Projetos de *Machine Learning* e *Deep Learning*:
Previsão da Severidade do Câncer
e
Classificação de Dígitos Manuscritos (MNIST)

Discente: Fernando José Zardinello Batistti
Docente: Prof. Dr. Joylan Maciel Nunes

UNIVERSIDADE FEDERAL DA INTEGRAÇÃO LATINO-AMERICANA
MESTRADO PROFISSIONAL EM FÍSICA APLICADA

23 de junho de 2025

UNILA

Sumário

UNILA

1. Coleta de Dados

1.1 Coleta de Dados: Projeto I: Severidade do Câncer

Fonte e Método

O *dataset* foi carregado a partir de um arquivo CSV ("global_cancer_patients_2015_2024.csv") armazenado no *Google Drive* e obtido na plataforma *Kaggle*.

- **Código:** A biblioteca `pandas` foi utilizada para a leitura com a função `pd.read_csv()`.
- **Justificativa:** É o método padrão e mais eficiente para manipular dados tabulares em *Python*.

Descrição dos Dados

O conjunto de dados é sintético e simula tendências globais de pacientes com câncer, contendo características demográficas, fatores de risco e informações clínicas.

UNILA

Fonte e Método

Utilizou-se o *dataset* MNIST-120k, um arquivo no formato "*pickle*", também carregado a partir do *Google Drive* e obtido na plataforma *Kaggle*.

- **Código:** Foi necessária uma função auxiliar ("*unpickle*") para desserializar e ler o objeto *Python* do arquivo.
- **Justificativa:** O formato "*pickle*"é comum para armazenar objetos complexos como *datasets* de imagens.

UNILA

Descrição dos Dados

O conjunto contém 120.000 imagens de 28x28 *pixels* de dígitos manuscritos (0 a 9) e seus respectivos rótulos.

Figura 1: Amostra dos 5 primeiros dígitos do *dataset*.

Fonte: Saída do Código..

2. Limpeza e Pré-processamento

2.1 Limpeza: Projeto I: Severidade do Câncer

Ações Realizadas

- **Verificação de Nulos:** A função `df.info()` foi usada para inspecionar os tipos de dados e a presença de valores nulos. Nenhum valor nulo foi encontrado.
- **Remoção de Coluna:** A coluna `Patient_ID` foi removida (`df.drop()`), pois é um identificador único sem valor preditivo para o modelo.

Justificativa

Remover *features* irrelevantes como IDs ajuda a simplificar o modelo e a evitar que ele aprenda padrões espúrios.

UNILA

Ações Realizadas

- **Normalização:** Os valores dos *pixels*, originalmente de 0 a 255, foram divididos por 255 e isso coloca todos os dados na escala de [0, 1].
- **Justificativa:** A normalização melhora a estabilidade e acelera a convergência do treinamento de redes neurais.
- **Remodelação (*Reshape*):** Os dados foram remodelados para o formato (96000, 28, 28, 1).
- **Justificativa:** Este é o formato de entrada esperado pelas camadas convolucionais do *Keras*/*TensorFlow*: (amostras, altura, largura, canais de cor).

UNILA

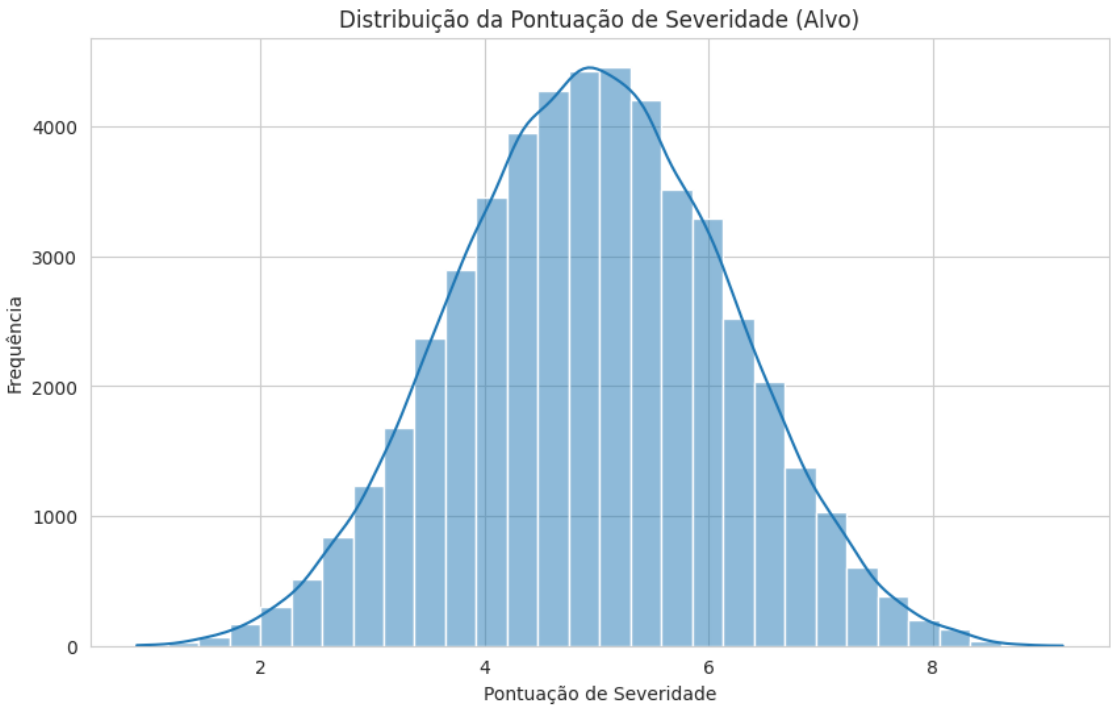
3. Análise Exploratória (EDA)

