UNIVERSIDADE PRESBITERIANA MACKENZIE

FELIPE NERES SILVA BEZERRA

GUILHERME DA SILVA BREVILATO

ISABELA KAZUE SINODUKA

PEDRO HENRIQUE BITTENCOURT DE FREITAS

São Paulo

2025

UNIVERSIDADE PRESBITERIANA MACKENZIE

FELIPE NERES SILVA BEZERRA

GUILHERME DA SILVA BREVILATO

ISABELA KAZUE SINODUKA

PEDRO HENRIQUE BITTENCOURT DE FREITAS

HAMBURGUERIA GALLIATE

OBJETIVO: ANALISE E PESQUISA DE PUBLICO ALVO PARA EXPANSÃO DE FRANQUIAS

Artigo apresentado à Universidade Presbiteriana Mackenzie como parte das exigências da matéria de "Projeto Aplicado II" do curso de "Tecnologia em Banco de Dados: Análise, Mineração e Engenharia de Dados" da turma 201825166.000.03ª do 2º semestre de 2025.

Docente: Prof. Anderson Adaime de Borba

São Paulo

2025

RESUMO

O setor de hamburguerias artesanais apresenta crescimento acelerado no Brasil, impulsionado pela popularização do delivery e pela busca por experiências gastronômicas diferenciadas. Nesse cenário competitivo, estratégias de marketing digital tornam-se fundamentais para otimizar investimentos em tráfego pago e orientar decisões estratégicas. Este trabalho tem como objetivo desenvolver um modelo preditivo capaz de antecipar a performance de criativos e postagens do instagram, utilizando imagens dos produtos e dados históricos de campanhas anteriores. Para isso, será aplicada uma abordagem de visão computacional baseada em redes neurais convolucionais (CNN), dada sua eficiência na identificação de padrões visuais relevantes. A proposta integra análise de imagens, dados temporais e metadados de marketing para prever métricas como CTR, taxa de conversão, CPA e ROAS, além de indicar o melhor timing para publicação e ranquear criativos conforme o potencial de retorno sobre o investimento, evitando assim testes por tentativa e erro (teste A/B). Espera-se, com isso, oferecer uma ferramenta inteligente que auxilie empresas do setor na tomada de decisões mais precisas e eficientes, contribuindo para maior competitividade e expansão sustentável.

Palavras-chave: Marketing digital; Previsão de performance; Redes neurais convolucionais; Análise de dados; Redes sociais; Teste A/B.

Sumário

Índice de Ilustrações

Índice de Tabelas

1. INTRODUÇÃO

No Brasil, o mercado de hamburguerias artesanais vem crescendo de forma acelerada, impulsionado por consumidores cada vez mais exigentes e atentos às tendências do setor.
Entre os principais movimentos que se destacam estão a expansão do delivery, que exige eficiência e agilidade para garantir a satisfação do cliente; a inovação nos cardápios, com a inclusão de opções vegetarianas, veganas e plant-based; além da valorização da qualidade das carnes utilizadas na produção[1].

Apesar do potencial, esse segmento enfrenta desafios significativos, como o aumento da concorrência e a necessidade constante de diferenciação. Nesse contexto, estratégias de marketing digital desempenham um papel central para atrair clientes, fortalecer a marca e otimizar investimentos em mídia paga. No entanto, a definição de quais criativos apresentam maior probabilidade de gerar bons resultados ainda ocorre, em muitos casos, como testes A/B, o que pode resultar em desperdício de recursos e decisões pouco assertivas.

Com os avanços recentes em ciência de dados e inteligência artificial, especialmente no campo da visão computacional, existem novas possibilidades para enfrentar esse desafio. Entre as técnicas disponíveis, as redes neurais convolucionais (CNN)[2] se destacam por sua capacidade de extrair padrões visuais complexos de imagens, tornando-se particularmente adequadas para prever o desempenho de criativos em campanhas de marketing.

A partir disto, este projeto tem como objetivo desenvolver um modelo capaz de prever a performance de publicações pagas e orgânicas, a partir da análise de imagens de produtos e do histórico de campanhas anteriores. A proposta busca oferecer uma ferramenta que permita identificar, de forma antecipada, quais criativos apresentam maior potencial de engajamento e conversão, possibilitando decisões mais precisas e estratégicas para empresas do setor.

1 CNN BRASIL, 2025.

2 Tensorflow, 2025.

2. DEFINIÇÃO DA ORGANIZAÇÃO: Hamburgueria Galliate

Figura 1: Logotipo da Empresa

A Hamburgueria Galliate[33] tem sua origem do Norte do Paraná.

Quando tudo parecia estável, a vida surpreendeu Thomas Louzano com uma notícia que mudaria tudo: sua esposa estava grávida — e de gêmeas. Mais do que emoção, o moamento despertou nele um senso profundo de responsabilidade. Era hora de transformar planos em ação, sonhos em realidade, e buscar uma nova fonte de sustento e legado para sua família.

Com espírito empreendedor e muita coragem, Thomas foi em busca de uma oportunidade real de investimento. Apaixonado por gastronomia e atento às tendências do mercado, decidiu apostar no que muitos consideram uma arte: o hambúrguer artesanal.

Sem sócios, sem equipe, apenas com vontade de vencer, iniciou a operação em Ribeirão do Pinhal (PR) — uma cidade pequena, mas cheia de potencial. Lá, montou a primeira versão da Hamburgueria Galliate, fazendo tudo sozinho: do preparo na chapa ao atendimento. Cada lanche carregava um pouco da sua história, cuidado e dedicação.

E o resultado? Sucesso imediato.

Os moradores aprovaram a proposta, os elogios começaram a circular, e o nome Galliate rapidamente ganhou força. A demanda cresceu e cruzou os limites da cidade.

Percebendo o interesse dos moradores da vizinha Santo Antônio da Platina, Thomas decidiu dar um passo ousado: abrir uma nova operação ali. Foi o início de uma nova fase. Com organização, padrão de qualidade e atendimento ágil, a Galliate se consolidou.

Hoje, a hamburgueria realiza mais de 2 mil pedidos por mês, com um faturamento médio de R$ 115 mil. Mas mais importante do que os números é a visão de futuro: Thomas entende que o próximo grande passo é conhecer profundamente seus clientes — entender seus gostos, hábitos, horários e preferências — para que possa oferecer experiências ainda mais personalizadas e expandir a Galliate para novas cidades.

O que começou com a chegada inesperada de duas filhas, se tornou a semente de um negócio sólido, com propósito e sabor. A Galliate não é apenas uma hamburgueria — é uma história de amor, coragem e visão empreendedora.

3 Para saber mais sobre a empresa e seus produtos, acesse: instagram.com/hamburgueriagalliate

3. OBJETIVO

O objetivo deste projeto é desenvolver um modelo capaz de prever a performance de publicações pagas em redes sociais, utilizando imagens dos produtos ofertados pelo cliente e dados históricos de campanhas anteriores. Com base nessas análises preditivas, será possível identificar antecipadamente quais criativos apresentam maior potencial de retorno sobre o investimento (ROAS), permitindo a otimização dos recursos investidos em tráfego pago e tornando as decisões mais precisas e eficientes[4].

O projeto integra análise de imagens, dados temporais e metadados de campanhas de marketing digital para oferecer um sistema inteligente de predição, capaz de:

- **Prever métricas de performance:** como CTR (taxa de clique), taxa de conversão, CPA (custo por aquisição) e ROAS (retorno sobre investimento) antes do lançamento dos anúncios.
- **Otimizar o timing de publicações:** identificando os melhores horários e dias da semana para cada criativo, aumentando as chances de engajamento e conversão.
- **Rankear criativos por performance esperada:** facilitando a priorização dos materiais que têm maior probabilidade de gerar resultados positivos para a marca.

3.1. Cenário de Aplicação

A equipe de marketing frequentemente recebe um portfólio extenso de materiais criativos, provenientes de fotógrafos e designers, em quantidade superior ao que será utilizado em campanhas de mídia paga. Como resultado do projeto, será entregue uma aplicação capaz de quantificar e estimar, de forma prévia e automatizada, a probabilidade de sucesso de cada imagem, garantindo que apenas os criativos com melhores estimativas de performance sejam selecionados para veiculação.

4 APVENDA, 2024.

4. DADOS

Os dados que serão utilizados foram fornecidos pelo cliente, onde encontram-se dados de post, interações e alcance nas redes, e também valores e custos investidos em cada segmento dentro deste negócio.

A analise será feita em etapas, extraindo dados numéricos das informações fornecidas, juntamente com métricas fornecidas pela plataforma de anúncios que servirão de variáveis resposta.

4.1. Publicações

A coleta será realizada com o objetivo de garantir que os dados sejam relevantes, atualizados e representativos da área de estudo, e serão obtidos como segue:

Nome	Tipo	Descrição
id	sting	Identificador da publicação
criativo	png	Imagem da publicação
datetime	datetime	Momento da publicação, contendo minuto, hora, dia, mês e ano.
week	int	Dia da semana (0=Domingo, 6=Sábado)
impressions	int	Número de vezes que o anúncio foi exibido
clicks	int	Número de cliques recebidos
ctr	float	Taxa de clique: `clicks / impressions`
conversions	int	Número de conversões associadas
conversion_rate	float	Taxa de conversão: `conversions / clicks`
cost	float	Custo total investido (R$ ou outra moeda)
cpc	float	Custo por Clique: `cost / clicks`
cpm	float	Custo por mil impressões: `cost / (impressions/1000)`
roas	float	Retorno sobre o investimento: `receita / cost`
engagement	int	Número total de interações (comentários, shares etc)

Tabela 1: Relatório de anúncios.

4.2. Tensores[5]

Para processamento matemático, as imagens serão convertidas em matrizes tridimensionais (tensores):

Dimensão	Ordem	Descrição
eixo_x	De acordo com a resolução da imagem	Identifica o pixel da imagem no eixo X
eixo_y	De acordo com a resolução da imagem	Identifica o pixel da imagem no eixo X
rgb	3	Identifica o valor do pixel em cada uma das três cortes da escala rgb (vermelho, verde e azul)

Tabela 2: Dimensões das imagens convertidas em Tensores.

A fim de padronizar as ordens das dimensões, as imagens terão suas resoluções reduzidas, a depender do desempenho do modelo quando de seu ajuste.

5 TENSORFLOW, 2024.

5. Análise Exploratória

5.1. Objetivo

O objetivo da Análise Exploratória é apresentar a análise exploratória dos anúncios pagos veiculados entre março e setembro de 2025, destacando os principais indicadores de desempenho e investimento.

A entrega visa fornecer uma visão clara do comportamento da base de dados, identificando padrões, variações e possíveis anomalias em métricas-chave como alcance, impressões, frequência, cliques, CTR, CPC, CPM e valor investido.

Escopo e Funcionalidades

A análise conta com imagens e metadados de campanhas de marketing digital para oferecer insights preditivos capazes de:

- Prever métricas-chave de performance
- Otimizar o timing das publicações
- Priorizar criativos por performance esperada
- Compreender o desempenho histórico das campanhas;
- Identificar anúncios de destaque e pontos de ineficiência;
- Subsidiar decisões estratégicas para otimização de criativos e alocação de recursos em tráfego pago;
- Fornecer insumos para o desenvolvimento do modelo preditivo de performance de anúncios.

