning Automatizado (Auto-ML), analisar seus benefícios e limitações, e explorar a interação entre automação e o papel do desenvolvedor nos fluxos de trabalho de ML.	O estudo foi motivado pelo rápido crescimento do e uso crescente das ferramentas de Auto-ML, que promete simplificar o desenvolvimento do aprendizado e reduzir esforço manual. Contudo, há deficit na compreensão sobre como essas ferramentas são aplicadas na prática, até mesmo por falta de estudos apropriados, e como os usuários percebem o equilíbrio entre automação e intervenção humana.
Os autores realizaram um estudo qualitativo com 16 usuários de Auto-ML em contextos reais, utilizando entrevistas semiestruturadas. As entrevistas foram conduzidas de 2019 a 2020, com aproximadamente uma hora de duração, cada. Elas foram gravadas, transcritas e codificadas indutivamente para extrair temas comuns. A análise subsequente seguiu um método de codificação indutiva para identificar padrões e temas recorrentes.	As ferramentas de Auto-ML aceleram o desenvolvimento e a prototipagem, mas carecem de customização e transparência. Além disso, exigem supervisão humana significativa, especialmente para préprocessamento e ajuste de hiperparâmetros. Também falta suporte abrangente para todo o ciclo de vida de ML, necessitando de ajustes manuais para melhorar a qualidade. Por fim, há desafios de confiança e interpretabilidade, o que leva muitos a adotar uma abordagem híbrida, combinando automação com ajustes manuais.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

Informações:

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

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

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

B – RESOLUÇÃO

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

1 -

```
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

 Importações e Carga dos Dados

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

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

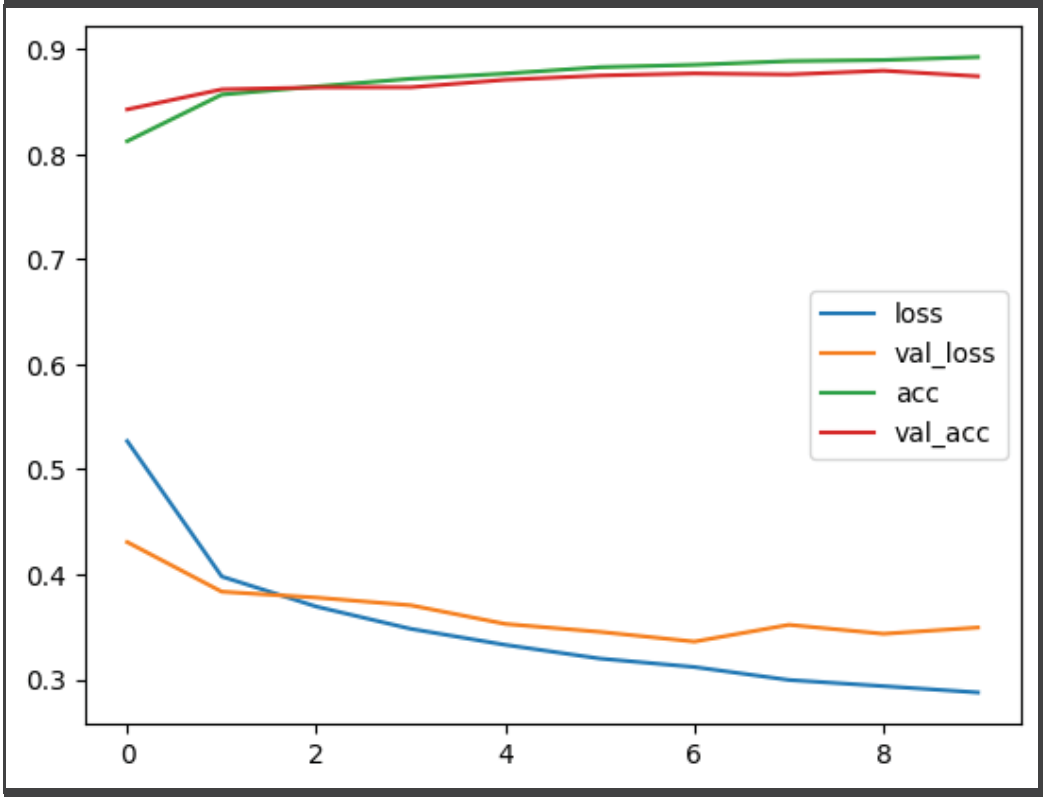
```
#(X_TRAIN, Y_TRAIN), (X_TEST, Y_TEST) = TF.KERAS.DATASETS.MNIST.LOAD_DATA()

PRINT("X_TRAIN.SHAPE: ", X_TRAIN.SHAPE)
PRINT("Y_TRAIN.SHAPE: ", Y_TRAIN.SHAPE)
PRINT("X_TEST.SHAPE: ", Y_TEST.SHAPE)
PRINT("Y_TEST.SHAPE: ", Y_TEST.SHAPE)



DOWNLOADING DATA FROM HTTPS://STORAGE.GOOGLEAPIS.COM/TENSORFLOW/TF-KERAS-DATASETS/TRAIN-LABELS-IDX1-UBYTE.GZ
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0S 0US/STEP
DOWNLOADING DATA FROM HTTPS://STORAGE.GOOGLEAPIS.COM/TENSORFLOW/TF-KERAS-DATASETS/TRAIN-IMAGES-IDX3-UBYTE.GZ
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 0S 0US/STEP
DOWNLOADING DATA FROM HTTPS://STORAGE.GOOGLEAPIS.COM/TENSORFLOW/TF-KERAS-DATASETS/T10K-LABELS-IDX1-UBYTE.GZ
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0S 1US/STEP
DOWNLOADING DATA FROM HTTPS://STORAGE.GOOGLEAPIS.COM/TENSORFLOW/TF-KERAS-DATASETS/T10K-IMAGES-IDX3-UBYTE.GZ
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 0S 0US/STEP
X_TRAIN.SHAPE:  (60000, 28, 28)
Y_TRAIN.SHAPE:  (60000,)
X_TEST.SHAPE:  (10000,)
Y_TEST.SHAPE:  (10000,)

 PRÉ-PROCESSAMENTO

X_TRAIN, X_TEST = X_TRAIN/255.0, X_TEST/255.0

I = TF.KERAS.LAYERS.INPUT(SHAPE=(28, 28))
X = TF.KERAS.LAYERS.FLATTEN()(I)
X = TF.KERAS.LAYERS.DENSE(128, ACTIVATION="RELU")(X)
X = TF.KERAS.LAYERS.DROPOUT(0.2)(X)
X = TF.KERAS.LAYERS.DENSE(10, ACTIVATION="SOFTMAX")(X)
MODEL = TF.KERAS.MODELS.MODEL(I, X)

COMPILAÇÃO E TREINAMENTO DO MODELO

MODEL.COMPILE(OPTIMIZER='ADAM',
              LOSS='SPARSE_CATEGORICAL_CROSSENTROPY',
              METRICS=['ACCURACY'])

R = MODEL.FIT(X_TRAIN,
              Y_TRAIN,
              VALIDATION_DATA=(X_TEST, Y_TEST),
              EPOCHS=10)

EPOCH 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 8S 4MS/STEP - ACCURACY: 0.7645 - LOSS: 0.6641 - VAL_ACCURACY: 0.8427 - VAL_LOSS:
0.4307
EPOCH 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9S 5MS/STEP - ACCURACY: 0.8550 - LOSS: 0.4028 - VAL_ACCURACY: 0.8618 - VAL_LOSS:
0.3834
EPOCH 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9S 4MS/STEP - ACCURACY: 0.8622 - LOSS: 0.3708 - VAL_ACCURACY: 0.8637 - VAL_LOSS:
0.3778
EPOCH 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9S 4MS/STEP - ACCURACY: 0.8701 - LOSS: 0.3514 - VAL_ACCURACY: 0.8638 - VAL_LOSS:
0.3705
```

