UNIVERSIDADE FEDERAL DO MARANHÃO

BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA

INTELIGÊNCIA ARTIFICIAL

CLASSIFICAÇÃO DE ÍRIS: UMA ABORDAGEM COM REDES NEURAIS E KERAS

João Henrique Calixto Teixeira

João Victor Lima Salomao

Juan Pablo Furtado Mondego

São Luís

2025

1. Introdução

O avanço na área de Inteligência Artificial (IA) tem revolucionado diversos setores, desde o reconhecimento de imagens até a análise preditiva de grandes volumes de dados. Entre as diferentes abordagens de IA, as **Redes Neurais Artificiais (RNAs)** têm se destacado por sua capacidade de aprender padrões complexos de maneira automática (Haykin, 1999). As RNAs são inspiradas na forma como o cérebro humano processa informações, utilizando neurônios artificiais e conexões ponderadas para classificar, prever e reconhecer correlações em bases de dados (Nielsen, 2015).

O presente trabalho tem como objetivo aplicar uma Rede Neural Artificial ao clássico conjunto de dados **Iris**, proposto inicialmente por Fisher (1936), a fim de demonstrar a eficiência desse modelo em tarefas de classificação multiclasse. O dataset Iris contém 150 instâncias de flores, distribuídas em três espécies (Setosa, Versicolor e Virginica), cada qual descrita por quatro características: comprimento e largura da pétala, comprimento e largura da sépala. Por se tratar de um problema considerado “fácil”, o Iris é frequentemente usado como benchmark para validar novos algoritmos de classificação e ilustrar conceitos fundamentais em Machine Learning (Dua & Graff, 2019).

Neste artigo, serão abordados os principais fundamentos de redes neurais, desde sua inspiração biológica até a execução prática com a biblioteca **Keras**, evidenciando as camadas, funções de ativação e a função de custo adequadas à classificação multiclasse. Além disso, serão apresentados resultados experimentais que demonstram a taxa de acerto (acurácia), a matriz de confusão e outras métricas importantes, com análises qualitativas e quantitativas do desempenho do modelo.

Para tornar a apresentação mais didática e organizada, estruturamos este documento em seções detalhadas. Na subseção **1.1**, descrevem-se as características essenciais das **Redes Neurais Artificiais**, relacionando o modelo biológico às técnicas computacionais (Rumelhart, Hinton & Williams, 1986;). Em **1.2**, introduz-se a biblioteca **Keras**, utilizada para a implementação prática e que se destaca pela simplicidade de uso na construção de modelos de Deep Learning (Chollet, 2018;). As seções subsequentes abrangem a metodologia adotada, a descrição do conjunto de dados e os experimentos realizados, culminando na apresentação dos resultados, discussão e conclusão.

1.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) surgiram a partir de uma analogia ao funcionamento do cérebro humano, no qual bilhões de neurônios se interconectam para processar informações de forma paralela e distribuída (McCulloch & Pitts, 1943). Cada neurônio artificial recebe entradas associadas a pesos, realiza uma combinação linear desses sinais e aplica uma função de ativação para gerar a saída (Haykin, 1999). Essa capacidade de aprendizado ocorre por meio do ajuste dos pesos sinápticos com base em algoritmos de retropropagação do erro (Rumelhart, Hinton & Williams, 1986), que minimizam uma função de custo específica do problema, como a entropia cruzada no caso de classificações (slides-12).

Ao contrário de métodos tradicionais de aprendizado de máquina, que frequentemente exigem engenharia de atributos manual, as RNAs são capazes de extrair representações dos dados de forma mais autônoma, sobretudo quando se empregam redes profundas (Deep Learning). Esse enfoque garante maior flexibilidade e um poder de generalização superior em diversos cenários, como reconhecimento de voz, processamento de texto e classificação de imagens (LeCun, Bengio & Hinton, 2015).

1.2 A Biblioteca Keras

A biblioteca **Keras** (Chollet, 2018) é uma **API de alto nível** para construção e treinamento de redes neurais em Python. Ela abstrai grande parte da complexidade subjacente aos frameworks de baixo nível, como TensorFlow ou Theano, tornando mais simples a elaboração de arquiteturas neurais customizadas (slides-12). Por meio de funções e classes intuitivas, o pesquisador ou estudante pode definir, compilar e ajustar modelos em poucas linhas de código, além de monitorar métricas e visualizações de treino em tempo real.

Entre as principais vantagens do Keras, destacam-se:

- **Simplicidade e Produtividade**: a sintaxe clara permite prototipagens rápidas.

- **Integração**: compatível com várias plataformas de Deep Learning, incluindo TensorFlow, possibilitando a utilização de GPU e TPU para otimizações de maior escala.

- **Comunidade Ativa**: rica documentação e grande volume de exemplos, tutoriais e aplicações em diferentes domínios.