5.2. Resumo Executivo

Essa análise tem como objetivo apresentar uma análise exploratória da performance dos anúncios veiculados, identificando padrões de eficiência, pontos de atenção e oportunidades de otimização.

Principais insights:

- CTR médio de aproximadamente 0,67%, com destaques de anúncios que atingiram taxas acima de 2,7%, mostrando alto engajamento.
- CPC médio em torno de R$14,73, porém com forte dispersão: alguns anúncios entregam cliques a menos de R$1, enquanto outros ultrapassam R$50 por clique.
- Investimento médio por anúncio em torno de R$192, mas com variação significativa (desde R$0,19 até campanhas acima de R$500).
- Anúncios de lançamento/inauguração e especiais de mês tiveram melhor performance, tanto em CTR quanto em eficiência de custos.
- Alguns anúncios apresentaram alto alcance e muitas impressões, mas baixo engajamento, indicando necessidade de revisão de criativo ou segmentação.

Para a análise exploratória do projeto, foram encontradas e utilizadas as seguintes métricas:

- **Alcance:** média de 6.705 pessoas, variando entre 27 e 14.000+.
- **Impressões:** em média 39 mil, mas com grande dispersão (até 150 mil).
- **Frequência**: em média 4,6 vezes por usuário.
- **Resultados:** média de 4.042, mas com alta variabilidade.
- **Investimento (BRL):** média de R$191, com alguns anúncios de baixo custo (<R$1).
- **CPM:** custo por mil impressões na faixa de R$5,32 em média.
- **Cliques no link:** média de 239 cliques, mas com forte assimetria (muitos anúncios com poucos cliques e alguns com picos altos).
- **CPC:** custo por clique varia muito — média de R$14,73, mas com casos abaixo de R$1 (eficientes) e acima de R$100 (ineficientes).
- **CTR:** taxa média de 0,67%, com destaque para anúncios com CTR > 1% (acima da média do mercado).

5.3. Visão Geral da Base

Os dados foram disponibilizados pelo cliente contendo a seguinte estrutura:

Estrutura do dataset:

- **Total de anúncios analisados:** 38
- **Período de veiculação:** datas variam entre março e setembro de 2025 (com alguns anúncios ainda ativos em “Contínuo”).
- **Número de colunas (variáveis):** 24
- **Identificação:** ID, nome da campanha, conjunto de anúncios, nome do anúncio.
- **Descritivos**: tipo de arquivo, status de veiculação, nível de veiculação.
- **Datas:** início, término, início/término dos relatórios.

- **Métricas de performance:** alcance, impressões, frequência, resultados, cliques, leads.
- **Indicadores financeiros:** valor investido, custo por resultado, CPM, CPC**.**

Estatísticas gerais das métricas principais

- **Alcance:**
  - Média ~ 6.705 pessoas
  - Variação de 27 a 14.000+ pessoas

- **Impressões:**
  - Média ~ 39 mil impressões
  - Máximo próximo de 150 mil

- **Frequência:**
  - Usuários expostos em média 4,6 vezes a um mesmo anúncio

- **Cliques no link:**
  - Média ~ 239 cliques
  - Anúncios de destaque superaram 1.300 cliques

- **CTR (taxa de cliques):**
  - Média ~ 0,67%
  - Variando entre 0,004% (baixa efetividade) até 3,1% (altamente atrativo)

- **CPC (custo por clique):**
  - Média ~ R$14,73
  - Casos eficientes abaixo de R$1 e ineficientes acima de R$100

- **CPM (custo por mil impressões):**
  - Média ~ R$5,32
  - Variação de R$1,87 a R$9,5

- **Investimento por anúncio:**
  - Média ~ R$192
  - Variação de R$0,19 até R$500+

5.4. Análise Detalhada

5.4.1. Distribuição e Tendências

- Alcance: a maior parte dos anúncios alcançou entre 3.000 e 10.000 pessoas, com poucos outliers (ex.: um anúncio com apenas 27 pessoas).
- Impressões: seguem padrão semelhante ao alcance, mas variam de forma mais acentuada, indo de 32 até quase 150 mil impressões.
- Frequência: usuários foram expostos em média 4,6 vezes por anúncio, sugerindo uma estratégia de repetição moderada.
- Insight: anúncios com impressões muito altas, mas CTR baixo, sugerem desperdício de verba em públicos pouco responsivos.

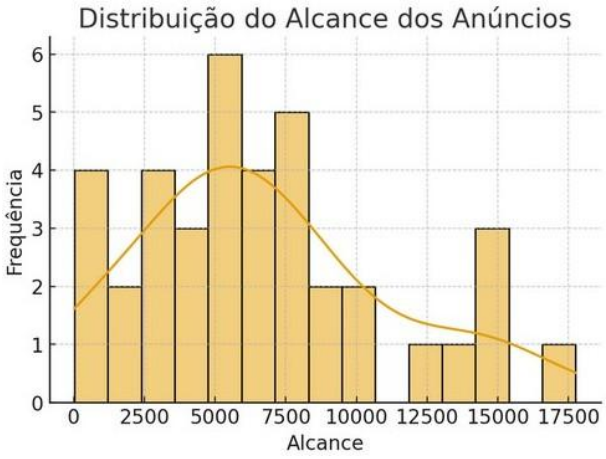

Figura 2: Grafico de Distribuição de Alcance

5.4.2. Relação Impressões × Cliques

O cruzamento entre impressões e cliques no link mostra:

- Correlação positiva esperada: mais impressões tendem a gerar mais cliques.
- Porém, alguns anúncios destoam:
    - Muitas impressões, mas pouquíssimos cliques (indicando criativos pouco atrativos).
    - Baixo volume de impressões, mas alta taxa de cliques (bom engajamento, mesmo com pouco alcance).

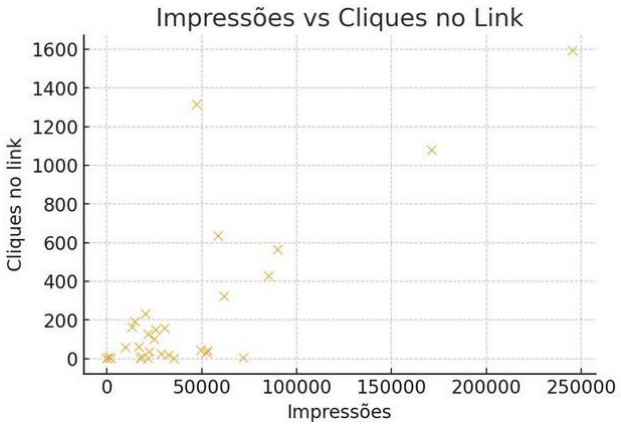

Figura 3: Impressões vs Cliques no Link

5.4.3. Ranking de Performance

Top 5 por CTR (mais atrativos):

- Anúncios de inauguração e promoções especiais destacaram-se com CTR acima de 2,7%.
- Exemplos: “AD01 - INAUGURAÇÃO” e “Especial do Mês - Agosto”.

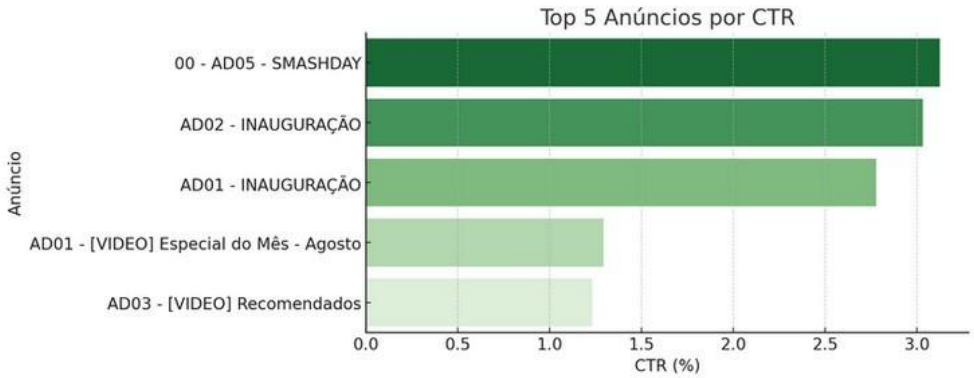

Figura 4: Gráfico Top 5 melhores CTR

Piores 5 por CTR (menos atrativos):

- CTR abaixo de 0,1%, alguns com milhares de impressões mas quase nenhum clique.
- Indicam problemas de segmentação ou baixa qualidade criativa.

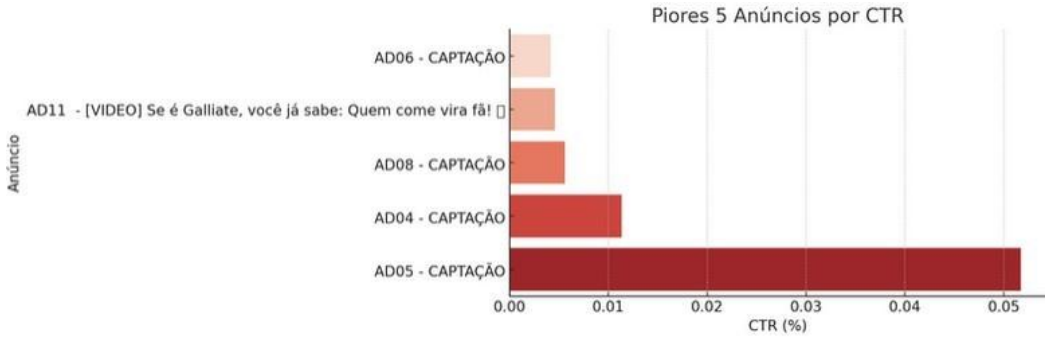

Figura 5: Top 5 piores CTR

Top 5 por CPC (mais eficientes):

- Custo por clique abaixo de R$1, extremamente rentáveis.
- Normalmente anúncios com criativos diretos e promoções.

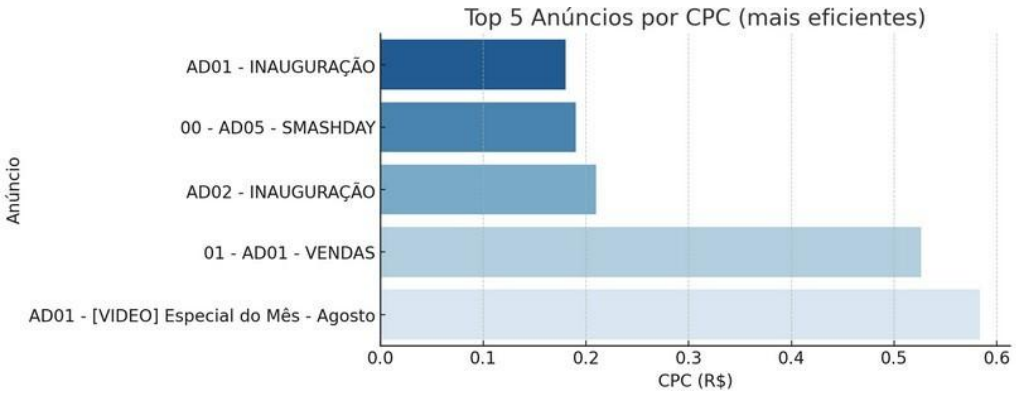

Figura 6: Top 5 melhores CPC

Piores 5 por CPC (menos eficientes):

- CPC acima de R$50, chegando a valores superiores a R$100.
- Totalmente inviáveis do ponto de vista financeiro.