```
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.8770 - loss: 0.3340 - val_accuracy: 0.8709 - val_loss:
0.3527
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.8851 - loss: 0.3170 - val_accuracy: 0.8750 - val_loss:
0.3452
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.8887 - loss: 0.3064 - val_accuracy: 0.8770 - val_loss:
0.3359
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 5ms/step - accuracy: 0.8881 - loss: 0.2978 - val_accuracy: 0.8760 - val_loss:
0.3519
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 5ms/step - accuracy: 0.8895 - loss: 0.2976 - val_accuracy: 0.8796 - val_loss:
0.3433
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7s 4ms/step - accuracy: 0.8938 - loss: 0.2796 - val_accuracy: 0.8743 - val_loss:
0.3493
```

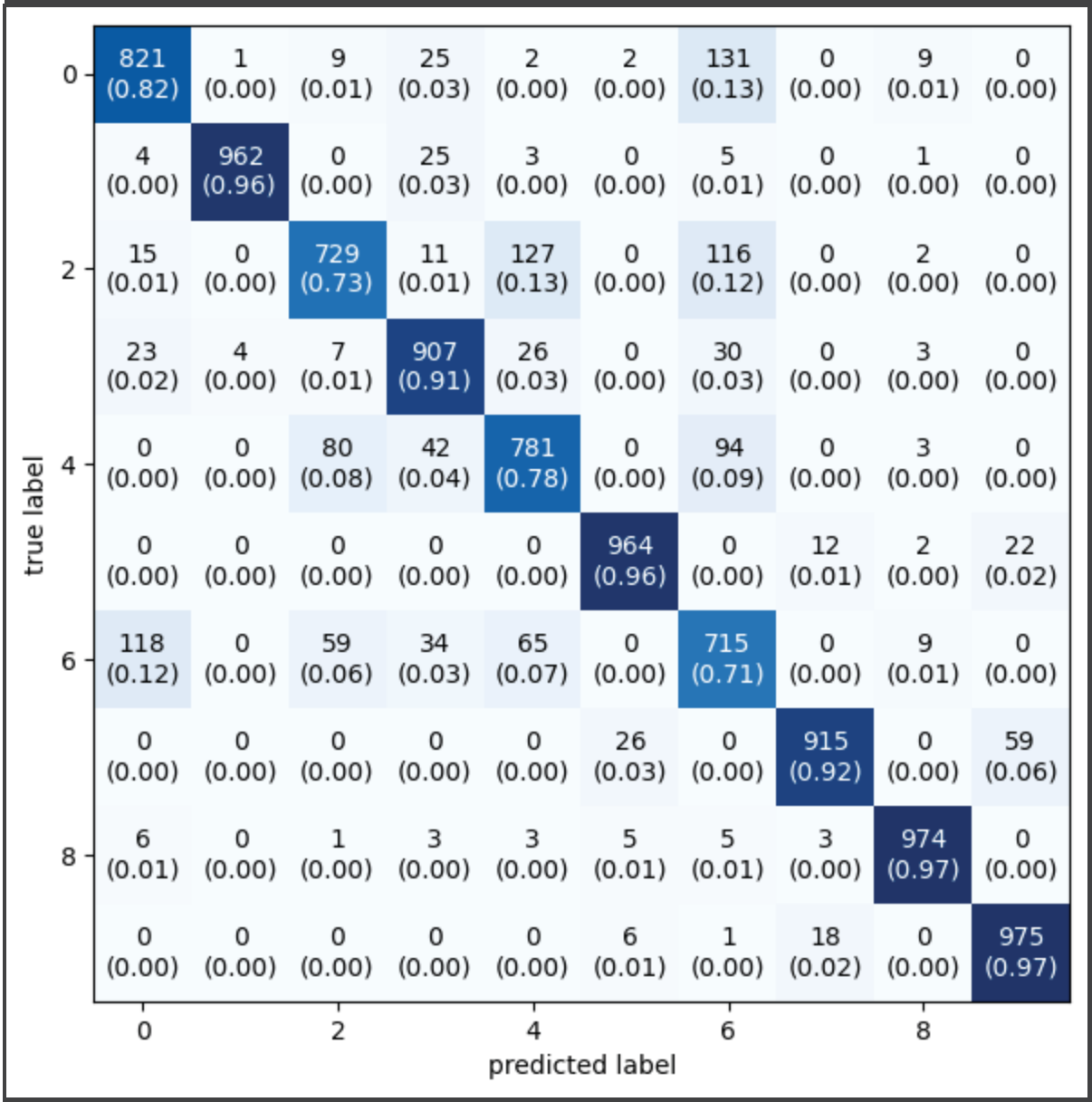
Avaliação do Modelo

```
# Plotar a função de perda
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
# Plotar a acurácia
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
# Avaliar o modelo com a base de teste
print( model.evaluate(x_test, y_test) )
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8765 - loss: 0.3424
[0.34930503368377686, 0.8743000030517578]
```

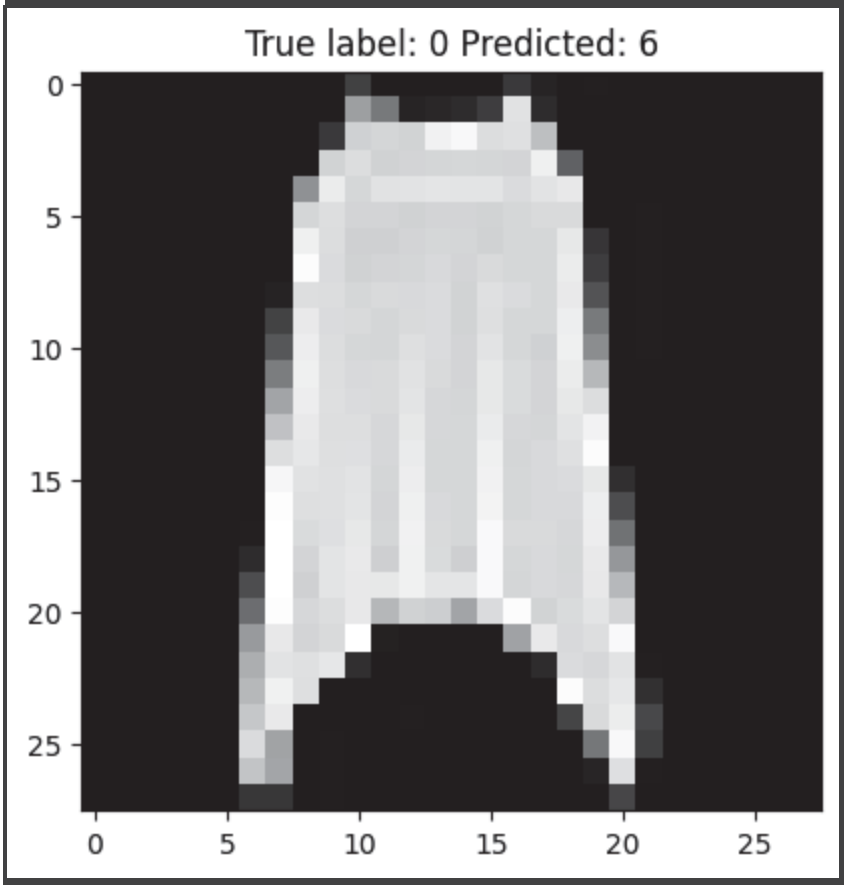


```
Efetuar Predições

# Efetuar predições
y_pred = model.predict(X_test).argmax(axis=1)
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
 show_normed=True)

313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step

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




```
Mostrar algumas classificações erradas
```

```
import numpy as np # Importing the NumPy library and assigning it the alias 'np'

# mostrar algumas classificações 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]))

Text(0.5, 1.0, 'True label: 0 Predicted: 6')
```



2-

```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.preprocessing import StandardScaler
from tensorflow.python.keras import backend
from sklearn.model_selection import train_test_split
```

```
FROM SKLEARN.METRICS IMPORT R2_SCORE, MEAN_SQUARED_ERROR
FROM MATH IMPORT SQRT
IMPORT SEABORN AS SNS
%MATPLOTLIB INLINE
TF.__VERSION__
```

```
2.17.1
```

```
CARGA DA BASE
```

```
URL = "HTTPS://ARCHIVE.ICS.UCI.EDU/ML/MACHINE-LEARNING-DATABASES/WINE-QUALITY/WINEQUALITY-RED.CSV"
DATA = PD.READ_CSV(URL, DELIMITER=';')

DATA.COLUMNS = [
'ACIDEZ_FIXA', # FIXED ACIDITY
'ACIDEZ_VOLATIL', # VOLATILE ACIDITY
'ACIDO_CITRICO', # CITRIC ACID
'ACUCAR_RESIDUAL', # RESIDUAL SUGAR
'CLORETOS', # CHLORIDES
'DIOXIDO_DE_ENXOFRE_LIVRE', # FREE SULFUR DIOXIDE
'DIOXIDO_DE_ENXOFRE_TOTAL', # TOTAL SULFUR DIOXIDE
'DENSIDADE', # DENSITY
'PH', # PH
'SULFATOS', # SULPHATES
'ALCOOL', # ALCOHOL
'SCORE_QUALIDADE_VINHO' # QUALITY
]

#X = DATA[:,0:11].ASTYPE(FLOAT)
#Y = DATA[:,3].ASTYPE(FLOAT)

X = DATA[['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
]]

Y = DATA[['SCORE_QUALIDADE_VINHO', # QUALITY
          ]]
```