No contexto do dataset Iris, a biblioteca Keras facilita a criação de um modelo neural multicamadas (feedforward) com funções de ativação como ReLU e Softmax, adequadas para tarefas de classificação multiclasse (Chollet, 2018;). Dessa forma, obtém-se uma ferramenta didática para ilustrar conceitos teóricos ao mesmo tempo em que se atinge alto desempenho em um problema amplamente estudado.

2 Metodologia

O desenvolvimento deste trabalho seguiu os seguintes passos: i) mensuração de dados, ii) tratamento de dados, iii) desenvolvimento do modelo, iv) treinamento do modelo e v) validação do modelo. A metodologia empregada segue uma sequência lógica de etapas, desde a coleta e preparação dos dados até a avaliação do modelo treinado, mantendo um padrão em todos os passos. Em resumo, a metodologia apresentada neste projeto demonstra uma abordagem sistemática para a classificação de flores Íris utilizando redes neurais, abrangendo desde a preparação dos dados até a avaliação do modelo, com o objetivo de construir um classificador preciso e robusto.

2. 1 Mensuração de dados

Nesta etapa inicial, o foco principal é a obtenção e a análise exploratória dos dados. O conjunto de dados Iris, amplamente utilizado em aprendizado de máquina, é carregado usando a biblioteca sklearn.datasets do Python. Esse conjunto contém informações sobre três espécies de flores Iris (Setosa, Versicolor e Virginica), com quatro características (comprimento e largura da sépala e pétala) medidas para cada flor. O objetivo dessa etapa é coletar e analisar os dados brutos do conjunto de dados Íris para entender a natureza dos dados e preparar para as etapas subsequentes e para isso o processo é feito da seguinte forma:

Primeiro, o conjunto de dados Iris é carregado utilizando a função *load_iris()* da biblioteca *sklearn.datasets*.

As características das flores (variáveis preditoras) são armazenadas na variável **x**, enquanto as classes correspondentes (variável alvo) são armazenadas na variável **y**.

Para obter uma compreensão visual da distribuição dos dados, gráficos de dispersão são gerados utilizando as bibliotecas matplotlib.pyplot e seaborn. Esses gráficos permitem a visualização da relação entre as características das flores e suas respectivas classes, auxiliando na identificação de padrões e tendências nos dados.

2.2 Tratamento de dados

O tratamento de dados tem como objetivo preparar os dados de entrada e saída para que o modelo de machine learning possa aprender de forma eficaz e ter seu desempenho avaliado corretamente. Isso é feito por meio de categorização das saídas, normalização das entradas e divisão dos dados em conjuntos de treinamento e teste.

2.2.1 Categorização das saídas

Para adequar as saídas ao formato necessário para o treinamento de modelos de classificação multiclasse, as etiquetas das espécies de Iris (setosa, versicolor e virginica) foram convertidas em um vetor binário utilizando a função *to_categorical* da biblioteca Keras (Chollet, 2015). Essa função transforma as etiquetas de classe em uma representação one-hot encoding, onde cada classe é representada por um vetor binário, com o valor 1 indicando a classe à qual a amostra pertence e 0 para as demais classes. Essa abordagem é essencial para o treinamento de redes neurais em problemas de classificação com múltiplas classes.

2.2.2 Normalização das entradas

As entradas, correspondentes às características das flores (comprimento e largura das pétalas e sépalas), foram normalizadas utilizando o método *MinMaxScaler* da biblioteca scikit-learn (Pedregosa et al., 2011). Este método redimensiona os valores das características para um intervalo entre 0 e 1, garantindo que todas as características tenham a mesma escala de magnitude. A normalização é uma etapa importante no pré-processamento de dados, pois contribui para a estabilidade e a convergência mais rápida do processo de treinamento do modelo.

2.2.3 Separação do conjunto de dados

O conjunto de dados foi dividido em dois subconjuntos: treino e teste. Utilizou-se a função *train_test_split* da biblioteca scikit-learn para realizar esta divisão, com uma proporção de 80% dos dados para treinamento e 20% para teste. Adicionalmente, o parâmetro *stratify* foi utilizado para garantir uma distribuição balanceada das classes em ambos os subconjuntos. Esta separação permite avaliar a capacidade de generalização do modelo, ou seja, sua capacidade de classificar corretamente dados não vistos durante o treinamento.

2. 3 Desenvolvimento do modelo

Nesta etapa, o foco é a construção da estrutura da rede neural que será utilizada para classificar as flores Iris. A escolha da arquitetura da rede neural é crucial para o sucesso do projeto, pois influencia diretamente na capacidade do modelo de aprender e generalizar os padrões presentes nos dados. Essa etapa tem como objetivo definir a arquitetura da rede neural, especificando suas camadas, funções de ativação e outros parâmetros relevantes. Para isso foi utilizado o seguinte processo:

Primeiramente era necessário escolher um modelo e o selecionado foi o Multilayer Perceptron (MLP), uma rede neural artificial comumente utilizada para tarefas de classificação.