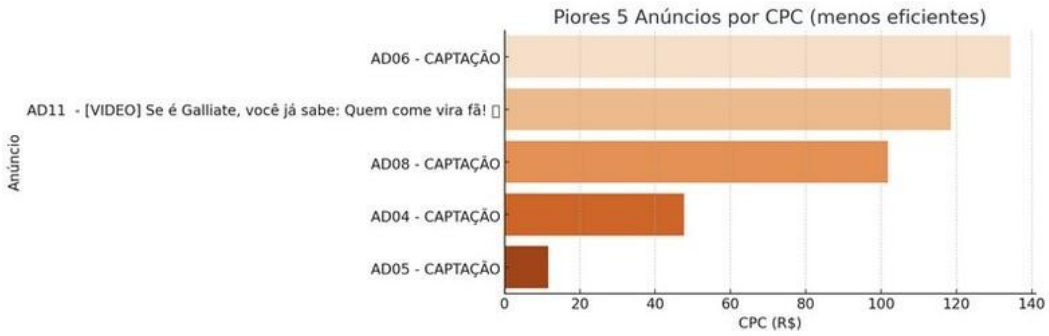

Figura 7: Top 5 piores CPC

Diagnóstico:

- Os anúncios mais eficientes combinam criativos atrativos (alta CTR) com baixo custo por clique (CPC).
- Os piores anúncios drenam orçamento, entregando muitas impressões sem conversão em cliques.

5.5. Insights e Diagnósticos

5.5.1. O que funcionou bem

- Anúncios de inauguração e campanhas sazonais (ex.: “Especial do Mês – Agosto”) apresentaram CTR acima de 2% e CPC baixo, mostrando que mensagens ligadas a novidade, promoção ou exclusividade são altamente atrativas.
- Formatos em vídeo tiveram desempenho competitivo, aparecendo entre os mais eficientes, sugerindo que o público responde melhor a conteúdos mais dinâmicos.
- Campanhas de menor alcance, mas com mensagem clara e direta, apresentaram engajamento proporcionalmente alto, indicando que segmentação mais precisa pode gerar melhor custo-benefício.

5.5.2. Pontos críticos

- CTR muito baixo (<0,1%) em diversos anúncios com milhares de impressões → evidencia baixa atratividade criativa ou segmentação inadequada.
- CPC extremamente elevado (>R$50) em alguns casos → desperdício de verba com cliques caríssimos.
- Anúncios com alto alcance, mas quase nenhum clique → indicam conteúdo pouco relevante para o público atingido.
- Alguns anúncios apresentam investimento alto (>R$500) mas resultados fracos → é preciso rever a estratégia de orçamento.

5.5.3. Relação entre métricas

- CTR vs CPC: anúncios com CTR alto tendem a ter CPC baixo, comprovando que engajamento reduz custo.
- CPM vs Alcance: quanto maior o alcance, menor o CPM médio, sugerindo que campanhas amplas são mais baratas por mil impressões, mas nem sempre eficazes em cliques.
- Cliques vs Investimento: nem sempre maior gasto gera mais cliques — a eficiência depende mais da qualidade do criativo e da segmentação.

5.6. Conclusão

A análise dos 38 anúncios avaliados evidencia que o desempenho das campanhas não está diretamente relacionado ao volume de investimento, mas sim à qualidade do criativo e à adequação da segmentação. Os anúncios mais eficientes foram aqueles que

souberam alinhar mensagem clara, apelo comercial direto e formato atrativo (especialmente vídeos), alcançando CTR elevado e CPC baixo.

Por outro lado, a presença de campanhas com alto alcance, mas engajamento mínimo, e de anúncios com CPC excessivamente alto, reforça a necessidade de um monitoramento contínuo e ajustes estratégicos.

Com base nos insights levantados, recomenda-se a redistribuição de verba para anúncios comprovadamente mais eficientes, a revisão criativa de campanhas de baixo engajamento e a implementação sistemática de testes A/B. Essa abordagem permitirá não apenas otimizar resultados imediatos, mas também construir um processo de aprendizado contínuo para campanhas futuras.

6. Referencial Teórico: Redes Neurais Convolucionais

6.1. Introdução

As redes neurais convolucionais (*Convolutional Neural Networks* – CNN) são um tipo especializado de redes neurais artificiais, projetadas para processar dados que possuem uma estrutura em grade, como imagens. Inspiradas no sistema visual humano, as CNNs aprendem automaticamente a identificar padrões e características relevantes nos dados, como bordas, texturas, formas e objetos, de forma hierárquica. Isso permite que sejam altamente eficazes em tarefas de reconhecimento de imagens, processamento de vídeo e, de forma mais ampla, em problemas de visão computacional.

As CNNs se destacam por sua capacidade de reduzir a necessidade de pré-processamentos complexos e extrair automaticamente atributos importantes, facilitando o desenvolvimento de sistemas precisos e robustos, para diversas aplicações, incluindo diagnósticos médicos, segurança e carros autônomos. Também podem ser empregadas para prever valores contínuos a partir de dados de entrada estruturados. Exemplos incluem a estimativa de ângulos, contagem de objetos, ou predição de métricas contínuas em imagens e sinais (caso em que se enquadra este projeto).

Todo algoritmo será desenvolvido em ambiente *Python*, versão 3.11, e o modelo implementado utilizará recursos da API *Keras*, inclusa na biblioteca *Tensorflow*, versão 2.20.0. Keras é uma API que fornece uma interface mais simples, modular e intuitiva para criar e treinar modelos de *deep learning* rapidamete. *Tensorflow*, é uma biblioteca criada pela Google para oferecer uma ampla gama de funcionalidades para computação numérica e aprendizado de máquina, incluindo operações de baixo nível, otimizações, suporte a GPU/TPU, etc.

6.2. Imagens como Tensores

Tensor é um termo generalista que abrange objetos de estudo da álgebra linear. Grosso modo, são conjuntos de qualquer dimensionalidade e tamanho. Um escalar seria um tensor de ordem 0, um vetor é um tensor de ordem 1, uma matriz de ordem 2, e assim sucessivamente.

Uma imagem pode ser entendida como uma matriz de pixels, onde cada pixel corresponde a uma coordenada para altura e largura da imagem. Já o pixel, por sua vez, pode de ser representado em número quando disposto numa escala, como RGB, onde sua cor é representada por três números,

6 Conteúdo baseado principalmente na documentação oficial do TensorFlow/Keras <https://www.tensorflow.org/guide/keras>, complementado por IBM Brasil <https://www.ibm.com/br-pt/think/topics/convolutional-neural-networks>, e GANESH (Towards Data Science, 2019) <https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37>.

numa escala que vai de 0 a 255, cada qual representando a intensidade de vermelho, verde e azul, respectivamente.

Se uma imagem multicolorida é composta de outras três imagens monocromáticas, então temos um tensor de ordem 3 (uma matriz cúbica), cujas dimensões são altura, largura e canal: HxWxC.

É uma boa prática na arquitetura de redes convolucionais a criação de bach, que seria a concatenação de tensores de ordem 3. Assim como uma matriz quadrada surge da concatenação de vetores e uma matriz cúbica como a concatenação de matrizes quadradas, aqui um bach é um tensor de ordem 4, que possibilita à rede neural calcular grupos de imagens paralelamente. O resultado tem as dimensões bach, altura, largura e canais; BxHxWxC.

Essa prática não altera de que forma os cálculos são processados ao longo das camadas da rede, mas cria um paralelismo (várias imagens são processadas de uma vez), ajuda a diminuir ruídos (se comparado com o uso de imagens individuais; *batch*=1) e é um dos procedimentos para evitar *overfitting*.

6.3. Preparação e Pré-processamento de Dados

Alguns requisitos devem ser atendidos para que os dados estejam preparados para alimentar a rede neural convolucional. Com efeito, pouco é feito fora da própria arquitetura do modelo: os dados devem ser separados entre três subgrupos: treino, validação e teste (a necessidade é explicada mais a diante); deve-se assegurar que todos os exemplos não estejam duplicados e estejam devidamente atrelados a seus rótulos, que, neste projeto, são as variáveis contínuas dependentes que se almeja prever, normalizados e sem valores ausentes.

Quanto aos passos inclusos nas funcionalidades da API que abrangem o pré-processamento incluem a formação de *batches*, o redimensionamento das imagens para tamanhos iguais e padronizados e a normalização dos valores da escala RGB.

Para tornar o modelo mais robusto, um procedimento recomendado é o *data augmentation*, que consiste em criar alterações nos atributos do objeto que, garantidamente, não alterariam seus rótulos. Por exemplo: na tentativa de identificar o rosto de uma pessoa em uma imagem, essa tarefa deve ser cumprida independentemente da nitidez, brilho, temperatura, posição e orientação da imagem. Além de tornar a base de dados mais plural, submete o modelo maiores generalizações. Entretanto, este não é um método conceitualmente aplicável neste projeto, já que os rótulos que se almeja prever são mais abstratos. Por exemplo: não sabemos se a alteração do brilho ou da orientação de uma imagem poderiam alterar o alcance ou cliques na rede social.