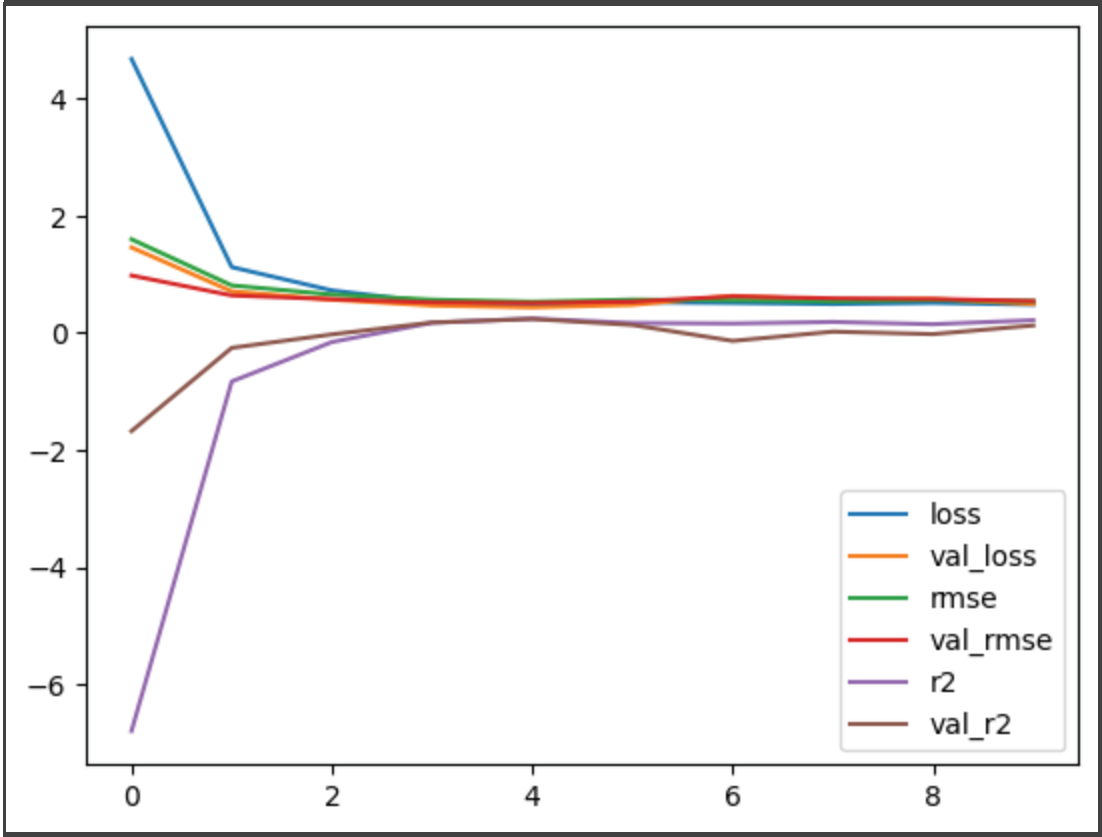
```
SEPARAR A BASE EM TREINO E TESTE
```

```
FROM SKLEARN.MODEL_SELECTION IMPORT TRAIN_TEST_SPLIT

X_TRAIN, X_TEST, Y_TRAIN, Y_TEST = TRAIN_TEST_SPLIT(X, Y, TEST_SIZE=0.25, RANDOM_STATE=0)
```

```
Normalização

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)



Criação do Modelo
i = tf.keras.layers.Input(shape=(11,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)
model = tf.keras.models.Model(i, x)



Adição de métricas R2 e RMSE

import tensorflow.keras.backend as backend

# Define métricas personalizadas
def rmse(y_true, y_pred):
    # Certifique-se de que os tipos são compatíveis
    y_true = backend.cast(y_true, dtype='float32')
    return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))

def r2(y_true, y_pred):
    y_true = backend.cast(y_true, dtype='float32')
    mean_y_true = backend.mean(y_true)
    ss_tot = backend.sum(backend.square(y_true - mean_y_true))
    ss_res = backend.sum(backend.square(y_true - y_pred))
    return 1 - ss_res / (ss_tot + backend.epsilon())



Compilação e Treino do Modelo
# Compilar o modelo
optimizer = tf.keras.optimizers.Adam(learning_rate=0.05)
model.compile(optimizer=optimizer,
                loss="mse",
                metrics=[rmse, r2])



EarlyStopping:

early_stopping = EarlyStopping(
    monitor='val_loss',             # Monitorar a métrica de validação
    patience=5,                       # Parar após 5 épocas sem melhora
    restore_best_weights=True      # Restaurar os melhores pesos
)
```

```
# Converter os dados de treino e teste para float32
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
y_train = y_train.astype('float32')
y_test = y_test.astype('float32')

# Treinar o modelo
r = model.fit(X_train,
                y_train,
                validation_data=(X_test, y_test),
                epochs=10,
                batch_size=32,
                callbacks=[early_stopping],  # Callback para interromper treino
                verbose=1)



Epoch 1/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - loss: 9.3568 - r2: -14.1953 - rmse: 2.3677 - val_loss: 1.4581 -
val_r2: -1.6769 - val_rmse: 0.9787
Epoch 2/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 1.2925 - r2: -1.1508 - rmse: 0.8878 - val_loss: 0.7080 - val_r2:
-0.2567 - val_rmse: 0.6384
Epoch 3/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.6887 - r2: -0.0790 - rmse: 0.6303 - val_loss: 0.5657 - val_r2:
-0.0266 - val_rmse: 0.5738
Epoch 4/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.5347 - r2: 0.1987 - rmse: 0.5771 - val_loss: 0.4657 - val_r2:
0.1765 - val_rmse: 0.5182
Epoch 5/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - loss: 0.4343 - r2: 0.2521 - rmse: 0.5178 - val_loss: 0.4349 - val_r2:
0.2371 - val_rmse: 0.5080
Epoch 6/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - loss: 0.4715 - r2: 0.2425 - rmse: 0.5363 - val_loss: 0.4771 - val_r2:
0.1342 - val_rmse: 0.5322
Epoch 7/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step - loss: 0.4584 - r2: 0.1971 - rmse: 0.5278 - val_loss: 0.6314 - val_r2:
-0.1377 - val_rmse: 0.6229
Epoch 8/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - loss: 0.5624 - r2: 0.0298 - rmse: 0.5900 - val_loss: 0.5596 - val_r2:
0.0204 - val_rmse: 0.5890
Epoch 9/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4439 - r2: 0.2444 - rmse: 0.5286 - val_loss: 0.5764 - val_r2:
-0.0220 - val_rmse: 0.5778
Epoch 10/10
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 0.4981 - r2: 0.1499 - rmse: 0.5518 - val_loss: 0.4933 - val_r2:
0.1258 - val_rmse: 0.5429



Avaliação do Modelo



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

```
<matplotlib.legend.Legend at 0x7c6f846b5210>
```




```
Predições

y_pred = model.predict(X_test).flatten()
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)


13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
mse =  0.43487635
rmse =  0.6594515543605908
r2 =  0.28848034143447876
```

3 -

```
1. Importações

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

2. Obtenção dos dados

!pip install gdown
import gdown

file_id = "1ZoHnICPb-95txfvdXW38_NQf4_a5hFmr"
download_url = f"https://drive.google.com/uc?id={file_id}"
output_file = "arquivo.csv"
gdown.download(download_url, output_file, quiet=False)
df = pd.read_csv(output_file, on_bad_lines='skip')
print(df.head())
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.10/dist-packages (5.2.0)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (from gdown) (4.12.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from gdown) (3.16.1)
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.10/dist-packages (from gdown) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from gdown) (4.66.6)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4->gdown)
(2.6)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from
requests[socks]->gdown) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from
requests[socks]->gdown) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from
requests[socks]->gdown) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from
requests[socks]->gdown) (2024.8.30)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.10/dist-packages (from
requests[socks]->gdown) (1.7.1)
Downloading...
From: https://drive.google.com/uc?id=1ZoHnICPb-95txfvdXW38_NQf4_a5hFmr
To: /content/arquivo.csv
100%|██████████| 12.8M/12.8M [00:00<00:00, 188MB/s]
```

	ISBN	Titulo	\
0	2005018	Clara Callan	
1	60973129	Decision in Normandy	
2	374157065	Flu: The Story of the Great Influenza Pandemic...	
3	393045218	The Mummies of Urumchi	
4	399135782	The Kitchen God's Wife	

	Autor	Ano	Editora	ID_usuario	Notas;;;;;;
0	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0;;;;;;
1	Carlo D'Este	1991	HarperPerennial	276726	2;;;;;;
2	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6;;;;;;
3	E. J. W. Barber	1999	W. W. Norton & Company	276729	1;;;;;;

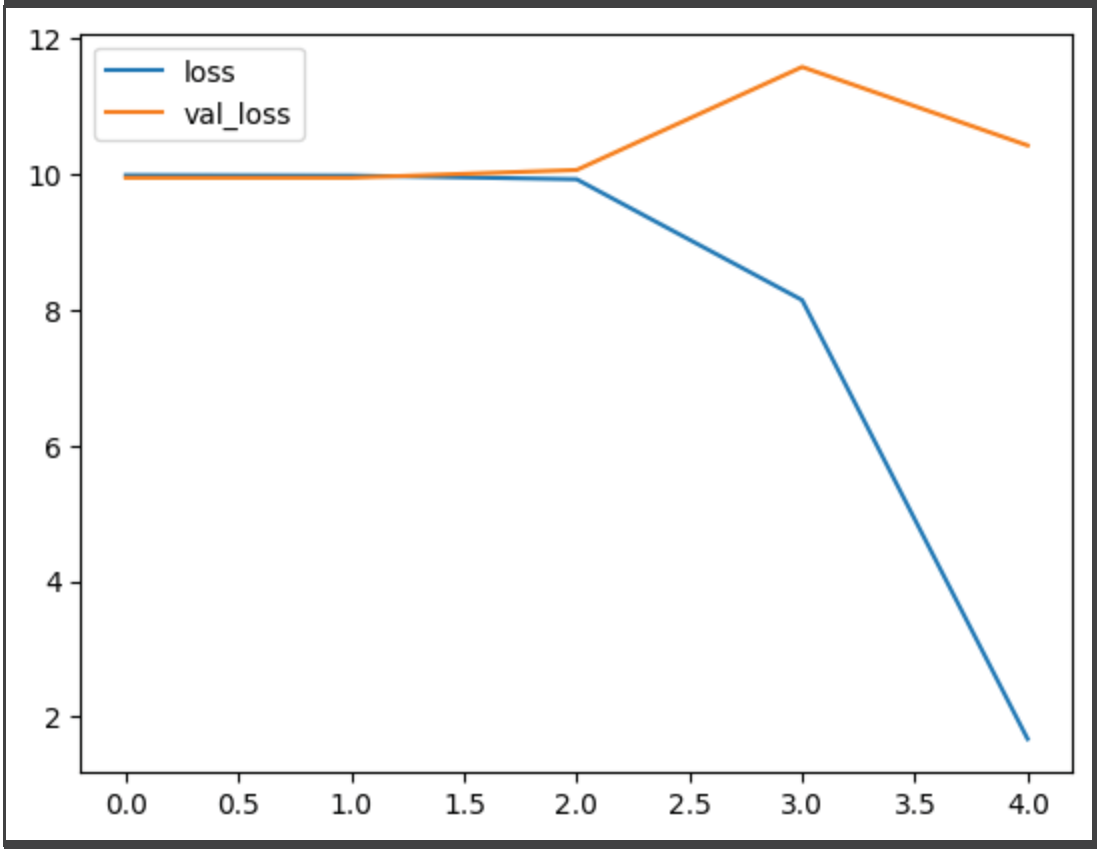
```
4                Amy Tan  1991         Putnam Pub Group      276729      9;;;;;;