Depois, a arquitetura da rede neural foi definida utilizando a API Keras do TensorFlow. A rede neural consiste em:

- Uma camada de entrada (InputLayer) com 4 neurônios, correspondendo às 4 características das flores (comprimento e largura da sépala e pétala).

- Uma camada oculta (Dense) com 512 neurônios e função de ativação ReLU (relu). Essa camada é responsável por extrair características complexas dos dados de entrada. A inicialização dos pesos é feita de forma aleatória, utilizando RandomNormal com uma semente definida para reprodutibilidade.

- Uma camada de saída (Dense) com 3 neurônios e função de ativação Softmax (softmax). Essa camada produz as probabilidades de cada flor pertencer a uma das três classes (Setosa, Versicolor e Virginica).

2.4 Treinamento do modelo

Nesta etapa crucial, a rede neural definida na etapa anterior é treinada utilizando os dados de treinamento preparados. O processo de treinamento envolve a apresentação dos dados de entrada à rede neural e o ajuste iterativo dos pesos da rede para minimizar o erro de classificação. O objetivo é que o modelo aprenda os padrões presentes nos dados e seja capaz de generalizar para novos dados não vistos durante o treinamento. O objetivo é ajustar os pesos da rede neural utilizando os dados de treinamento para que o modelo aprenda a classificar as flores Iris com precisão. Para isso ocorreu o seguinte processo:

Antes do treinamento, o modelo é compilado utilizando a função compile(). Nessa etapa, são definidos:

- A função de perda (loss): *categorical_crossentropy*, adequada para problemas de classificação multiclasse.

- O otimizador (optimizer): *rmsprop*, responsável por atualizar os pesos da rede durante o treinamento.

- As métricas de avaliação (metrics): *categorical_accuracy*, utilizada para medir a precisão do modelo durante o treinamento e validação.

O treinamento é realizado utilizando a função fit(). Os dados de treinamento e validação são fornecidos como entrada, juntamente com o número de épocas (epochs) e outros parâmetros. Dentre as funções estão:

- *EarlyStopping*: Um callback *EarlyStopping* é utilizado para interromper o treinamento caso o desempenho do modelo na validação não melhore por um determinado número de épocas (patience), evitando o overfitting e restaurando os melhores pesos encontrados.

- *validation_split*: Uma porcentagem dos dados de treinamento é reservada para validação, permitindo monitorar o desempenho do modelo em dados não utilizados diretamente no treinamento.

2.5 validação do modelo

Após o treinamento da rede neural, é fundamental avaliar seu desempenho em dados que não foram utilizados durante o processo de treinamento. Essa etapa, conhecida como avaliação do modelo, permite verificar se o modelo aprendeu os padrões dos dados de forma eficaz e se é capaz de generalizar para novos dados, ou seja, classificar corretamente novas flores Iris. Com o objetivo de avaliar o desempenho do modelo treinado utilizando dados não vistos durante o treinamento, para determinar sua capacidade de generalização e precisão na classificação de novas flores Iris. Para isso foi utilizado o seguinte processo de avaliação:

1. O modelo treinado é avaliado utilizando o conjunto de dados de teste, que foi separado dos dados de treinamento e validação. A função *evaluate()* é utilizada para calcular a perda e a acurácia do modelo nos dados de teste.

2. O histórico do treinamento, armazenado durante a etapa de treinamento, é utilizado para gerar gráficos que mostram a evolução da perda e da acurácia ao longo das épocas de treinamento. Esses gráficos permitem visualizar o progresso do aprendizado do modelo e identificar potenciais problemas como overfitting ou underfitting.

3. Para demonstrar a capacidade do modelo de classificar novas flores Iris, um exemplo de dados de entrada é criado e a função *predict()* é utilizada para obter as probabilidades de cada classe. Os resultados são apresentados em termos de porcentagens para cada classe de flor.

3 Resultados

3.1 Distribuição dos dados: Visão geral

A análise exploratória dos dados foi conduzida através da visualização gráfica da distribuição das características das flores Iris. Utilizou-se a biblioteca *Seaborn* para gerar gráficos de dispersão, permitindo a observação de padrões e relações entre as variáveis. As espécies de Iris foram diferenciadas por cores para facilitar a interpretação.