6.4. Arquitetura da CNN

A arquitetura típica, recomendada na documentação da Keras, é como segue:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
```

Figura 8: Exemplo extraído de <https://www.tensorflow.org/tutorials/images/cnn?hl=pt-br>.

O modelo é criado determinando o tipo – neste caso, será um modelo sequencial, em que cada *batch* é processada em cada camada uma após a outra. Em seguida, uma sequência de camadas intercaladas de convolução (Conv2D) e *pooling* (MaxPooling2D) são adicionadas ao modelo. Essa alternância permite que a rede aprenda representações progressivamente mais abstratas e complexas, iniciando por características locais e indo a decisões mais globais. A última camada é antecedida por um processo de achatamento (*Flatten*), que entrega vetores às camadas densas finais (Dense), que funcionam como redes neurais densas comuns. Os números são parâmetros de tamanho e quantidade de *kernels*, números de neurônios e, em algumas camadas, a função de ativação. Os detalhes desses parâmetros são explicados com mais detalhes a seguir.

6.5. Kernel e Stride

Muitos dos processos de um algoritmo convolucional se valem do conceito de *kernel*, que se trata de uma janela que faz uma varredura em um tensor e retorna um novo tensor, podendo ou não ter os mesmos tamanhos e dimensões. A ideia central é que um tensor de menor tamanho faça uma varredura sobre o tensor de interesse e gere um novo resultado, podendo esse processo ser feito várias vezes para produzir uma nova dimensão com valores distintos.

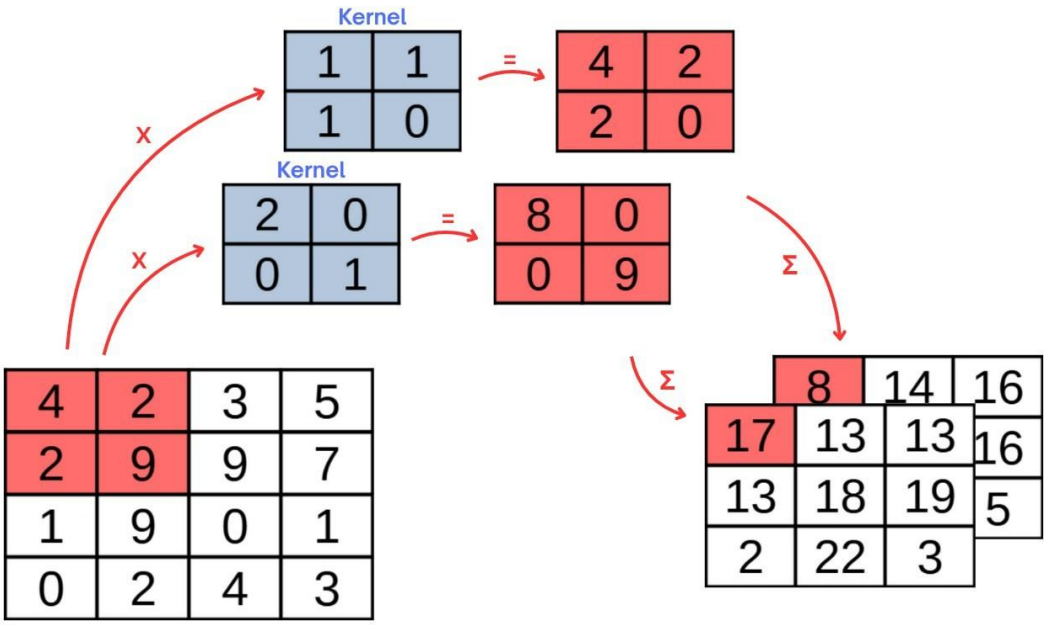

Figura 9: Exemplo aplicação de varredura por kernel. Elaboração própria.

No exemplo acima, um tensor 4x4 passa por dois 2x2, resultando em um tensor 3x3x2. Cada atuação dos kernels multiplica valores paralelamente e seu resultado é o somatório desses produtos (o que equivaleria a um novo par de valores para aquele conjunto de pixel). Então, o kernel dá um passo para o lado e executa o mesmo cálculo. O número de células de defasagem entre uma aplicação do kernel e a aplicação vizinha é chamado de *stride*.

6.6. Camada convolucional

A cada batch, esta camada faz uma varredura com um número K de kernels pré-definido. Normalmente, o kernel tem dimensão 3x3xC e resulta em um tensor H-2 x W-2 x K para cada imagem. O tensor de saída chamado de mapa de características (feature map), então é entregue para a camada seguinte.

Se uma nova camada convolucional for incluída logo depois da primeira, com um número L de kernels de tamanho 3x3, os kernels terão dimensão 3x3xK e resultarão em tensores H-4 x W-4 x L.

Em comparação comum neurônio em uma rede neural densa, os valores dentro do kernel são como coeficientes e, além deles, deve haver um viés que é somado ao seu resultado.

6.6.1. Função de ativação

A função de ativação é parte do processo de convolução. É aplicada ao resultado da soma ponderada e do viés, e gera uma transformação que insere não-linearidade ao processamento da rede neural, o que é necessário para que a rede represente e aproxime funções complexas, aumentando muito sua capacidade de aprendizagem e generalização.

A função de ativação mais amplamente utilizada é a *ReLU* (Unidade Linear Retificada), que pode ser definida como $f(x)=max(0, x)$. Ou seja, se o resultado da soma ponderada dos pixels e do viés for negativo, o que a camada entregará para o mapa de características resultante será zero.

6.6.2. Zero-Padding

Como visto acima, cada processo de convolução tem saída menor que a entrada, em largura e em altura. Uma forma de evitar essa perda de tamanho é adicionando uma borda (*padding*) de valores ao redor do tensor formada apenas de zeros, permitindo que a convolução abranja as extremidades sem perder detalhes nas bordas da imagem. Com um *kernel* 3x3xC e *stride*=1, uma camada unitária em torno de todo o tensor antes da convolução compensa a perda, mas, para *kernels* e *stride* diferentes, pode ser necessário bordas de 0 mais largas ou assimétricas.

6.7. Camada de *Pooling*

Entre as camadas de convolução, é acrescentada uma camada de agrupamento (*Pooling*), que tem como objetivo reduzir as dimensões espaciais dos mapas de características, destacando os valores mais relevantes ou representativos. Essa redução ajuda a diminuir a complexidade computacional e torna a rede mais resistente a pequenas variações e deslocamentos nas imagens.

O funcionamento também é baseado na varredura por kernel: são definidas altura e largura do kernel, que é aplicado em cada canal, preservando o número de canais mas reduzindo as outras dimensões. Em casos mais habituais, aplica-se kernel 2x2 e *stride*=2; isso faz com que, por exemplo, uma entrada de tamanho 180x180x32 resulte em um mapa de ativação de 90x90x32.

A camada de *Pooling* não executa os mesmos cálculos das camadas convolucionais, mas sim valores mais representativos, dependendo do método: o método *Max Pooling* extrai o maior valor

dentro de cada janela, enquanto o *Average Pooling* resulta na média dos valores que o *kernel* abrange.

6.8. *Flatten* e Camada Totalmente Conectada

Após uma série de convoluções intercaladas por camadas de *pooling*, o que se tem é uma quantidade maior de canais de menores altura e largura. A camada totalmente conectada interage com todos os valores resultantes de uma imagem como se fosse uma só entrada para uma rede neural densa convencional. Para isso, o tensor resultante até então passa por um processo de "achatamento" (*flatten*), que consiste em transformá-lo em vetores concatenados, cujos valores servem, cada um, como neurônio de entrada para a camada densa que segue. Assim, a Camada Totalmente Conectada ignora conceitos de altura, largura e canais.

Essa última etapa, a Camada Totalmente Conectada, é composta por neurônios organizados também em camadas. Cada neurônio de determinada camada é conectada a todas as saídas da camada anterior, que recebe esses valores ponderados cada um por um peso e somados junto ao *viés* do neurônio. Assim como nas camadas de convolução, há uma função de ativação que processa a saída do neurônio antes de entregá-la à próxima cada ou como resultado final.
Para a camada de neurônios de saída, uma função de ativação própria é aplicada para que o resultado esteja de acordo com os rótulos, para que esses sejam comparados. Como a aplicação de CNN é amplamente utilizada para classificação de imagens, as funções mais comuns são:

Sigmoide: $f(x)=\frac{1}{1+e^{-x}}$

cria uma curva que ajusta, de forma mais precisa, classificações binárias, fornecendo a chance percentual de determinado objeto de entrada ser positivo. É aplicado em camadas de saída com apenas um neurônio.

Softmax: $\sigma(z)_i=\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

$$\sigma(z_i)=\frac{e^{z_i}}{e^{z_1}+e^{z_2}+\ldots+e^{z_K}}$$

é uma normalização da função sigmoide para múltiplas classes. Cada neurônio de saída responde a uma das classes possíveis e a soma dos resultados nunca ultrapassa 100%.
Para as finalidades deste projeto, o resultado é linear, o que não exige qualquer ativação na camada final. Outra característica conveniente é a possibilidade de configurar nossa rede com múltiplas saídas (*multi-output neural network*): isso nos possibilita prever várias métricas com um mesmo treinamento, com cada variável sendo incumbência de um neurônio de saída da camada totalmente conectada.

6.9. *Backpropagation* e as Épocas

O *backpropagation*, ou retropropagação, é o processo pelo qual a rede, de fato, aprende. Todo processo até então pode ser entendido como um palpite da rede para tentar prever os rótulos dos objetos de entrada (que, na nossa aplicação, são as métricas na rede social). Após estabelecer aleatoriamente os pesos e vieses para os kernels e os neurônios na camada totalmente conectada, um resultado é obtido e comparado com o rótulo real esperado; muito provavelmente a rede não dará uma resposta correta em sua primeira tentativa com o primeiro *batch*, mas isso permite computar o erro (e, para o caso deste projeto, a intensidade desse erro).

Em redes convolucionais, assim como em redes densas, os gradientes do erro em relação a cada peso dos *kernels* são calculados, levando em conta a contribuição desse peso para o erro final.