df.describe()
ISBN    Titulo   Autor    Ano      Editora  ID_usuario       Notas;;;;;;
count    128868 108796 108803 108775 108762 108752 108704
unique   128867 97177  46743  157    8609   10865  65
top      486404242      Little Women      Agatha Christie   2002   Pocket   11676  8;;;;;;
freq     2      15     358    7750   2214   11306  10004

# Limpar os nomes das colunas
df.rename(columns=lambda x: x.strip(';'), inplace=True)

# Remover linhas com valores NaN
df.dropna(inplace=True)

# Limpar a coluna "Notas" e convertê-la para numérico
df['Notas'] = df['Notas'].str.split(';').str[0]  # Mantém apenas o valor antes do primeiro ponto e vírgula
df['Notas'] = pd.to_numeric(df['Notas'], errors='coerce')  # Converte para numérico, transformando valores
inválidos em NaN

# Remover linhas com NaN gerados após a conversão
df.dropna(subset=['Notas'], inplace=True)
```

```
3. Conversão de ID_usuario e ISBN para categoria

# Embeddings, devem ser categóricos
df.ID_usuario = pd.Categorical(df.ID_usuario)
df['new_ID_usuario'] = df.ID_usuario.cat.codes
df.ISBN = pd.Categorical(df.ISBN)
df['new_ISBN'] = df.ISBN.cat.codes

df.head()

ISBN    Titulo   Autor    Ano      Editora  ID_usuario       Notas    new_ID_usuario    new_ISBN
0       2005018Clara Callan      Richard Bruce Wright      2001    HarperFlamingo Canada      276725 0.0
        6628    22734
1       60973129       Decision in Normandy       Carlo D'Este     1991    HarperPerennial    276726 2.0
        6629    72277
2       374157065      Flu: The Story of the Great Influenza Pandemic...    Gina Bari Kolata 1999    Farrar
Straus Giroux     276727 6.0     6630    40145
3       393045218      The Mummies of Urumchi     E. J. W. Barber 1999    W. W. Norton & Company    276729
        1.0     6631    45697
4       399135782      The Kitchen God's Wife     Amy Tan 1991    Putnam Pub Group 276729 9.0     6631
        47822


3. Dimensões

#N = len(set(df.ID_usuario))
```

```
#M = len(set(df.ISBN))
# dimensão do embedding (tentar outros)
#K = 10



# Get N and M AFTER converting to categorical codes to ensure correct range
N = df['new_ID_usuario'].max() + 1  # Maximum user ID + 1
M = df['new_ISBN'].max() + 1         # Maximum ISBN + 1
# dimensão do embedding (tentar outros)
K = 10



4. Criar o modelo


# usuário
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)
# saída : num_samples, K
# livro
m = Input(shape=(1,))
m_emb = Embedding(M, K)(m)
m_emb = Flatten()(m_emb)
# saída : num_samples, 1, K
# saída : num_samples, K
x = Concatenate()([u_emb, m_emb])
x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)
model = Model(inputs=[u, m], outputs=x)



5. Compilar o modelo


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



6. Separação dos dados e pré-processamento


# Garantir que a coluna "Notas" seja numérica
df['Notas'] = pd.to_numeric(df['Notas'], errors='coerce')

ID_usuario, ISBN, Notas = shuffle(df.ID_usuario, df.ISBN, df.Notas)
Ntrain = int(0.8 * len(Notas))

# separar os dados 80% x 20%
train_usuario = ID_usuario[:Ntrain]
train_livro = ISBN[:Ntrain]
train_Notas = Notas[:Ntrain]
test_usuario = ID_usuario[Ntrain:]
test_livro = ISBN[Ntrain:]
```

```
test_Notas = Notas[Ntrain:]

# centralizar as notas
avg_Notas = train_Notas.mean()
train_Notas = train_Notas - avg_Notas
test_Notas = test_Notas - avg_Notas



# Replace original ID_usuario and ISBN with new categorical codes
train_usuario = df.loc[df.index[:Ntrain], 'new_ID_usuario'].values
train_livro = df.loc[df.index[:Ntrain], 'new_ISBN'].values
test_usuario = df.loc[df.index[Ntrain:], 'new_ID_usuario'].values
test_livro = df.loc[df.index[Ntrain:], 'new_ISBN'].values



7. Treino do modelo

# Verificar e tratar valores inválidos (como NaN)
def preprocess_data(data):
    #Converter o array numpy para pandas.Series temporariamente
    data = pd.Series(data)
    # Substituir valores não numéricos por 0
    data = pd.to_numeric(data, errors='coerce').fillna(0)
    # Converter de volta para numpy array como int32
    return np.array(data, dtype='int32')

# Aplicar a função de pré-processamento
train_usuario = preprocess_data(train_usuario)
train_livro = preprocess_data(train_livro)
train_Notas = np.array(train_Notas, dtype='float32')

test_usuario = preprocess_data(test_usuario)
test_livro = preprocess_data(test_livro)
test_Notas = np.array(test_Notas, dtype='float32')



epochs = 25
r = model.fit(
x=[train_usuario, train_livro],
y=train_Notas,
epochs=epochs,
batch_size=1024,
verbose=2, # não imprime o progresso
validation_data=([test_usuario, test_livro], test_Notas)
)



model.save('/content/model.keras')
model.save('/content/model.h5')



8. Plotar a função de perda
```

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




```
9. Recomendações para usuário

df_clean = df.dropna()

# Lista os IDs válidos
print("ID_usuario e new_ID_usuario sem dados NaN:")
print(df_clean[['ID_usuario', 'new_ID_usuario']])
```

```
ID_usuario e new_ID_usuario sem dados NaN:
        ID_usuario  new_ID_usuario
0           276725            6628
1           276726            6629
2           276727            6630
3           276729            6631
4           276729            6631
...            ...             ...
128863       29888            8262
128864       29888            8262
128865       29888            8262
128866       29888            8262
128867       29888            8262
```

```
[108649 rows x 2 columns]



# Recomendações para o usuário ------
# Gerar o array com o usuário único
# repete a quantidade de filmes
input_usuario = np.repeat(a=6628, repeats=M)
livro = np.array(list(set(df.new_ISBN)))
preds = model.predict( [input_usuario, livro] )
# descentraliza as predições
rat = preds.flatten() + avg_Notas
# índice da maior nota
idx = np.argmax(rat)
#print("Recomendação: Livro - ", livro[idx], " / ", rat[idx] , "*")

isbn_recomendado = livro[idx]
titulo_recomendado = df.loc[df['new_ISBN'] == isbn_recomendado, 'Titulo'].values[0]
print(f"Recomendação: Livro - '{titulo_recomendado}' / ISBN - {isbn_recomendado} / Nota - {rat[idx]} *")
```

4 -

```
1. Importação das bibliotecas
import tensorflow as tf
import numpy as np

import matplotlib as mpl

import IPython.display as display
import PIL.Image

2. Importação da imagem

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

# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
  name = url.split('/')[-1]
  image_path = tf.keras.utils.get_file(name, origin=url)
  img = PIL.Image.open(image_path)
  if max_dim:
    img.thumbnail((max_dim, max_dim))
  return np.array(img)

# Normalização da imagem
def deprocess(img):
  img = 255*(img + 1.0)/2.0
  return tf.cast(img, tf.uint8)

# Mostra a imagem
```

```
def show(img):
  display.display(PIL.Image.fromarray(np.array(img)))


# Redução do tamanho da imagem para facilitar o trabalho da RNN
original_img = download(url, max_dim=500)
show(original_img)
display.display(display.HTML('Image cc-by: <a
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg">Von.grzanka</a>'))

Downloading data from https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-cat_on_snow.jpg
2125399/2125399 ━━━━━━━━━━━━━━━━━━━━ 4s 2us/step
```

```
Image cc-by: Von.grzanka

3. Preparar o modelo de extração de recursos

base_model = tf.keras.applications.InceptionV3(include_top=False, weights='imagenet')

Downloading data from
https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels
_notop.h5
87910968/87910968 ━━━━━━━━━━━━━━━━━━━━ 4s 0us/step

# 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)


4. Cálculo da perda (loss)
```

```
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)