3.1.1 Distribuição das pétalas

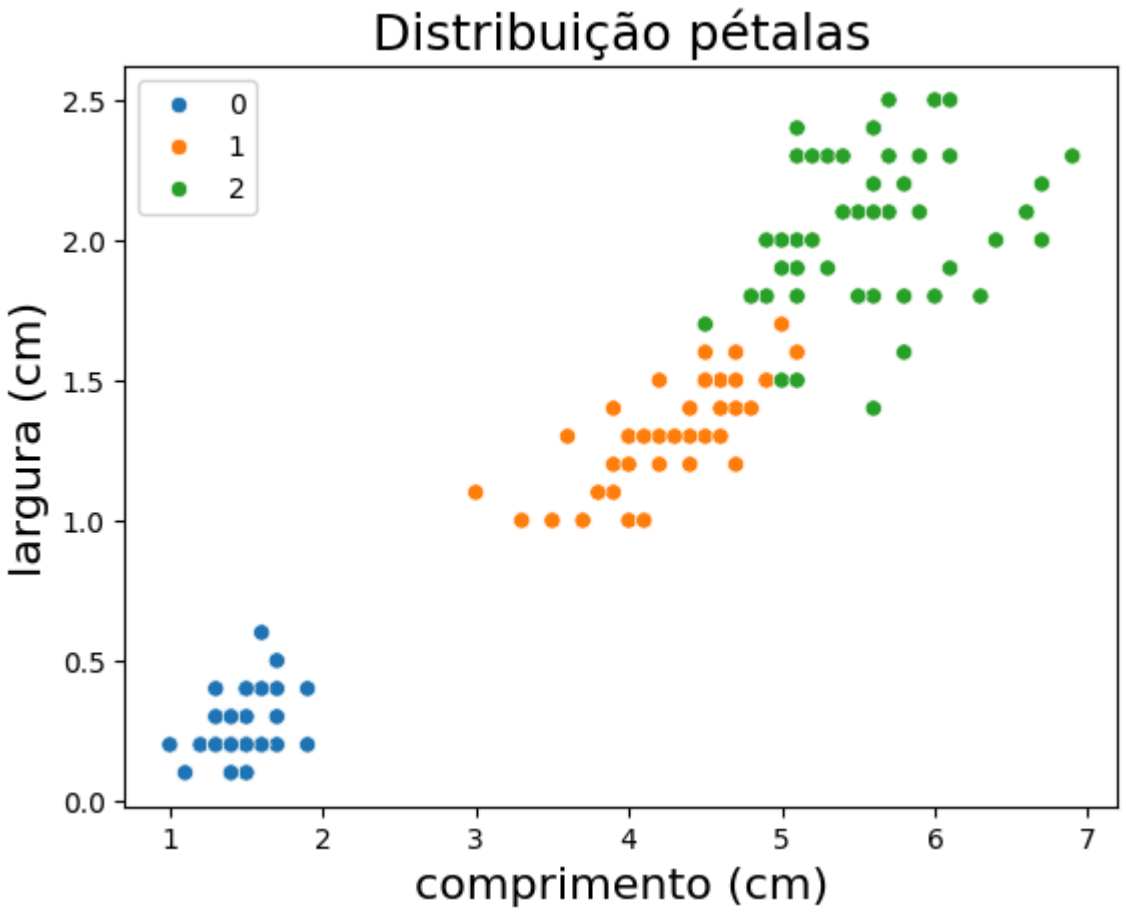

Figura 1: Modelo gráfico da distribuição das pétalas

O gráfico de dispersão representando o comprimento versus a largura das pétalas revelou uma distinção clara entre as espécies, particularmente entre a *Iris setosa* e as demais. As flores da espécie *setosa* exibiram pétalas menores, com comprimento e largura concentrados na região inferior esquerda do gráfico, indicando valores mais baixos para ambas as dimensões. As espécies *Iris versicolor* e *Iris virginica* apresentaram pétalas maiores, com sobreposição considerável na distribuição, sugerindo maior similaridade entre essas duas espécies em relação às dimensões das pétalas.