Esses gradientes são então multiplicados pela taxa de aprendizagem para atualizar cada peso no sentido de minimizar o erro. O viés do *kernel* também é atualizado de forma similar. A diferença está que, enquanto num neurônio as entradas são valores individuais que multiplicam diretamente os pesos, nos *kernels* essa operação envolve convoluções e somas locais, e as entradas podem influenciar múltiplos pesos que são compartilhados ao longo da entrada, devido à natureza dos filtros deslizantes. Em suma, a atualização dos pesos é um processo reverso à convolução, aplicando a taxa de aprendizagem e o erro sobre os pesos dos *kernels*.

A cada *batch*, a rede realiza uma previsão e calcula o erro em relação ao valor esperado. Esse erro atualiza os pesos e vieses de toda rede a depender dos valores atuais e das entradas, o que modifica qual será a saída para o próximo *batch* calculado pela rede. Depois que todos os *batches* são processados pela rede, tem-se o que é chamado de Época. A rede percorre toda a base de dados um número de épocas pré-determinado na arquitetura do modelo.

6.10. Função de Perda e Ajuste de Hiperparâmetros

Uma prática altamente recomendada é separar a base de dados em três porções, para: treinamento (de 60% a 80%), validação (de 10% a 20%) e teste (de 10% a 20%),. Isso é necessário para identificar o quanto o modelo está aprendendo a lidar com o problema em mãos, mais do que aos dados disponíveis[7].

Todo processo descrito até então utiliza apenas a porção da base destinada para treino. Mas, a cada época, ocorre um cálculo geral de performance para que se possa entender o quanto o modelo está errando em relação aos valores esperados. Se esse valor de performance fosse baseado apenas nos dados de treino, não seria possível distinguir o quanto a rede aprendeu a lidar com o problema de forma geral e o quanto ele aprendeu a lidar unicamente com a base de dados. Seria o equivalente a aplicar uma prova cujas questões são as mesmas usadas durante o estudo: não saberíamos se o modelo aprendeu a resolver problemas ou apenas decorou as respostas. Para isso, entrega-se também um conjunto de dados que não é usado para treinar, mas apenas para saber se a rede não está sofrendo de *overfitting*: quando aquelas soluções que a rede desenvolveu são adequadas apenas a seus dados de treinamento, mas não para dados aos quais ela não teve acesso.

7 ENCORD, 2024.

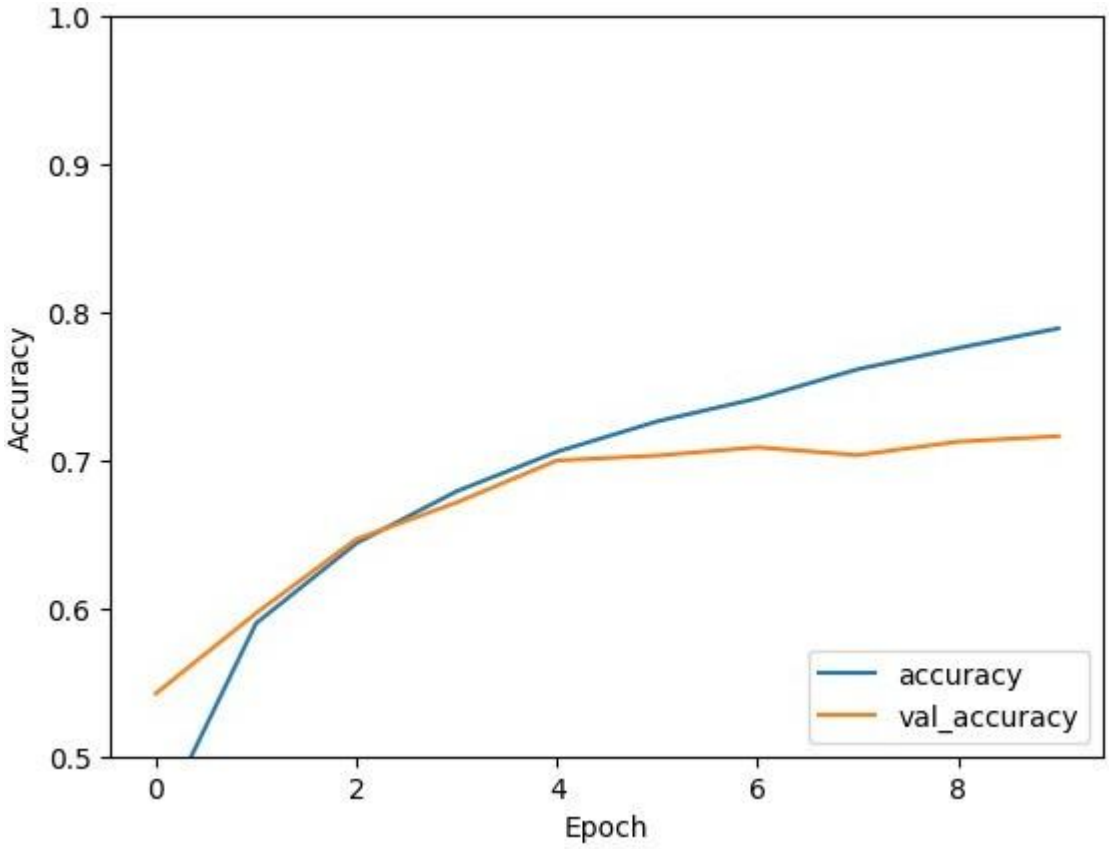

Figura 10: Exemplo extraído de <https://www.tensorflow.org/tutorials/images/cnn? hl=pt-br>. Aqui a linha azul demonstra a evolução da performance do modelo medida com a própria base de treino, enquanto a linha laranja mostra a avaliação na base de validação.

Uma métrica muito comum é a média dos quadrados dos erros (*Mean Squared Error* – MSE): para cada exemplo da validação, obtém-se uma previsão a partir dos parâmetros que a rede atingiu ao fim de uma época e compara-se com os valores reais (esperados) para aqueles exemplos – um princípio igual ao erro que está no início do *backpropagation*. Eleva-se esses erros ao quadrado, o que torna todos positivos, e calcula-se a média deles. Quanto maior o MSE para uma época, pior a performance da rede até então.

Outra medida similar é a média dos erros absolutos (*Mean Absolute Error* – MAE), que funciona de forma muito similar ao MSE, mas que utiliza valores absolutos em vez de quadrados para que erros negativos não anulem os positivos. Pode-se ainda obter a raiz quadrada do MSE (*Root Mean Squared Error* – RMSE), mais utilizado quando a grandeza original da entrada precisa ser comparada aos erros.

A métrica de performance escolhida para este projeto é o R^2, que é definido como:

$$\sum^{n}(y-\hat{y})^2$$

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

$$\sum_{i=1}(y_i - \bar{y})$$

e mede a proporção da variabilidade das respostas esperadas que o modelo consegue explicar, variando de 0 a 1, sendo que valores mais próximos de 1 indicam melhor ajuste.[8]

Ao longo de um processo de treinamento, é possível identificar o quanto a rede está sofrendo de *overfitting*: quando, ao longo das épocas, a performance para os dados de treino melhora significativamente, mas não observa-se o mesmo progresso nos dados de validação. A partir desse relatório, podem ser realizados experimentos de ajustes de hipermarâmetros: número e tamanho de *kernels*, *stride*, quantidade de camadas, taxa de aprendizagem.

Esse processo de refinamento de hiperparâmetros gera uma nova esfera de ajustes que também podem incorrer em *overfitting*: o modelo pode estar perfeitamente ajustado aos dados dentro do conjunto de validação, mas não aos dados fora dele. Para tal, uma terceira porção, os dados de teste, é separada para medir a performance final do modelo.

Para que o processo de treinamento, validação e teste do CNN seja eficaz, seria fundamental dispor de uma base de dados abundante e diversificada. Isso porque, quanto maior e mais representativa a base de dados, maior a capacidade do modelo em generalizar para novos dados, possibilitando validações e testes com amostras consistentes e resultados mais confiáveis. Se a base de dados é pequena, o risco de que o modelo identifique padrões recorrentes na amostra e que não são relevantes num contexto generalista aumenta consideravelmente. Segundo Golestaneh, Taheri e Lederer (2025), espera-se que o erro superior da predição (o valor máximo esperado para o erro que o modelo pode cometer ao fazer previsões) está diretamente ligado ao tamanho da amostra, numa proporção de $\epsilon \approx 1/\sqrt{n}$.

Como visto na sessão “Análise Exploratória da Amostra”, este projeto dispõe de apenas 38 exemplos nesta etapa do projeto e, conforme explicado em “Preparação e Pré-processamento de Dados” nesta sessão, o aumento artificial dessa quantidade não é conceitualmente aplicável. Reconhecemos que isso pode impactar na robustez das conclusões e no treinamento do modelo. Porém, a equipe está empenhada em buscar uma forma viável de ampliar a base ou contornar sua deficiência na próxima etapa do projeto, garantindo resultados mais representativos.

---

8 Carvalho Junior, A. Métricas de Desempenho em Modelos de IA (parte 2). Disponível em: <https://eailab.labmax.org/2025/05/13/metricas-de-desempenho-em-modelos-de-ia-parte-2/>. Acesso em: 2 out. 2025.

7. Aplicação do Modelo

7.1. Transfer Learning

Usamos uma rede neural convolucional (CNN) MobileNet pré-treinada no ImageNet como extratora de atributos visuais (include_top=False, pooling global). Sobre ela, acoplamos uma cabeça de regressão para estimar diretamente a métrica-alvo. O treino ocorreu em duas fases: primeiro com o backbone congelado (treina apenas a cabeça, mais estável com pouco dado); depois, um fine-tuning parcial descongelando camadas finais com taxa de aprendizado menor para ajustar a rede ao nosso domínio visual. Com este modelo a ideia é ajustar partes deste para o contexto do projeto.

Abaixo estão as etapas do fluxo seguido disponível no github:

- **Importações e configuração**
  - Importação de bibliotecas como TensorFlow / Keras, manipulação de dados, visualização.
  - Definição de parâmetros: tamanhos de batch, imagem, épocas, etc.
- **Preparação e pré-processamento dos dados**
  - Carregamento das imagens na forma de tensores.
  - Criação de objeto DataSet da biblioteca Tensorflow a partir das imagens e rótulos
  - O redimensionamento e a normalização [-1,1] dos canais estão inclusos na arquitetura da rede.
  - Não foi aplicada estratificação. Recorreu-se a Leave One Out Cross-Validation no pipeline.
  - O tamanho dos batches foi estabelecido como um campo ajustável na função de recursos do LOOCV.
  - A saída regressora permite que os rótulos sejam mantidos nas proporções originais (embora caiba testes futuros com normalização log1p).
- **Carregamento de modelo pré-treinado**
  - Carregamento da rede MobileNet com pesos pré-treinados com ImageNet, sem a camada totalmente conectada (apenas backbone)
  - Adição de uma nova saída com duas camadas: GlobalAveragePooling2D (para vetorizar a saída das convoluções) e Dense com ativação Softplus, que garante previsões quantitativas positivas.

- **Congelamento das camadas base (feature extractor)**
  - As camadas iniciais (responsáveis por extrair características gerais) são mantidas fixas (pesos não treináveis) para manter o conhecimento aprendido.