5. Subida de gradiente (Gradient ascent)

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

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

      for n in tf.range(steps):
        with tf.GradientTape() as tape:
          # Gradientes relativos a img
          tape.watch(img)
          loss = calc_loss(img, self.model)

        # Calculo do gradiente da perda em relação aos pixels da imagem de entrada.
        gradients = tape.gradient(loss, img)

        # Normalizacao dos gradintes
        gradients /= tf.math.reduce_std(gradients) + 1e-8

        # Na subida gradiente, a "perda" é maximizada.
        # Você pode atualizar a imagem adicionando diretamente os gradientes (porque eles têm o mesmo formato!)
        img = img + gradients*step_size
        img = tf.clip_by_value(img, -1, 1)

      return loss, img


deepdream = DeepDream(dream_model)
```

```
6. Circuito principal (Main Loop)

def run_deep_dream_simple(img, steps=100, step_size=0.01):

  img = tf.keras.applications.inception_v3.preprocess_input(img)
  img = tf.convert_to_tensor(img)
  step_size = tf.convert_to_tensor(step_size)
  steps_remaining = steps
  step = 0
  while steps_remaining:
    if steps_remaining>100:
      run_steps = tf.constant(100)
    else:
      run_steps = tf.constant(steps_remaining)
    steps_remaining -= run_steps
    step += run_steps

    loss, img = deepdream(img, run_steps, tf.constant(step_size))

    display.clear_output(wait=True)
    show(deprocess(img))
    print ("Step {}, loss {}".format(step, loss))


  result = deprocess(img)
  display.clear_output(wait=True)
  show(result)

  return result


dream_img = run_deep_dream_simple(img=original_img,
                                  steps=100, step_size=0.01)
```

```
7. Levando o modelo até um oitava

import time
start = time.time()

OCTAVE_SCALE = 1.30

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[:-1]
float_base_shape = tf.cast(base_shape, tf.float32)

for n in range(-2, 3):
  new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

  img = tf.image.resize(img, new_shape).numpy()

  img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```

```
249.2511751651764
```

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

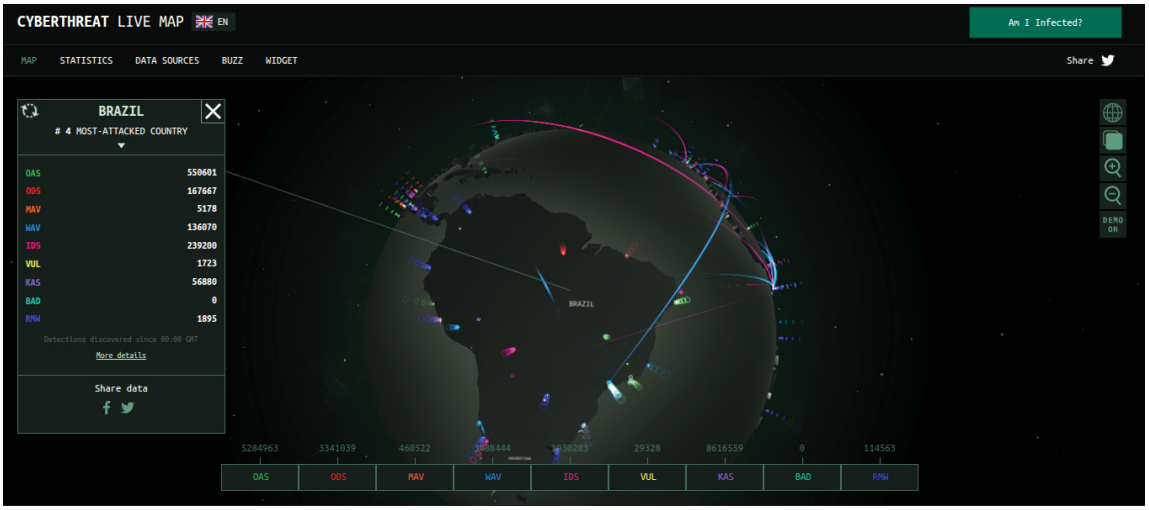
- O **conjunto de dados brutos** (**ou uma visualização de dados** que você acredite que possa ser **melhorada**);

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

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

B – RESOLUÇÃO

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



IAA007 - Visualização de Dados e Storytelling

28.02.2025

—

Dados Brutos

O conjunto de dados escolhido é disponibilizado pelo CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil) e pode ser acessado no link: https://stats.cert.br/incidentes/

Os dados representam a evolução de incidentes de segurança da informação reportados no Brasil ao longo do tempo, com informações mensais e anuais. O objetivo é entender como esse tipo de ataque cibernético tem se comportado no país.

Ano	Mês	Total	DoS	DoS (%)	Fraude	Fraude (%)	Invasão	Invasão (%)	Scan	Scan (%)	Web	Web (%)	Outros	Outros (%)
2016	jan	49.761	942	1,89	11.820	23,75	272	0,55	32959	66,23	2.266	4,55	1.502	3,02
2016	fev	52.843	137	0,26	8.761	16,58	95	0,18	40.911	77,42	1.422	2,69	1.517	2,87
2016	mar	88.934	12.851	14,45	10.541	11,85	106	0,12	58.962	66,3	4.526	5,09	1.948	2,19
2016	abr	43.161	6.147	14,24	11.660	27,02	126	0,29	22.531	52,2	1.637	3,79	1.060	2,46
2016	mai	46.017	338	0,73	14.431	31,36	125	0,27	27.496	59,75	2.361	5,13	1.266	2,75
2016	jun	57.195	823	1,44	12.616	22,06	126	0,22	41.210	72,05	1.554	2,72	866	1,51
2016	jul	36.597	1.676	4,58	6.213	16,98	116	0,32	25.763	70,4	1.804	4,93	1.025	2,8
2016	ago	47.619	260	0,55	7.352	15,44	137	0,29	27.970	58,74	10.802	22,68	1.098	2,31
2016	set	43.874	9.171	20,9	6.067	13,83	145	0,33	21.596	49,22	6.027	13,74	868	1,98
2016	out	34.072	410	1,2	4.175	12,25	86	0,25	23.257	68,26	5.050	14,82	1.094	3,21
2016	nov	56.257	15.735	27,97	5.207	9,26	320	0,57	25.858	45,96	7.758	13,79	1.379	2,45
2016	dez	62.013	11.942	19,26	3.354	5,41	41	0,07	35.390	57,07	10.234	16,5	1.052	1,7
2016	Total	618.343	60.432	9,77	102.197	16,53	1.695	0,27	383.903	62,09	55.441	8,97	14.675	2,37
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2024	fev	33.893	1.156	3,41	1.600	4,72	150	0,44	30.498	89,98	392	1,16	97	0,29

Ano	Mês	Total	DoS	DoS (%)	Fraude	Fraude (%)	Invasão	Invasão (%)	Scan	Scan (%)	Web	Web (%)	Outros	Outros (%)
2024	mar	39.950	943	2,36	1.763	4,41	370	0,93	36.306	90,88	470	1,18	98	0,25
2024	abr	32.003	94	0,29	2.223	6,95	1.102	3,44	27.791	86,84	393	1,23	400	1,25
2024	mai	29.843	91	0,3	2.404	8,06	114	0,38	26.385	88,41	588	1,97	261	0,87
2024	jun	32.011	98	0,31	2.238	6,99	123	0,38	28.534	89,14	615	1,92	403	1,26
2024	jul	40.808	4.563	11,18	2.327	5,7	165	0,4	32.995	80,85	551	1,35	207	0,51
2024	ago	55.186	19.042	34,51	1.664	3,02	233	0,42	33.636	60,95	469	0,85	142	0,26
2024	set	51.970	13.999	26,94	1.954	3,76	188	0,36	35.277	67,88	409	0,79	143	0,28
2024	out	73.957	10.238	13,84	1.909	2,58	193	0,26	58.910	79,65	2.145	2,9	562	0,76
2024	nov	47.627	6.723	14,12	1.707	3,58	179	0,38	34.135	71,67	839	1,76	4.044	8,49
2024	dez	38.768	477	1,23	1.393	3,59	129	0,33	35.797	92,34	675	1,74	297	0,77
2024	Total	516.556	57.498	11,13	23.637	4,58	3.100	0,6	417.621	80,85	7.937	1,54	6.763	1,31
2025	jan	53.324	2.552	4,79	1.605	3,01	232	0,44	47.869	89,77	744	1,4	322	0,6
2025	Total	53.324	2.552	4,79	1.605	3,01	232	0,44	47.869	89,77	744	1,4	322	0,6

Contexto e Público-Alvo

Contexto: O Brasil tem enfrentado um aumento significativo nos ataques cibernéticos nos últimos anos. Em 2024, o país registrou mais de 700 milhões de ataques, uma média de 1.379 por minuto, tornando-se o segundo país com mais ataques cibernéticos no mundo.

Esses ataques variam desde fraudes financeiras até invasões de sistemas corporativos, afetando tanto indivíduos quanto organizações.

Público-Alvo: Esta visualização é direcionada a profissionais de segurança da informação, gestores de TI, pesquisadores e formuladores de políticas públicas. O objetivo é fornecer insights claros

sobre as tendências de incidentes cibernéticos no Brasil, auxiliando na tomada de decisões estratégicas para mitigação de riscos.