3.1.2 Distribuição das sépalas

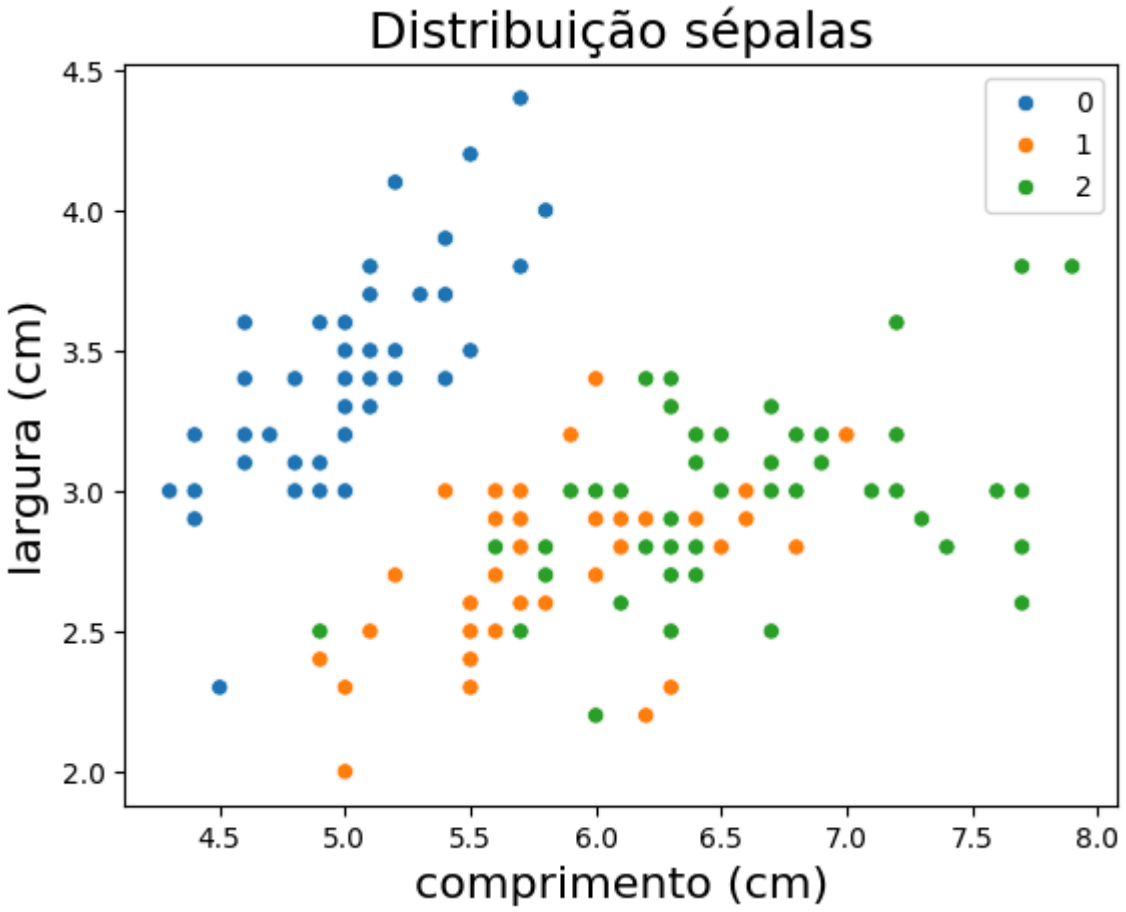

Figura 2: Modelo gráfico da distribuição das sépalas

A análise do gráfico de dispersão do comprimento versus a largura das sépalas demonstrou uma menor separação entre as espécies, especialmente entre *Iris versicolor* e *Iris virginica*. As flores da espécie *setosa* também apresentaram sépalas menores em comparação com as outras duas espécies. No entanto, a sobreposição entre as distribuições de *versicolor* e *virginica* foi mais pronunciada, indicando maior dificuldade na distinção dessas espécies com base apenas nas dimensões das sépalas.

3.1.3 Distribuição dos dados: Discussão e Conclusões

A análise exploratória dos dados indica que as características das pétalas são mais eficazes na discriminação das espécies de Iris do que as características das sépalas, evidenciado pela clara separação entre a espécie *setosa* e as demais no gráfico de dispersão das pétalas. A sobreposição observada entre as distribuições de *versicolor* e *virginica* no gráfico das sépalas sugere maior dificuldade na distinção dessas espécies, demandando, possivelmente, a consideração de outras características ou técnicas de classificação mais complexas. A correlação positiva entre as dimensões das pétalas e das sépalas pode ser utilizada na seleção de atributos para otimizar a performance do modelo de classificação. Em suma, as características das pétalas demonstraram maior potencial para a discriminação das espécies, enquanto as características das sépalas apresentaram menor poder de separação, informações cruciais para guiar as etapas subsequentes do projeto, como a seleção de atributos e a construção do modelo de classificação.

3.2 Modelo e treinamento: Visão geral

Após a etapa de tratamento dos dados, procedeu-se à construção e treinamento do modelo de classificação para a predição da espécie de flor Iris. Optou-se por utilizar um modelo de *Multilayer Perceptron* (MLP), uma rede neural artificial amplamente empregada em tarefas de classificação (HAYKIN, 2001).

3.2.1 Desempenho do Treinamento

O modelo Multilayer Perceptron (MLP), com a arquitetura e os hiperparâmetros descritos na seção de metodologia, foi treinado utilizando o conjunto de dados Iris. A Figura 3 apresenta os gráficos de perda e acurácia durante o processo de treinamento, para os conjuntos de treinamento e validação.