- Apenas as camadas novas (“cabeça”) são treinadas inicialmente para o novo conjunto de dados.

- **Treinamento inicial**
  - Treinar apenas as camadas finais (cabeça) com os dados novos, mantendo a base congelada.
  - Monitorar métricas de desempenho (acurácia, perda) no treino e validação.
- **Fine-tuning / ajuste fino**
  - Descongelar algumas camadas superiores da parte base para que sejam ajustadas aos dados específicos.
  - Treinar com uma taxa de aprendizado menor, para evitar “quebrar” as representações gerais já aprendidas.
  - Número de épocas foi mantido ajustável tanto no Fine-Tuning quanto no treinamento inicial da cabeça, como hiperparâmetros testáveis recursivamente. Estabelece-se critério de parada de acordo com número de épocas sem melhora na acurácia
- **Avaliação e gráficos**
  - Devido ao baixíssimo número amostral, decidiu-se por aplicar estratégia "Leave One Out Cross-Validation": O modelo é treinado n vezes e, a cada iteração (fold), uma observação é usada como validação e as demais para treino.
  - Cada fold resulta em um gráfico de comparação de acurácia. Como não há variância no extrato de validação com apenas uma observação, não foi calculado R^2 nas iterações, mas foi armazenado o valor de previsão que, pareado como valor verdadeiro, possibilitou calcular um R^2 geral por fora do fold (Out Of Fold - OOF).
  - Também estão disponíveis MAE e RMSE OOF.

7.2. Random Search e Deploy

O LOOCV foi encapsulado num processo iterativo com os seguinte hiperparâmetros ajustáveis:

- Épocas de treinamento da cabeça
- Épocas de treinamento no fine-tuning
- Taxa de aprendizagem de treinamento da cabeça
- Taxa de aprendizagem de treinamento no fine-tuning
- Número de camadas descongeladas para o fine-tuning

Cada iteração seleciona aleatoriamente, dentro de listas de valores predeterminados, uma combinação de hiperparâmetros para treinar iniciar a rede e processar o LOOCV, armazenando um valor R^2 OOF (explicado a seguir) para os hiperparâmetros testados. Hiperparâmetros já armazenados não são testados novamente.

Incluiu-se critérios de parada de acordo com número de iterações (trials), tempo de processamento e número de combinações selecionadas repetidamente, bem como sistemade checkpoint em arquivo CSV.

A melhor combinação de hiperparâmetros que atinge melhores resultados segundo R^2 é utilizada para treinar o modelo uma vez com todo o DataSet. Esse modelo é armazenado como um arquivo Keras, portátil e reutilizável em futuras previsões.

7.3. Avaliação do modelo

O intuito é avaliar modelos treinados que vieram do processo de transfer learning / fine-tuning usando métricas apropriadas, comparar desempenho, interpretar erros e decidir se o modelo é adequado para o problema.

Abaixo estão as etapas do fluxo seguido disponível no github:

- **Importações e configuração**
  - Importação de bibliotecas para avaliação (por exemplo sklearn.metrics, matplotlib, etc.).
  - Recuperar os modelos já treinados (métricas por fold e oof para uma combinação de hiperparâmetros).
- **Cálculo de métricas de desempenho**
  - Criação de gráfico de comparação de Erro Absoluto Médio (MAE) ente treino e validação para cada fold.
  - Gráficos de comparação "Predito vs Real", "Resíduos vs Ajustado", "Distribuição de Resíduos" e "Quantil-Quantil"
  - **Relatório de treinamento por fold**: De acordo com as imagens abaixo foram feitas

as seguintes observações:

- MAE de validação muito baixo em comparação ao de treino.
- As curvas de validação por fold são inerentemente ruidosas em LOOCV, pois cada validação baseia-se em uma única observação.
- Na maioria dos folds houve pouca mudança no MAE de treino e leve oscilação no MAE de validação, coerente com o cenário de validação unitária e parada antecipada, com eventuais folds atípicos por conta da dificuldade/facilidade da amostra deixada de fora.
- Pouca variação na maioria dos folds, com parada antecipada
- Esperava-se 10 épocas, mas apenas 5 folds chegaram ou passaram da época 8.
- Eventuais folds com comportamento divergente refletem a heterogeneidade das amostras deixadas de fora e são esperados em LOOCV.

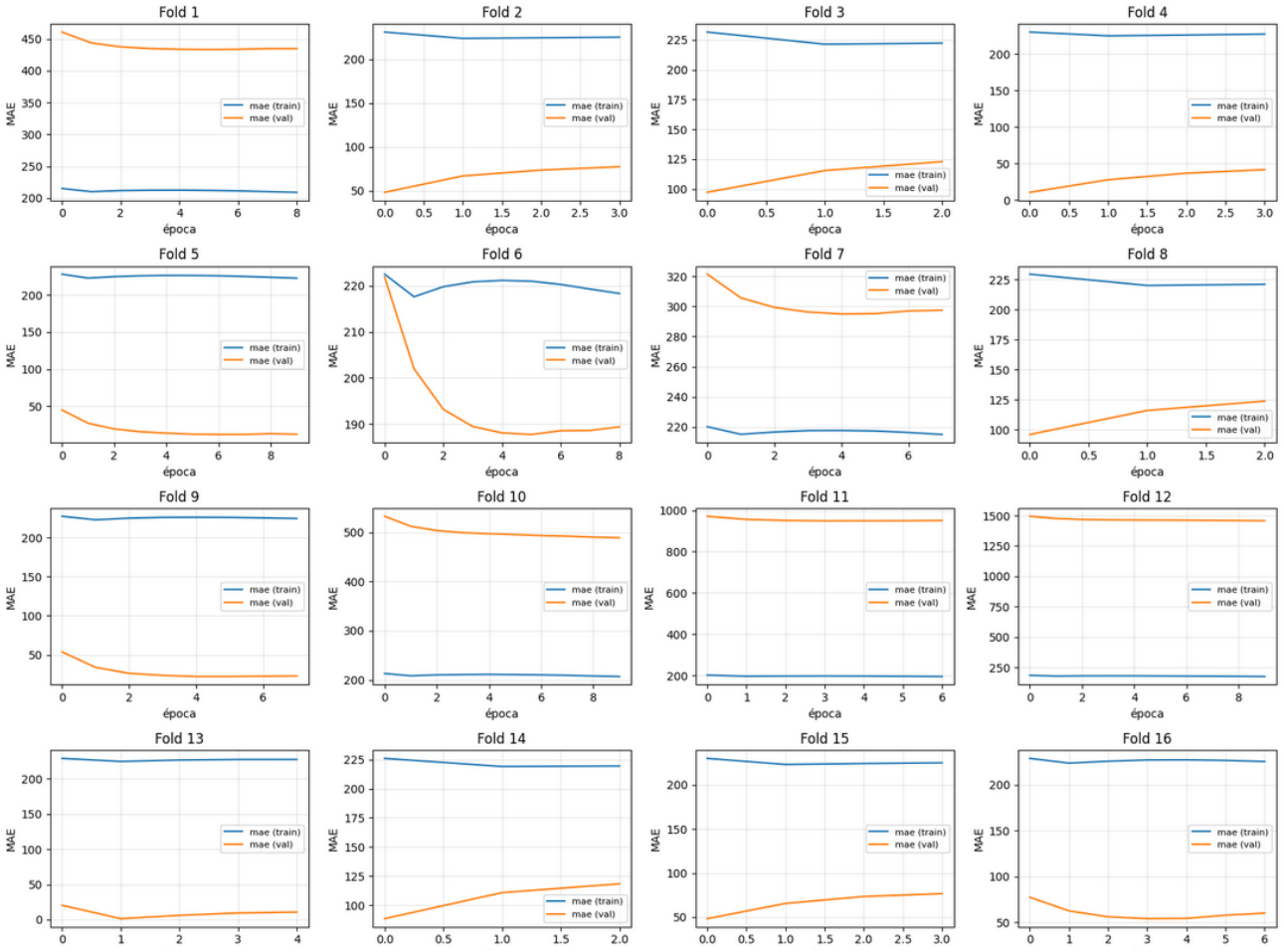

Figura 11: Gráficos 1 Fold

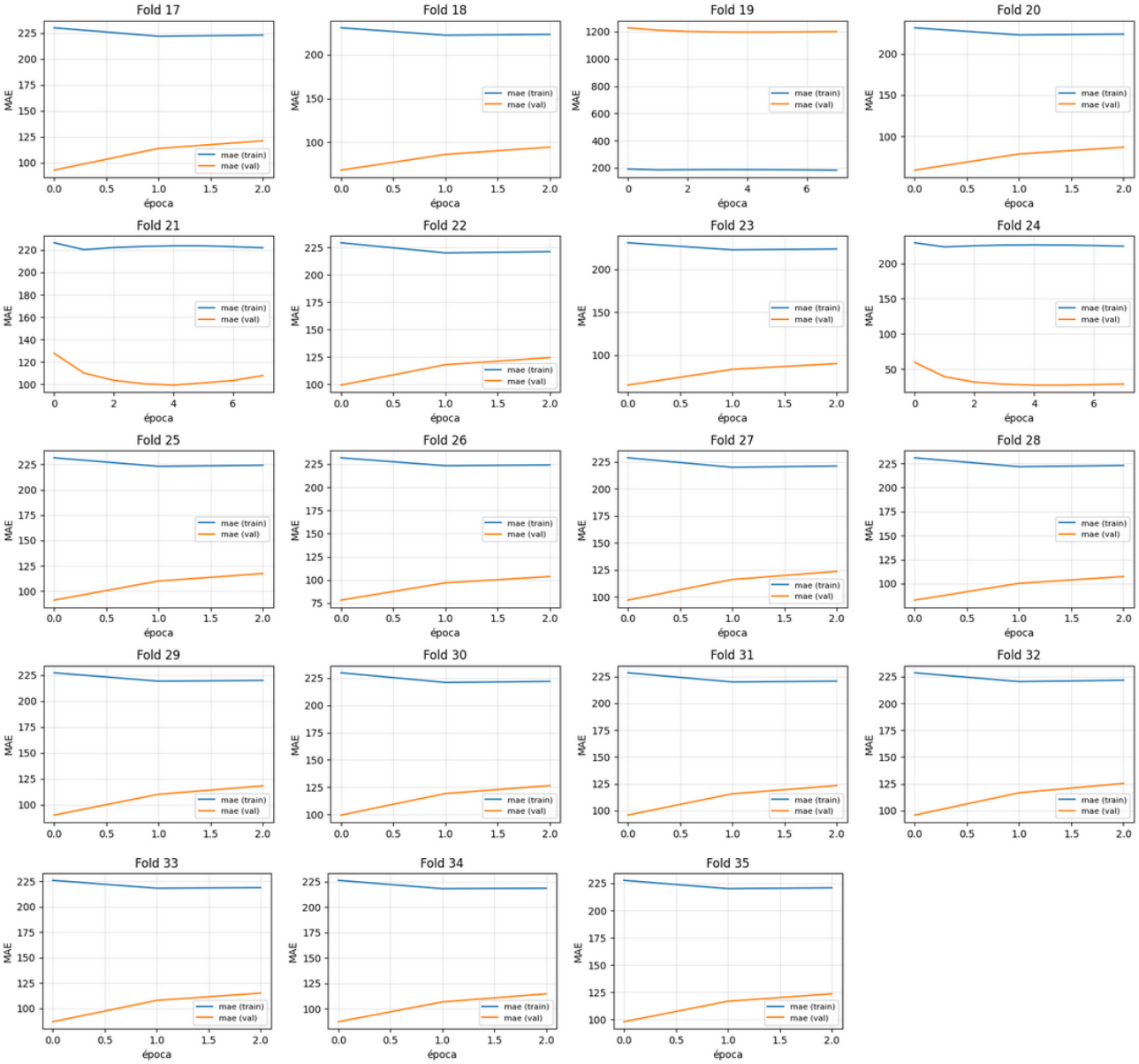

Figura 12: Grafico 2 Fold

- **Interpretação de erros / análise qualitativa**
  - Reporta-se R^2 agregado a partir das predições out-of-fold, por refletirem desempenho fora da amostra de forma mais estável do que curvas por época.

	y_true_oof	y_pred_oof	res
count	35.000000	35.000000	35.000000
mean	211.657143	113.102352	98.554791
std	386.796644	17.454937	377.162785
min	0.000000	86.904076	-99.441704
25%	2.000000	99.772491	-91.948902
50%	42.000000	107.296326	-64.746445
75%	177.500000	132.437737	40.570526
max	1594.000000	145.629242	1457.528519

Figura 13: Métricas

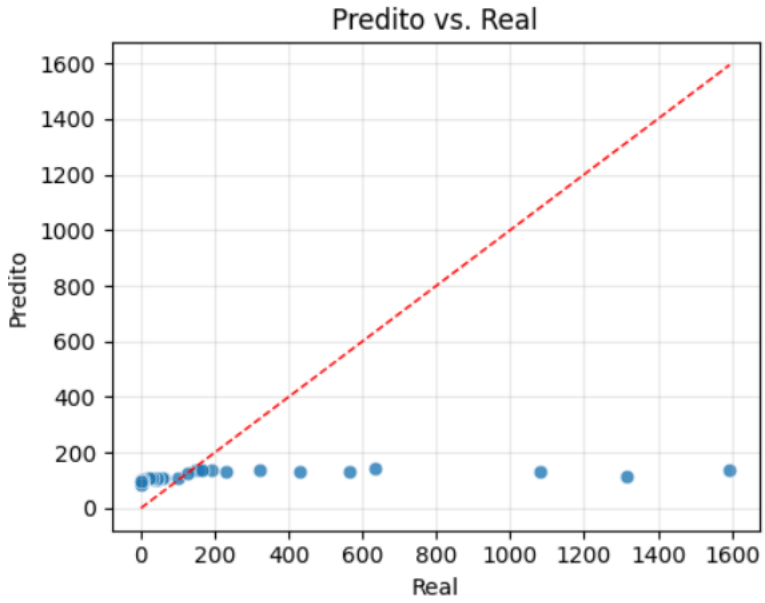

Figura 14: Predito vs Real

As predições fora dos folds foram comparadas aos valores reais; o espalhamento em torno da linha y=x quantifica viés e calibração. O modelo parece estar viesado para uma previsão em torno de 0 e 200, apresentando baixíssima correlaçao entre o real e o previsto. Isso indica compressão das predições: o modelo quase não varia a saída e não acompanha a amplitude do alvo , o que caracteriza subajuste/encolhimento para a média e má calibração.

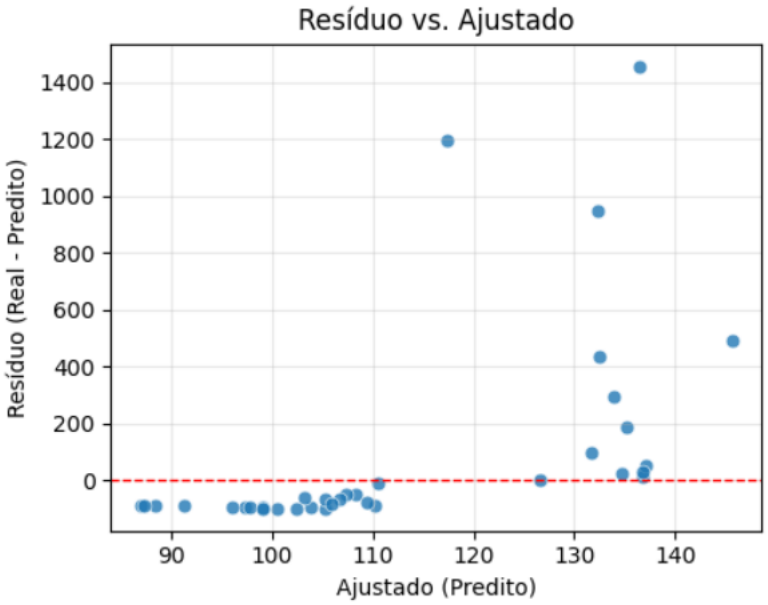

Figura 15: Resíduo vs Ajustado

Resíduo vs. Ajustado mostra o erro em função do valor previsto.

Não parece haver linearidade entre o erro e o valor previsto, mas observa-se que cerca de 20% das previsões com erros mais discrepantes estão acimada da média da previsão.

O padrão indica viés por faixa e heterocedasticidade: o modelo superestima quando o valor previsto é baixo (resíduos negativos) e passa a subestimar quando é alto (resíduos positivos), com a variância do erro abrindo em “leque” à medida que cresce.

Em termos visuais: concentração de erros negativos nas faixas baixas e erros positivos muito dispersos nas faixas altas sugerem que o modelo não acompanha as caudas e pode estar subespecificado nessa região.

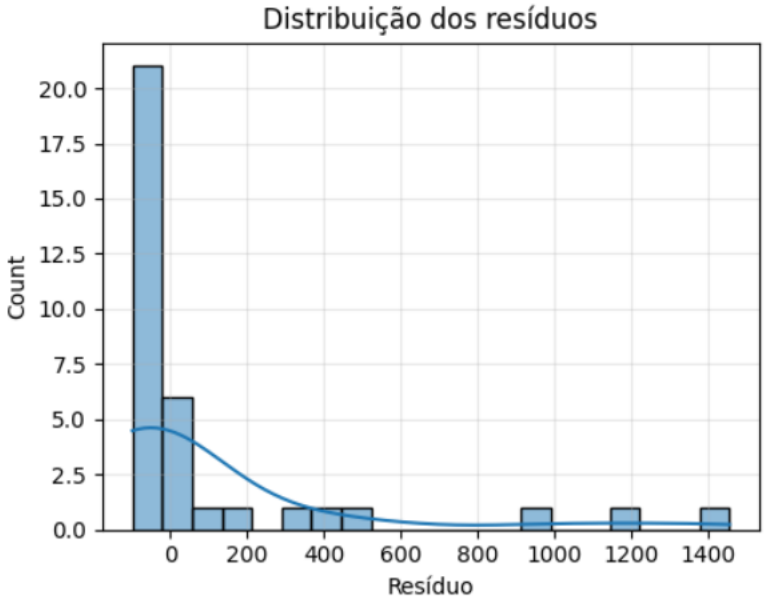

Figura 16: Distribuição de Resíduos

O histograma dos resíduos (OOF) mostra concentração entre −200 e 0, indicando viés de

sobreprevisão na maioria das observações, alguns casos de subprevisão moderada (0 a 100) e poucos erros mais dispersos, que poderiam ser considerados outliers em um tamanho amostral mais confiável.

A distribuição é assimétrica para o lado negativo, consistente com viés por faixa observado no gráfico Resíduo vs. Ajustado; alguns erros mais raros elevam a variabilidade e explicam a diferença entre MAE e RMSE mais abaixo:

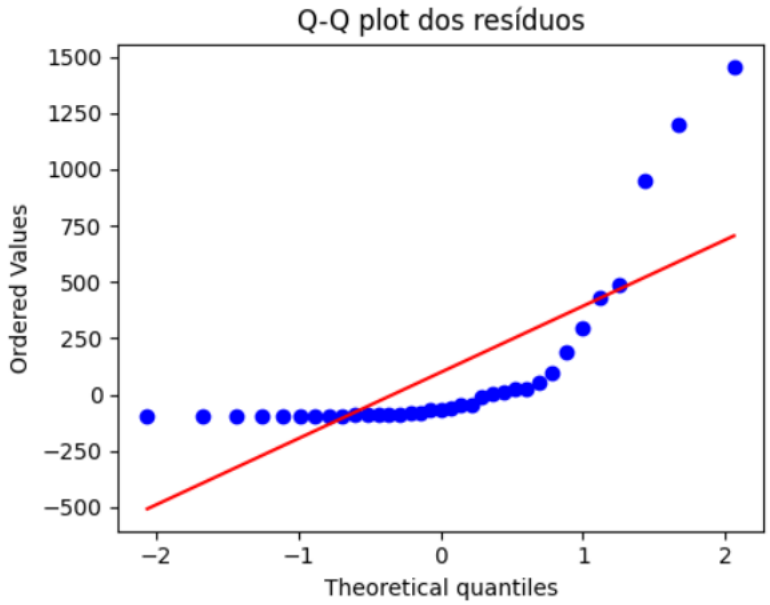

Figura 17: Q-Q plot dos Resíduos

- Theoretical quantiles: são os quantis da distribuição normal teórica.
- Ordered values: são os quantis dos resíduos, ordenados.

Se os resíduos seguissem uma distribuição normal, estariam mais próximos da linha de referência.

O Q-Q plot dos resíduos mostra desvio substancial da normalidade (padrão em 'S' e cauda direita acima da referência), sugerindo erros assimétricos e com cauda pesada, o que afeta a calibragem de intervalos e testes que assumem normalidade.

Medidas OOF:

- MAE (OOF): 200.0684790475028
- RMSE (OOF): 384.57831039922826
- R^2: -0.017637979847209273

Erro absoluto Médio (fora dos folds):

As predições se desviam em média 200 CPA (MAE=200.07) do valor real em validação fora da amostra.

Raiz do Erro Quadrático Médio (fora dos folds):

Esta medida é mais sensível a outliers que o MAE. Um resultado acima de 50% maior (RMSE=384.58) indica outliers puxando o desempenho para baixo.

O R^2 OOF foi -0.018, indicando desempenho marginalmente inferior ao preditor da média, consistente com a compressão das predições observada no gráfico Predito vs. Real e com a heterocedasticidade nos resíduos; reportamos também MAE OOF para interpretar o erro em unidades de cliques.

7.4. Avaliação do Produto

A despeito dos esforços metodológicos empregados — incluindo LOOCV e Random Search —, as limitações de capacidade computacional (tempo de treinamento e memória) e, principalmente, a baixa massa de dados disponível restringiram a qualidade preditiva alcançada neste ciclo. Ainda assim, o produto técnico entregue é um pipeline reprodutível e portável (pré-processamento, treinamento, validação OOF e geração de métricas/relatórios), pronto para ser reaplicado quando houver maior disponibilidade de dados e hardware, exigindo apenas ajustes mínimos de configuração para evoluir o modelo e incorporar novas fontes, reduzindo o lead time de futuras iterações.