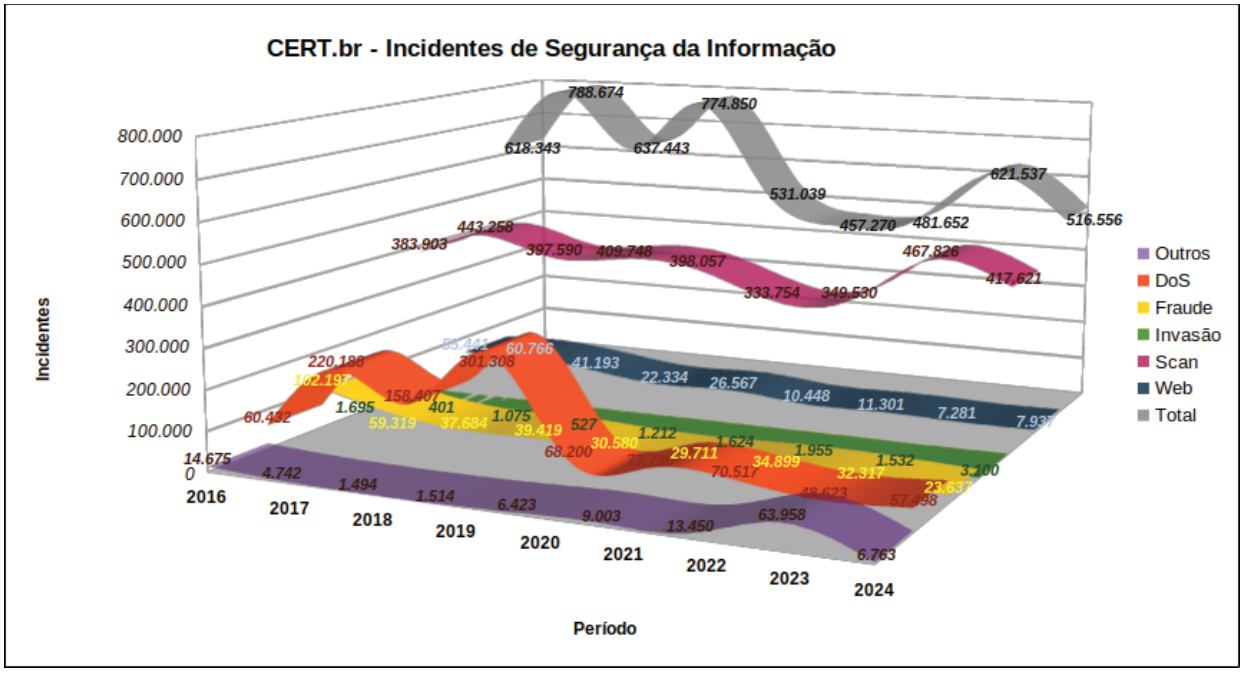
Visualização dos Dados

Gráfico de Linhas 3D

Mostra a tendência temporal do número total de incidentes notificados anualmente, permitindo identificar aumentos ou diminuições ao longo dos anos.

Ferramenta Utilizada

LibreOffice Calc - Componente de documento de planilha do LibreOffice. Essa ferramenta oferece flexibilidade e precisão na construção de visualizações personalizadas e de alta qualidade, além de ser gratuita.



Descrição da Narrativa/Storytelling

Introdução

"Nos últimos anos, os incidentes de segurança cibernética têm crescido sensivelmente no Brasil, afetando desde grandes corporações até usuários comuns. O CERT.br, responsável por monitorar e tratar esses incidentes, disponibiliza dados valiosos que nos ajudam a entender o cenário atual de ameaças."

Análise dos Dados:

Tendência Geral: Observa-se um aumento contínuo no número total de incidentes notificados ao CERT.br ao longo dos anos. Esse crescimento pode ser atribuído tanto ao aumento real de ataques quanto a uma maior conscientização e, consequentemente, maior número de notificações.

Categorias de Incidentes: A categoria Varreduras de rede (Scan) tem se destacado como a mais prevalente, seguida pela Negação de Serviço "DoS", indicando tentativas constantes de comprometer a disponibilidade e a segurança dos sistemas. “Fraudes” também apresentam números significativos, refletindo a sofisticação crescente de ataques de phishing e malware.

Conclusão e Recomendações:

Para Empresas: Investir em soluções de segurança, como firewalls, sistemas de detecção e prevenção de intrusões, além de programas contínuos de treinamento e conscientização para funcionários.

Para Usuários Finais: Adotar práticas seguras, como o uso de senhas fortes, autenticação de dois fatores e cautela ao clicar em links ou baixar anexos de fontes desconhecidas.

Para Governos e Reguladores: Desenvolver políticas públicas que incentivem a notificação de incidentes, promovam a educação em cibersegurança e estabeleçam padrões mínimos de segurança para organizações.

Glossário

Descrição das categorias de incidentes reportados ao CERT.br

DoS (DoS -- Denial of Service): notificações de ataques de negação de serviço, onde o atacante utiliza um computador ou um conjunto de computadores para tirar de operação um serviço, dispositivo ou rede.

Fraude: engloba as notificações de tentativas de fraude, ou seja, de incidentes em que ocorre uma tentativa de obter vantagem, que pode ou não ser financeira. O uso da palavra fraude é feito segundo Houaiss, que a define como "qualquer ato ardiloso, enganoso, de má-fé, com intuito de lesar ou ludibriar outrem, ou de não cumprir determinado dever; logro". Esta categoria, por sua vez, é dividida nas seguintes sub-categorias:

Phishing: notificações de casos de páginas falsas, tanto com intuito de obter vantagem financeira direta (envolvendo bancos, cartões de crédito, meios de pagamento e sites de comércio eletrônico), quanto aquelas em geral envolvendo serviços de webmail, acessos remotos corporativos, credenciais de serviços de nuvem, entre outros.

Malware: notificações sobre códigos maliciosos utilizados para furtar informações e credenciais.

Invasão: um ataque bem sucedido que resulte no acesso não autorizado a um computador ou rede.

Outros: notificações de incidentes que não se enquadram nas demais categorias.

Scan: engloba além de notificações de varreduras em redes de computadores (scans), notificações envolvendo força bruta de senhas, tentativas mal sucedidas de explorar vulnerabilidades e outros ataques sem sucesso contra serviços de rede disponibilizados publicamente na Internet.

Web: um caso particular de ataque visando especificamente o comprometimento de servidores web ou desfigurações de páginas na Internet.

Obs.: Vale lembrar que não se deve confundir scan com scam. Scams (com "m") são quaisquer esquemas para enganar um usuário, geralmente, com finalidade de obter vantagens financeiras. Ataques deste tipo são enquadrados na categoria fraude.