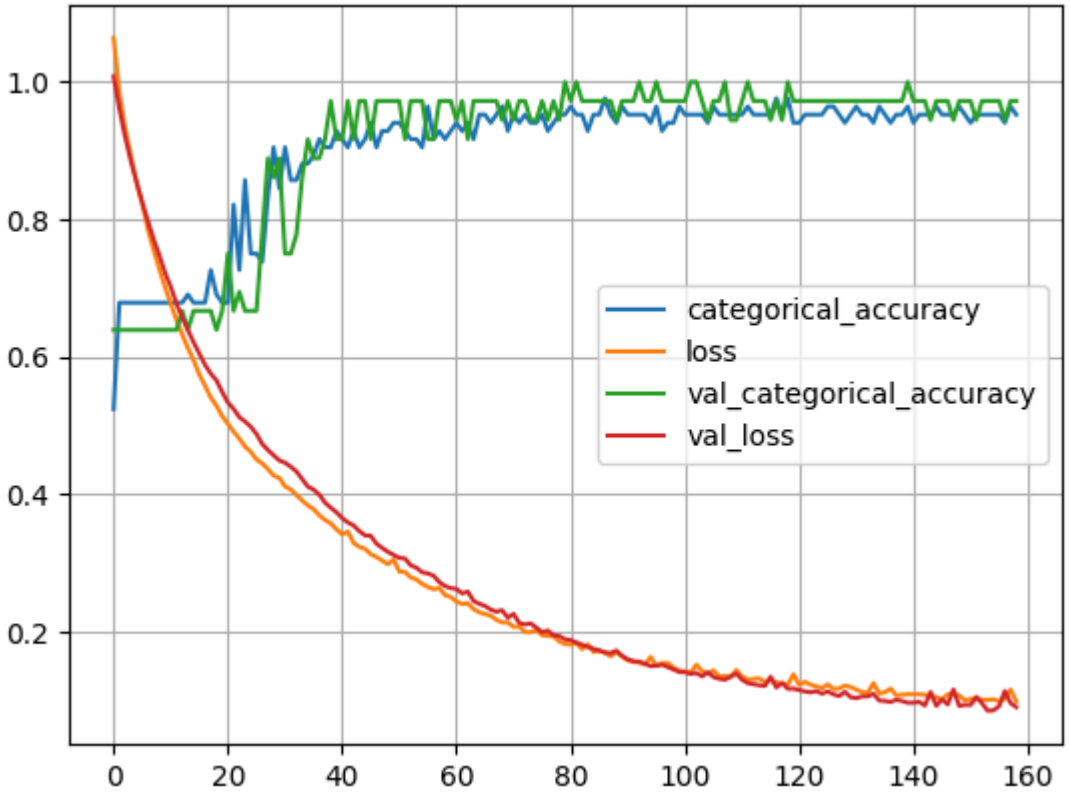

Figura 3: gráfico de perda e acurácia.

A perda no conjunto de teste também se manteve em um nível baixo, indicando que o modelo está fazendo previsões precisas.