Abaixo estão as etapas do fluxo seguido disponível no github:

- Carrega os modelos já treinados (com e sem fine-tuning).
- Compara as métricas (acurácia, f1, perda, etc.) para identificar o melhor.
- Salva o modelo vencedor em formato pronto para deploy (.h5 ou SavedModel).
- Cria funções de inferência que recebem uma imagem e retornam a predição.
- Testa o modelo empacotado com novos dados para verificar a consistência.
- Documenta o fluxo de uso (como carregar e prever).

8. Storytelling

8.1. Proposta de valor

Criativos decidem boa parte do resultado em performance, mas testes A/B são caros e lentos. Aqui, usamos visão computacional para transformar o conteúdo visual das peças em um sinal preditivo de eficiência. Em outras palavras: em vez de testar tudo às cegas, passamos a ranquear e priorizar o que tem mais chance de performar antes de investir verba.

8.2. Contexto

Ao longo de campanhas, acumulamos peças estáticas com métricas de mídia (Resultados e Valor usado). O desafio não era só medir o que funcionou, e sim entender o porquê e prever quais próximas peças devem performar melhor. A pergunta central de negócio foi:

“Quais padrões visuais reduzem o CPA, trazem mais resultados e como antecipar isso nos próximos testes?”

8.3. Objetivo

Nosso objetivo: explicar quais características visuais aparecem nas peças mais eficientes; prever uma métrica de eficiência diretamente a partir das imagens; e operacionalizar esse aprendizado para orientar a priorização de mídia e o briefing para novas criações, reduzindo tentativa-e-erro. Nesta primeira iteração, o foco é provar o pipeline e medir limites atuais; a identificação de padrões visuais ficará para a próxima rodada após ajustes.

8.4. Dados

Trabalhamos com um conjunto de 35 imagens, cada uma associada a Resultados, Valor usado (BRL), Nome do anúncio e tipo de resultado (compras no site). A métrica-alvo foi definida como CPA = Valor usado / Resultados, com regras claras para casos de Resultados igual a zero. As imagens foram padronizadas (tamanho e normalização) para garantir comparabilidade e evitar vieses de pré-processamento.

8.5. Abordagem

Usamos uma **rede neural convolucional (CNN) MobileNet** pré-treinada no **ImageNet** como extratora de atributos visuais (include_top=False, pooling global). Sobre ela, acoplamos uma cabeça de regressão para estimar diretamente a métrica-alvo. O treino ocorreu em duas fases: primeiro com o **backbone congelado** (treina apenas a cabeça, mais estável com pouco dado); depois, um **fine-tuning** parcial descongelando camadas finais com taxa de aprendizado menor para ajustar a rede ao nosso domínio

visual.

8.6. Validação

Para evitar conclusões frágeis com amostra pequena, usamos **Leave-One-Out Cross-Validation (LOOCV)**: a cada rodada, treinamos com 34 imagens e testamos na imagem restante, repetindo até todas terem sido previstas em hold-out. Isso nos dá um retrato honesto do desempenho fora da amostra. Aplicamos EarlyStopping para evitar overfitting, preservando os melhores pesos por fold.

8.7. Resultados

Avaliação do modelo (OOF) — LOOCV agregado

- MAE ≈ 200, RMSE ≈ 385, $R^2 \approx -0{,}02$.
- Predito vs. Real: previsões comprimidas (~90–150) e distantes da linha y=x → subajuste/baixa calibração.
- Resíduo vs. Ajustado: heterocedasticidade (padrão em leque); superestima em valores baixos e subestima em altos.
- Distribuição/QQ-plot: cauda pesada e desvio da normalidade.

Conclusão: o modelo atual não captura a amplitude do alvo; precisa de ajustes de alvo/dados/head antes de uso automático.

8.8. Evidências atuais e limitações

As curvas por fold são curtas e ruidosas (LOOCV). Não há evidência robusta de padrões visuais vencedores; o erro concentra-se por faixa do previsto, indicando subespecificação do modelo.

8.9. Impacto para o negócio

O pipeline já permite processar e comparar criativos de forma objetiva, mas as estimativas ainda não devem guiar investimento sozinhas. Deve ser usado como triagem exploratória e para institucionalizar o processo de avaliação.

8.10. Recomendações práticas

Para as próximas rodadas: priorizar enquadramento com foco no produto, fundo limpo e contraste marcante; limitar texto e evitar poluição visual; reaproveitar estéticas das vencedoras encontrando peças “irmãs” e manter um painel operacional com CPA por imagem + estimativa do modelo atualizados a cada ciclo.

8.11. Limitações

A amostra ainda é pequena (N=35), o que exige cautela nas generalizações; o LOOCV reduz viés, mas não o elimina. Existe viés de gasto (peças com mais verba tendem a acumular mais resultados), por isso analisamos e comparamos por CPA e, quando possível, por faixas de investimento. Casos com Resultados=0 precisam de tratamento consistente para evitar distorções.

8.12. Próximos Ajustes

Avaliar alvo com transformação log1p para reduzir assimetria e estabilizar a variância, bem como testar outras métricas como variável alvo.

Testar cabeça com outras configurações, como: ativação exponencial e loss calculado na distribuição de Poisson.

Estudar a viabilidade e o quão conceitualmente correto seria o ajuste do modelo com DataSet maior, cujos rótulos sejam quantitativos, ainda que de classes díspares às da natureza dos dados do stakeholder, e avaliar com o DataSet disponível atualmente.

8.13. Fechamento

Embora o resultado não seja assertivo e implantável, o pipeline foi desenvolvido com sucesso, incluindo mensuração de acurácia e portabilidade.

8.14. Próximos Passos

Visão

Transformamos o pipeline em um serviço de scoring que recebe imagens e devolve uma estimativa de métrica (hoje, CPA previsto), além de metadados de execução. O objetivo é operacionalizar a priorização de criativos em fluxo de trabalho real, com versionamento e monitoramento.

Contrato de I/O (API & batch)

- Entrada: image (arquivo ou URL), model_version, campaign_id (opcional).
- Saída: y_pred (valor previsto), prediction_interval (opcional), model_version, preprocess_hash, latency_ms, trace_id.

- Batch: CSV/Parquet com file_path → retorna arquivo com file_path, y_pred, model_version, scored_at.

Governança & versionamento

Cada entrega traz model_version (ex.: v0.2.1), data_window, target_def, preprocess_hash e model card resumido (métricas OOF, limitações, data de treino). Publicação só ocorre se passar por gates mínimos (ex.: MAE_OOF ≤ baseline−δ e R^2_OOF ≥ 0).

Monitoramento em produção

- Saúde: latência, taxa de erro e timeouts.
- Métricas de predição: distribuição de y_pred vs. janela histórica (drift por KS/PSI).
- Recalibração: quando existir rótulo real novo, calcular MAE/RMSE OOS e alertar se piorar > X%.
- Re-treino: gatilhos por volume de dados, drift e performance abaixo do gate.

Calibração e uso responsável

Dado o estado atual (compressão, $R^2 \approx 0$), o serviço roda em modo “assistido”: o score não decide sozinho; ele ranqueia para triagem e sugere um conjunto de teste. Para decisões automáticas, exigimos atingir os gates acima em nova iteração (ex.: usar log-CPA, perdas robustas e features auxiliares).

Integração no fluxo de mídia

- Pré-brief: usar o serviço para montar uma lista curta de peças a ativar primeiro.
- Pós-campanha: coletar Resultados/Valor usado, fechar rótulos, atualizar dataset e re-treinar.
- Painel: exibir y_pred por criativo, model_version, e alertas de drift

8.15. Video Storytelling

Link do youtube: https://youtu.be/LZRewP9Y-lk

9. REPOSITÓRIO

Todo material desenvolvido para este projeto poderá ser encontrado em: https://github.com/ISABELAKAZUE/ProjetoII

Como reproduzir o projeto:

1. Clone o repositório
2. Navegue até a pasta do projeto:

3. Instale as dependências necessárias listadas em `requirements.txt`;
4. Prepare os dados para ajuste do modelo na pasta `data/raw/for_fit/`;
5. Prepare as imagens, cujas métricas se deseja, prever na pasta `data/raw/for_predict/`;
6. Ajuste os parâmetros iniciais do arquivo `main.ipynb` na pasta `notebooks/` e execute-o;
7. Avalie os resultados no próprio arquivo `main.ipynb` ou dentre os relatórios em `reports/`.

7.1. Diretórios