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

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

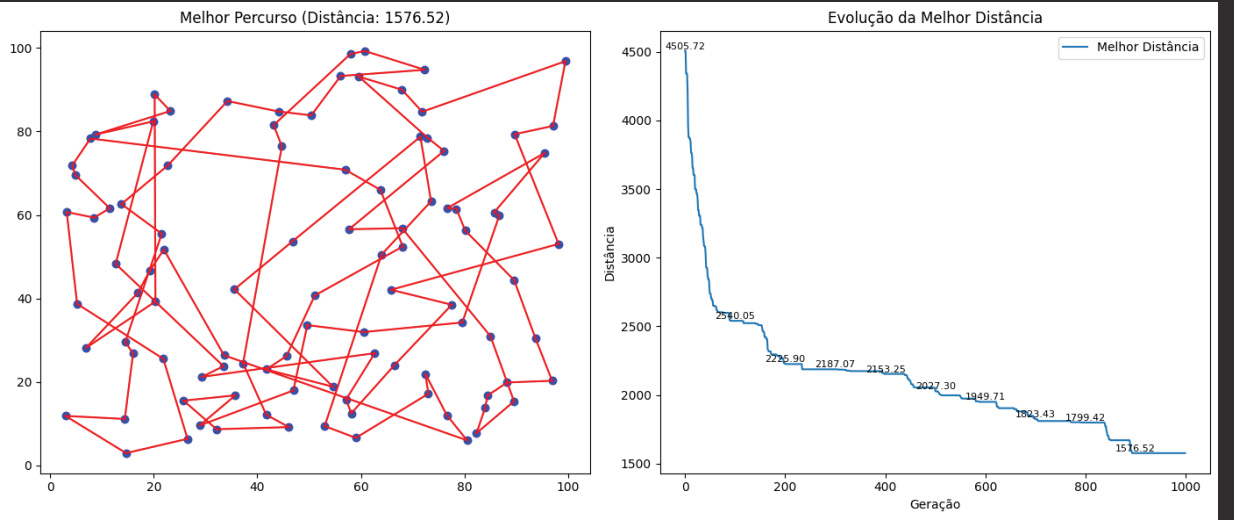
1 –

```
import numpy as np
import matplotlib.pyplot as plt
import random

# Parâmetros do Algoritmo Genético
NUM_CIDADES = 100
POP_SIZE = 100
NUM_GERACOES = 1000
TAXA_MUTACAO = 0.01
TAXA_CRUZAMENTO = 0.90
ELITISMO = True

# Gerar coordenadas aleatórias para as cidades
cidades = np.random.rand(NUM_CIDADES, 2) * 100

# Função para calcular a distância total de um percurso
def calcular_distancia(percurso):
    distancia = 0
    for i in range(NUM_CIDADES):
        cidade_atual = percurso[i]
        proxima_cidade = percurso[(i + 1) % NUM_CIDADES]
        distancia += np.linalg.norm(cidades[cidade_atual] - cidades[proxima_cidade])
    return distancia

# Função para inicializar a população
def inicializar_populacao():
    populacao = []
```

```
    for _ in range(POP_SIZE):
        percurso = list(range(NUM_CIDADES))
        random.shuffle(percurso)
        populacao.append(percurso)
    return populacao

# Função de seleção por torneio
def selecao_torneio(populacao, fitness, k=3):
    selecionados = []
    for _ in range(POP_SIZE):
        torneio = random.sample(list(zip(populacao, fitness)), k)
        vencedor = min(torneio, key=lambda x: x[1])[0]
        selecionados.append(vencedor)
    return selecionados

# Operador de cruzamento (OX - Order Crossover)
def cruzamento_ox(pai1, pai2):
    filho = [-1] * NUM_CIDADES
    inicio, fim = sorted(random.sample(range(NUM_CIDADES), 2))
    filho[inicio:fim] = pai1[inicio:fim]
    for i in range(NUM_CIDADES):
        if pai2[i] not in filho:
            for j in range(NUM_CIDADES):
                if filho[j] == -1:
                    filho[j] = pai2[i]
                    break
    return filho

# Operador de mutação (swap)
def mutacao(percurso):
    if random.random() < TAXA_MUTACAO:
        i, j = random.sample(range(NUM_CIDADES), 2)
        percurso[i], percurso[j] = percurso[j], percurso[i]
    return percurso

# Algoritmo Genético
def algoritmo_genetico():
    populacao = inicializar_populacao()
    melhor_percurso = None
    melhor_distancia = float('inf')
    historico = []

    for geracao in range(NUM_GERACOES):
        fitness = [calcular_distancia(percurso) for percurso in populacao]
        melhor_idx = np.argmin(fitness)
        if fitness[melhor_idx] < melhor_distancia:
            melhor_distancia = fitness[melhor_idx]
            melhor_percurso = populacao[melhor_idx]
        historico.append(melhor_distancia)

        nova_populacao = []
        if ELITISMO:
            nova_populacao.append(melhor_percurso)
```

```
        if geracao % 100 == 0:
            print(f'Geração {geracao}: Melhor distância = {melhor_distancia:.2f}')

        selecionados = selecao_torneio(populacao, fitness)
        while len(nova_populacao) < POP_SIZE:
            pai1, pai2 = random.sample(selecionados, 2)
            if random.random() < TAXA_CRUZAMENTO:
                filho = cruzamento_ox(pai1, pai2)
            else:
                filho = pai1 if random.random() < 0.5 else pai2
            filho = mutacao(filho)
            nova_populacao.append(filho)

        populacao = nova_populacao

    return melhor_percurso, melhor_distancia, historico

# Executar o algoritmo genético
melhor_percurso, melhor_distancia, historico = algoritmo_genetico()

# Plotar resultados
plt.figure(figsize=(14, 6))

# Plotar cidades e melhor percurso
plt.subplot(1, 2, 1)
plt.scatter(cidades[:, 0], cidades[:, 1], c='blue')
for i in range(NUM_CIDADES):
    cidade_atual = melhor_percurso[i]
    proxima_cidade = melhor_percurso[(i + 1) % NUM_CIDADES]
    plt.plot([cidades[cidade_atual][0], cidades[proxima_cidade][0]],
             [cidades[cidade_atual][1], cidades[proxima_cidade][1]], 'r-')
plt.title(f"Melhor Percurso (Distância: {melhor_distancia:.2f})")

# Plotar evolução da melhor distância com valores de 100 em 100 gerações
plt.subplot(1, 2, 2)
plt.plot(historico, label="Melhor Distância")
plt.title("Evolução da Melhor Distância")
plt.xlabel("Geração")
plt.ylabel("Distância")

# Adicionar valores na curva de 100 em 100 gerações
for i in range(0, NUM_GERACOES, 100):  # Corrigido para range(0, NUM_GERACOES, 100)
    plt.text(i, historico[i], f"{historico[i]:.2f}", fontsize=8, ha='center', va='bottom')

plt.legend()
plt.tight_layout()
plt.show()

Geração 0: Melhor distância = 4505.72
Geração 100: Melhor distância = 2540.05
Geração 200: Melhor distância = 2225.90
Geração 300: Melhor distância = 2187.07
Geração 400: Melhor distância = 2153.25
```

```
Geração 500: Melhor distância = 2027.30
Geração 600: Melhor distância = 1949.71
Geração 700: Melhor distância = 1823.43
Geração 800: Melhor distância = 1799.42
Geração 900: Melhor distância = 1576.52
```