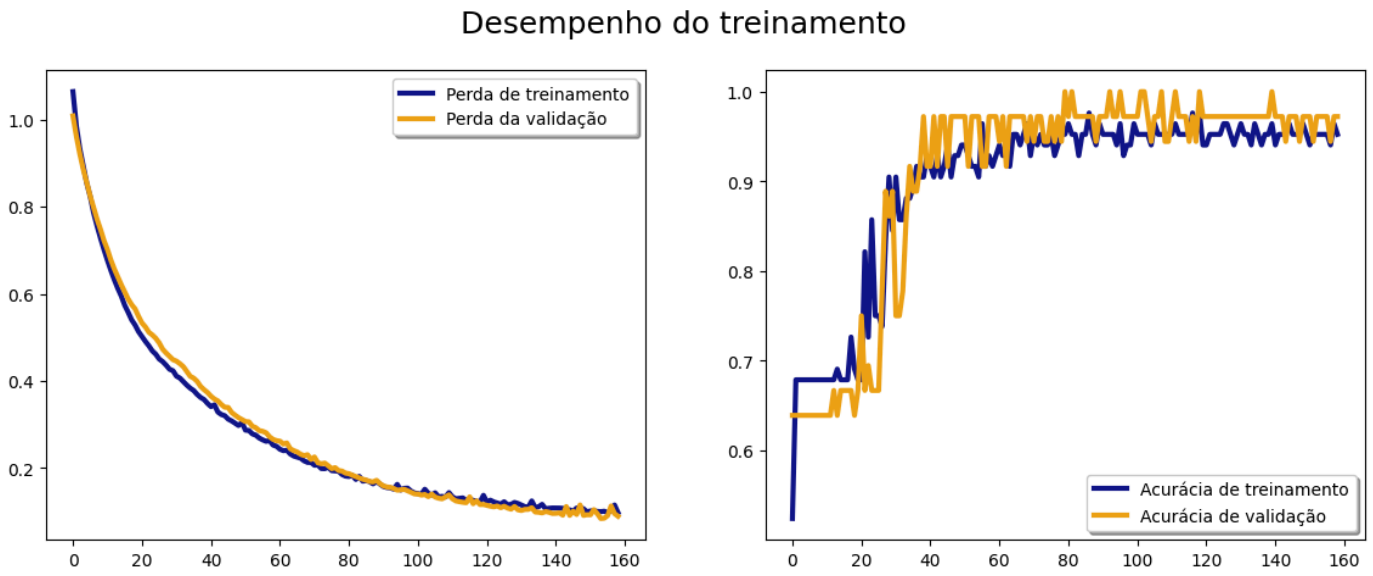

Figura 4: gráfico de validação

Se a acurácia no conjunto de treinamento for significativamente maior do que nos conjuntos de validação e teste, pode indicar *overfitting*, ou seja, o modelo está memorizando os dados de treinamento e não generalizando bem para novos dados (GOODFELLOW et al., 2016). Por outro lado, se a acurácia em todos os conjuntos for baixa, pode indicar *underfitting*, ou seja, o modelo não está aprendendo os padrões dos dados de forma adequada (RASCHKA; MIRJALILI, 2019).

No caso do modelo MLP para classificação de flores Iris, os resultados indicam um bom equilíbrio entre o desempenho nos três conjuntos, sugerindo que o modelo está generalizando bem e não sofrendo de *overfitting* ou *underfitting*.

3.2.2 Modelo e treinamento: Discussão e Conclusões

A análise dos gráficos de desempenho revela que o modelo MLP apresentou um bom aprendizado durante o treinamento, com a perda diminuindo e a acurácia aumentando ao longo das épocas, tanto para o conjunto de treinamento quanto para o conjunto de validação. A similaridade entre as curvas de treinamento e validação indica que o modelo não sofreu *overfitting*, demonstrando capacidade de generalização para dados não vistos durante o treinamento.

A utilização da técnica de *Early Stopping* contribuiu para evitar o *overfitting*, interrompendo o treinamento na época ideal, quando a perda na validação não apresentou mais melhora significativa. Essa técnica é amplamente recomendada para garantir que o modelo selecionado apresente o melhor equilíbrio entre desempenho no treinamento e capacidade de generalização (PREMA et al., 2021).

Os resultados do treinamento indicam que o modelo MLP é promissor para a classificação de flores Iris, com bom desempenho e capacidade de generalização.

3.3 Teste

A eficácia do modelo de classificação foi rigorosamente avaliada utilizando um conjunto de teste independente, correspondente a 20% do total de dados disponíveis. Nessa avaliação, o modelo demonstrou um desempenho notável, atingindo uma acurácia de 0,967 e uma perda (*loss*) de 0,103. Esses resultados evidenciam a capacidade do modelo em generalizar para dados inéditos, demonstrando alta precisão na classificação das diferentes espécies de íris, sem apresentar problemas de *overfitting* ou *underfitting*. A consistência desses resultados, quando comparados ao desempenho obtido durante as fases de treinamento e validação, reforça a robustez e a confiabilidade do modelo desenvolvido.

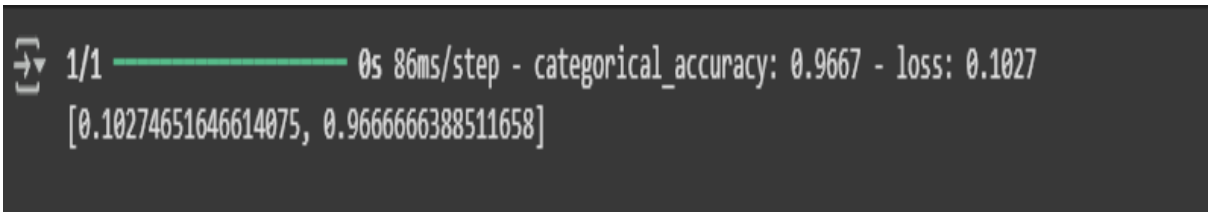

Figura 5: Teste de desempenho

Para avaliar a aplicação prática do modelo em um novo dado, o mesmo foi utilizado para classificar um conjunto de características não presentes no treinamento. Essas características, normalizadas, foram: comprimento da sépala de 0,61, largura da sépala de 0,50, comprimento da pétala de 0,69 e largura da pétala de 0,79. A análise do modelo resultou nas seguintes probabilidades para cada classe: *Setosa* com 0,00%, *Versicolor* com 13,27% e *Virginica* com 86,73%. Essas probabilidades sugerem que o modelo, com base nas características fornecidas, considera o novo dado mais provavelmente como pertencente à classe *Virginica*. Esse dado foi escolhido devido a suas características intermediárias, buscando simular uma entrada não totalmente similar a nenhuma classe do treinamento, e avaliar a capacidade do modelo de inferir novos dados.