```
Project-root/
 │
 ├── data/
 │ ├── raw/ # Dados brutos, imagens originais e metadados não processados
 │ │  ├── for_fit/ # Dados para treino e teste do modelo
 │ │  └── for_predict/ # Imagens a serem previstas
 │ └── processed/ # Dados pré-processados prontos para uso (matrizes, features)
 │
 ├── notebooks/ # Jupyter notebooks para exploração, análise e testes
 │
 ├── src/ # Código-fonte do projeto
 │ ├── data_preprocessing/ # Scripts para carregamento e limpeza dos dados,
 │ │  pré-processamento de imagens
 │ ├── feature_extraction/ # Scripts para extração de features visuais e
 │ │  textuais
 │ ├── models/ # Definição e treinamento dos modelos de deep learning
 │ ├── evaluation/ # Scripts para avaliação de métricas e análise dos
 │ │  resultados
 │ ├── utils/ # Funções utilitárias e helpers gerais
 │ └── inference/ # Código para gerar predições com modelos treinados
 │
 ├── experiments/ # Configurações, logs e resultados de experimentos de
 │  modelagem
 │
 ├── reports/ # Documentação do projeto, relatórios e apresentações
```

```
│
├── requirements.txt # Lista de dependências necessárias para rodar o projeto
├── README.md # Apresentação do projeto e instruções principais
└── .gitignore # Arquivos e pastas ignorados pelo Git
```

8. CRONOGRAMA

Abaixo encontra-se o cronograma preliminar das atividades que serão feitas durante o projeto com cada duração em dias, porém o cronograma será atualizado conforme andamento do projeto:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Etapa 1																														
Definição do Grupo																														
Reunião de KickOff																														
Recebimento dos dados da Empresa																														
Preparação da Documentação do Projeto																														
Extração de dados																														
Reunião final de definição																														
Etapa 2																														
Definição da linguagem de programação usada no projeto.																														
Análise exploratória da base de dados escolhida.																														
Tratamento da base de dados (Preparação e treinamento).																														
Definição e descrição das bases teóricas dos métodos.																														
Definição e descrição de como será calculada a acurácia.																														
Etapa 3																														
O método analítico definido na etapa anterior aplicado à base de dados escolhida.																														
Medidas de acurácia, usando os métodos definidos na etapa anterior.																														
Descrição dos resultados preliminares, apresentando um produto gerado e rascunhando um possível modelo de negócios.																														
Esboço do storytelling.																														
Etapa 4																														
O relatório técnico com todos as fases do projeto.																														
A apresentação do storytelling. Pode ser um arquivo PPT ou similares.																														
Definição do repositório no GitHub com todos os arquivos do projeto.																														
Disponibilização e edição de um vídeo com a apresentação no YouTube.																														

Figura 11: Cronograma

8.1. Milestones da Etapa 1

- Reunião de KickOff com o grupo para definição da organização e objetivos.
- Elaboração do documento com detalhes e objetivos do projeto em formato de artigo acadêmico.
- Repositório criado e sincronizado com todos os integrantes da equipe;
- Definição e extração de dados que serão utilizados tanto em texto quanto em imagem.
- Cronograma com tarefas e duração.

8.2. Milestones Etapa 2

- Definição da linguagem de programação
- Definição da base que será utilizada
- Tratamento da base de dados
- Analise dos dados retirados da base
- Definição e descrição das bases teóricas do modelo

8.3. Milestones Etapa 3

- Aplicação do Método Analítico
- Medidas de Acurácia
- Descrição dos Resultados Preliminares
- Esboço do Storytelling

8.4. Milestones Etapa 4

- Edição do Relatório Técnico
- Criação da apresentação do storytelling
- Atualização no repositório
- Edição e produção do video da apresentação.

9. REFERÊNCIAS

CNN BRASIL. Dia do Hambúrguer: Dados e tendências mostram presença do lanche no país. 2025. Disponível em: <https://www.cnnbrasil.com.br/viagemegastronomia/gastronomia/dia-do- hamburguer-dados-e-tendencias-mostram-presenca-do-lanche-no-pais/>. Acesso em: 5 set. 2025.

TENSORFLOW. Rede Neural Convolucional (CNN) | TensorFlow Core. 2024. Disponível em: <https://www.tensorflow.org/tutorials/images/cnn?hl=pt-br>. Acesso em: 02 out. 2025.

APVENDA. Métricas do Marketing Digital – CPA, ROAS, CPM, CPC, CTR. 2024. Disponível em: <https://apvenda.com.br/metricas-do-marketing-digital-cpa-roas-cpm-cpc-ctr/>. Acesso em: 5 set. 2025.

IBM Brasil. O que são redes neurais convolucionais?. 2025. Disponível em: <https://www.ibm.com/br-pt/think/topics/convolutional-neural-networks>. Acesso em: 02 out. 2025.

GANESH, Prakhar. Types of Convolution Kernels: Simplified – An intuitive introduction to different variations of the glamorous CNN layer. Towards Data Science, 2019. <https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37>. Acesso em: 02 out. 2025.

JUNIOR, Arnaldo de Carvalho. Métricas de Desempenho em Modelos de IA (parte 2). [S.l.], 2025. Disponível em: <https://eailab.labmax.org/2025/05/13/metricas-de-desempenho-em-modelos-de-ia-parte-2/>. Acesso em: 3 out. 2025.

GOLESTANEH, Pegah; TAHERI, Mahsa; LEDERER, Johannes. How many samples are needed to train a deep neural network? ICLR Conference, 2025. Disponível em: <https://arxiv.org/abs/2405.16696>. Acesso em: 3 out. 2025.

ENCORD. Training, Validation, Test Split for Machine Learning Datasets. 2023. Disponível em: https://encord.com/blog/train-val-test-split. Acesso em: 3 out. 2025.