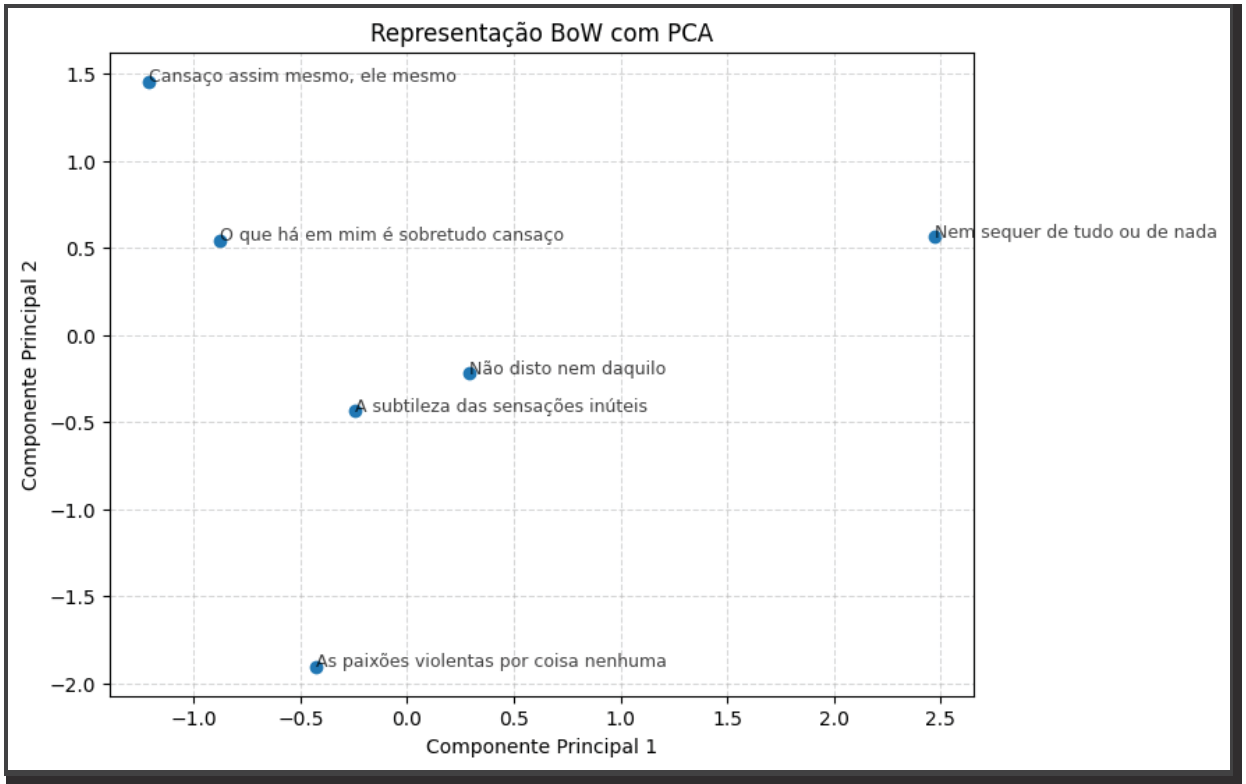
2 -

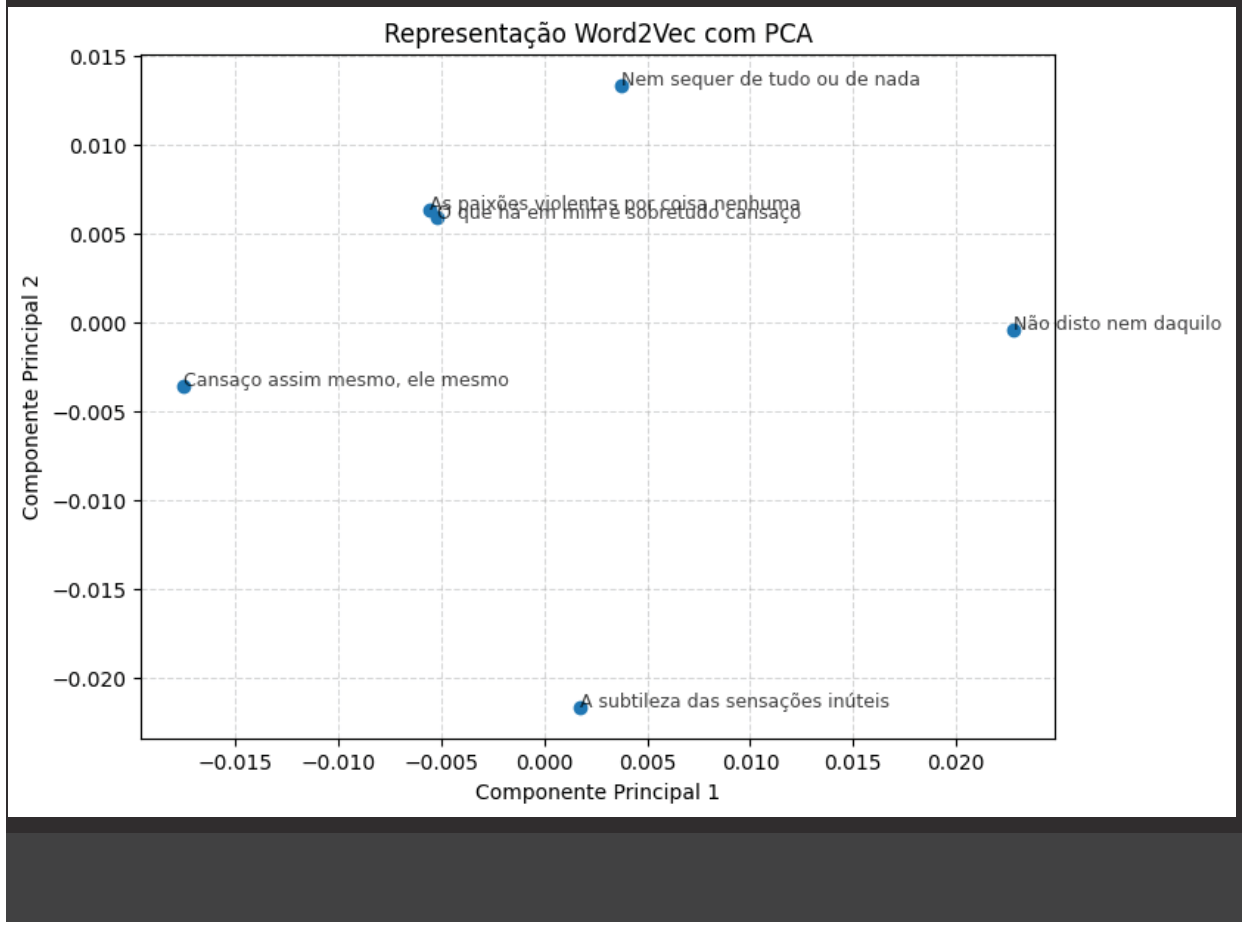
```python
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from gensim.models import Word2Vec
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Fragmentos de texto
textos = [
    "O que há em mim é sobretudo cansaço",
    "Não disto nem daquilo",
    "Nem sequer de tudo ou de nada",
    "Cansaço assim mesmo, ele mesmo",
    "A subtileza das sensações inúteis",
    "As paixões violentas por coisa nenhuma"
]

# 1. Representação Bag of Words (BoW)
vectorizer = CountVectorizer()
X_bow = vectorizer.fit_transform(textos).toarray()

print("Representação BoW:\n", X_bow)

# 2. Representação Word2Vec
# Tokenizando as sentenças
sentencas = [texto.split() for texto in textos]
```

```
# Treinando o modelo Word2Vec
model = Word2Vec(sentencas, vector_size=100, window=5, min_count=1, workers=4)

# Função para obter a representação vetorial de uma sentença
def get_sentence_vector(sentence, model):
    vectors = [model.wv[word] for word in sentence if word in model.wv]
    return np.mean(vectors, axis=0) if vectors else np.zeros(model.vector_size)

# Obtendo os vetores das sentenças
X_w2v = np.array([get_sentence_vector(sentence, model) for sentence in sentencas])

print("Representação Word2Vec:\n", X_w2v)

# 3. Redução de dimensionalidade com PCA
pca = PCA(n_components=2)

# Aplicando PCA para BoW
X_bow_pca = pca.fit_transform(X_bow)

# Aplicando PCA para Word2Vec
X_w2v_pca = pca.fit_transform(X_w2v)

# 4. Função para plotar os vetores
def plot_vectors(vectors, title, textos):
    plt.figure(figsize=(8, 6))
    plt.scatter(vectors[:, 0], vectors[:, 1])
    for i, txt in enumerate(textos):
        plt.annotate(txt, (vectors[i, 0], vectors[i, 1]), fontsize=9, alpha=0.75)
    plt.title(title)
    plt.xlabel('Componente Principal 1')
    plt.ylabel('Componente Principal 2')
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.show()

# Plotando os vetores BoW
plot_vectors(X_bow_pca, "Representação BoW com PCA", textos)

# Plotando os vetores Word2Vec
plot_vectors(X_w2v_pca, "Representação Word2Vec com PCA", textos)
```

```
Representação BoW:
 [[0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0]
 [0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 0]
 [0 1 1 0 0 0 0 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0]
 [1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1]]
Representação Word2Vec:
 [[ 1.42143574e-04  3.69573804e-03  1.76320667e-03 -8.44821916e-04
  -1.21004938e-04 -2.95322458e-03 -2.83163739e-04  3.03455465e-03
  -1.40727218e-03 -3.09175136e-03  9.42462473e-04 -2.27088504e-03
  -2.79429182e-03 -1.59696850e-03  1.37509813e-03  3.87226028e-04
```

```
  1.60709023e-04 -8.90918891e-04  1.04185520e-03 -2.87695858e-03
 -8.67416151e-04  1.62408466e-03  2.69007985e-03 -6.91289839e-04
 -1.37279963e-03  1.81623606e-03 -2.64866604e-03 -5.21331443e-04
  2.10835133e-03  9.58256191e-04 -1.24575943e-03 -2.84732413e-03
  3.13702226e-03  8.72914854e-04 -6.32880605e-04  2.65312614e-04
 -5.76417369e-04  1.56512228e-03 -1.28439360e-03 -5.50289231e-04
 -1.24532054e-03 -4.19446966e-04 -6.09606679e-04  4.00886266e-03
 -4.24873375e-04 -1.39611657e-03 -2.13272360e-04 -1.51247776e-03
 -2.36563943e-03  1.21569494e-04  9.96007948e-05 -1.33690133e-04
  1.64335666e-04 -1.13933720e-03 -3.63782863e-04 -3.46997258e-04
  1.20934099e-03  2.99073756e-04  3.21682706e-03  2.06495100e-03
 -4.14757943e-03 -2.51758518e-03  2.55290605e-03  2.31898529e-03
 -1.10723416e-03  1.53765432e-03  1.01628737e-03  1.61007326e-03
 -1.49088702e-03  7.29352876e-04 -1.57761516e-03  1.53842242e-03
  5.08851663e-04  1.72932958e-03  2.56270915e-03  1.94112351e-03
 -6.81154197e-05 -1.85502181e-03 -1.21593149e-03 -1.03957427e-03
 -1.27704372e-03  4.19378397e-04  3.01162619e-03  1.05924078e-03
  1.01795385e-03  2.38203723e-03  1.16205448e-03 -1.96027011e-03
  2.60584708e-03  2.18345248e-03  2.04465119e-03 -8.65034235e-06
  8.62921413e-04 -1.11192395e-03 -4.93961736e-04  2.42614362e-04
  2.85374932e-03  1.48197403e-03  1.99286640e-03  2.03519757e-03]
[ 1.91794126e-03 -9.23977408e-04  5.71497658e-05 -3.58609250e-04
 -1.59595953e-03  1.84657378e-03  2.78624427e-03  6.00594853e-04
 -5.90425543e-03 -2.17315578e-03 -1.10139826e-03 -1.05727313e-03
  1.44968810e-03 -4.67181322e-04 -8.64639878e-05 -3.20262229e-03
  …
  9.08854418e-05 -4.89573926e-03 -1.23332255e-04 -5.94064652e-04
  7.80099246e-04 -4.02577687e-04  4.35322436e-04  3.16394307e-03
  6.81607227e-04  1.01567083e-03  2.38119322e-03  6.78900338e-04
  8.09480262e-04  5.18390676e-03 -7.76373956e-04 -4.03795537e-04
  2.28370377e-03  2.88731139e-03  2.54691899e-04  4.85297758e-03
 -7.26605940e-04  2.76539737e-04  2.92785210e-03  1.43355725e-03
 -3.49269045e-04 -1.15332310e-03 -3.93623486e-06 -5.78932872e-04
 -2.49476492e-04  1.96747505e-03  1.66405182e-04  9.24309890e-04
  1.04372739e-03 -1.70573499e-03  1.43627450e-03 -3.66201042e-03
 -2.33643223e-03  6.67725923e-04  3.51808616e-03  1.21473837e-04
 -1.40112231e-03 -1.94242515e-03 -2.97558843e-03  1.50048023e-03
 -6.65812229e-04  4.18455078e-04 -2.39505293e-03 -1.24162342e-03
  4.40127542e-03  2.49166251e-03 -1.00843247e-03  3.28028761e-03
  1.59368070e-03 -4.32829838e-05  3.77423712e-03 -2.15464577e-04
 -5.77490835e-04 -2.31999831e-04 -1.28201023e-03 -1.84491405e-03]]
```


Representação BoW com PCA
Componente Principal 2
Componente Principal 1
Cansaço assim mesmo, ele mesmo
O que há em mim é sobretudo cansaço
Nem sequer de tudo ou de nada
Não disto nem daquilo
A subtileza das sensações inúteis
As paixões violentas por coisa nenhuma


Representação Word2Vec com PCA
Componente Principal 2
Componente Principal 1
Nem sequer de tudo ou de nada
As paixões violentas por coisa nenhuma
O que há em mim é sobretudo cansaço
Não disto nem daquilo
Cansaço assim mesmo, ele mesmo
A subtileza das sensações inúteis