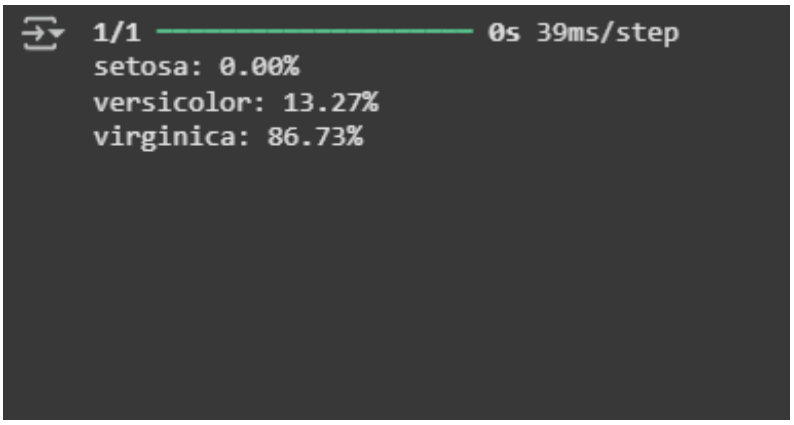

Figura 6: Saída do novo modelo

4 Conclusão

A análise conjunta das etapas de mensuração, tratamento de dados e desenvolvimento do modelo demonstra a eficácia das Redes Neurais Artificiais na classificação de flores Iris. Os resultados obtidos — evidenciados pelos gráficos de dispersão, curvas de perda e acurácia — mostram que a rede MLP é capaz de capturar os padrões essenciais do conjunto de dados e separar as espécies Setosa, Versicolor e Virginica com excelente desempenho. Em particular, notou-se uma distinção clara para a espécie Setosa, enquanto Versicolor e Virginica apresentaram características mais próximas, corroborando a sobreposição morfológica já apontada na literatura (LeCun, Bengio & Hinton, 2015).

O uso de técnicas de pré-processamento (normalização, codificação one-hot) e hiperparâmetros bem ajustados (como a quantidade de neurônios na camada oculta e o emprego de Early Stopping) contribuiu para evitar problemas de overfitting, garantindo que o modelo MLP generalizasse adequadamente para dados não vistos. Esse cuidado refletiu-se nas curvas de validação, que acompanham o treinamento de forma consistente, indicando um equilíbrio entre aprendizado e capacidade de generalização (Goodfellow et al., 2016).

A adoção da biblioteca Keras (Chollet, 2015) mostrou-se decisiva para a implementação ágil da rede, possibilitando experimentos iterativos com diferentes arquiteturas e otimizadores. Por meio da API de alto nível, foi possível ajustar rapidamente parâmetros como a função de ativação, o batch size e o otimizador, evidenciando quão determinante é a seleção criteriosa desses elementos para o sucesso de um modelo de classificação (Haykin, 1999).

Por fim, a abordagem proposta se revela promissora para problemas que envolvam a classificação de dados com múltiplas classes e características quantitativas. Ainda que o dataset Iris seja relativamente simples, ele cumpre o papel de demonstrar os conceitos centrais das RNAs, desde a inspiração biológica até o treinamento e avaliação em um ambiente prático. Trabalhos futuros podem incluir a aplicação de redes neurais mais profundas, testes em conjuntos de dados mais complexos e comparação com outros algoritmos de classificação, como Random Forest e Support Vector Machines (Raschka & Mirjalili, 2019). Dessa forma, amplia-se o conhecimento acerca dos limites e oportunidades oferecidos pelas Redes Neurais Artificiais em cenários reais de análise e predição de dados.

5 Referências

SEABORN. Seaborn: statistical data visualization. Disponível em: <https://seaborn.pydata.org/>. Acesso em: [18/02/25].

CHOLET, F. Keras: The Python Deep Learning library. 2015. Disponível em: https://keras.io.

Acesso em: [18/02/25].

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Disponível em: https://jmlr.org/papers/v12/pedregosa11a.html. Acesso em: [18/02/25].

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

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. 3. ed. Birmingham: Packt Publishing, 2019.

HAYKIN, S. **Redes neurais: princípios e prática**. 2. ed. Porto Alegre: Bookman, 2001.

CHOLLET, F. *Deep Learning with Python.* 1. ed. Shelter Island: Manning Publications Co., 2018.

DUA, D.; GRAFF, C. UCI Machine Learning Repository. 2019. Disponível em: http://archive.ics.uci.edu/ml. Acesso em: 20 fev. 2025.

FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, v. 7, n. 2, p. 179-188, 1936.

HAYKIN, S. *Neural networks: a comprehensive foundation.* 2. ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, v. 521, n. 7553, p. 436-444, 2015.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, v. 5, n. 4, p. 115-133, 1943.

NIELSEN, M. *Neural Networks and Deep Learning.* Determination Press, 2015.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533-536, 1986.

SLIDES-11. *Redes neurais p1: Inspiração biológica e aplicações.* Material de aula, 2024.

SLIDES-12. *Redes neurais p2: Conceitos.* Material de aula, 2024.