UNILA

Distribuição do Alvo

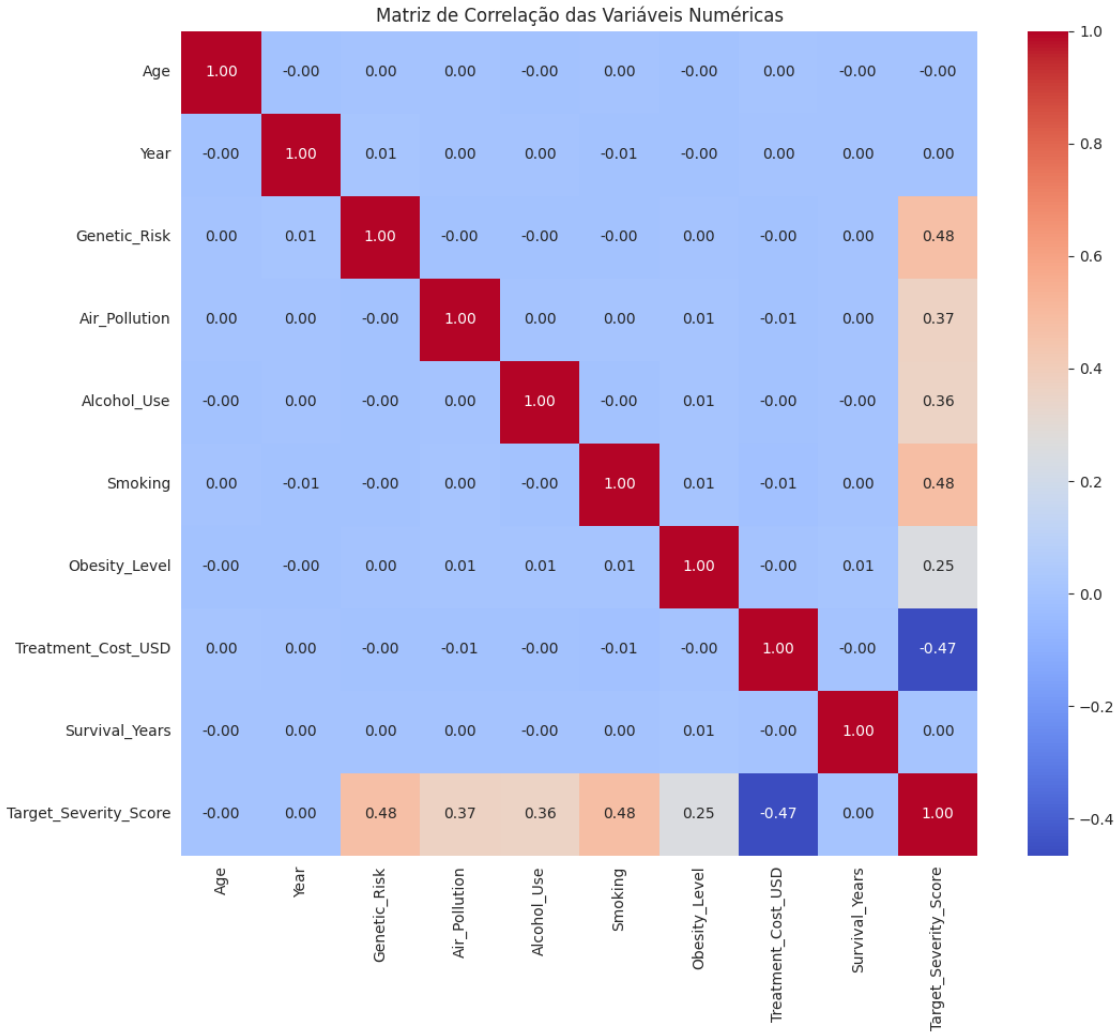
O histograma da variável alvo (`Target_Severity_Score`) mostrou uma distribuição razoavelmente normal. Isso indica que não há um grande desbalanceamento nos níveis de severidade, o que é favorável para o treinamento de um modelo de regressão.

Figura 2: Distribuição Alvo.



Fonte: Saída do Código.

Figura 3: Matriz de Correlação.

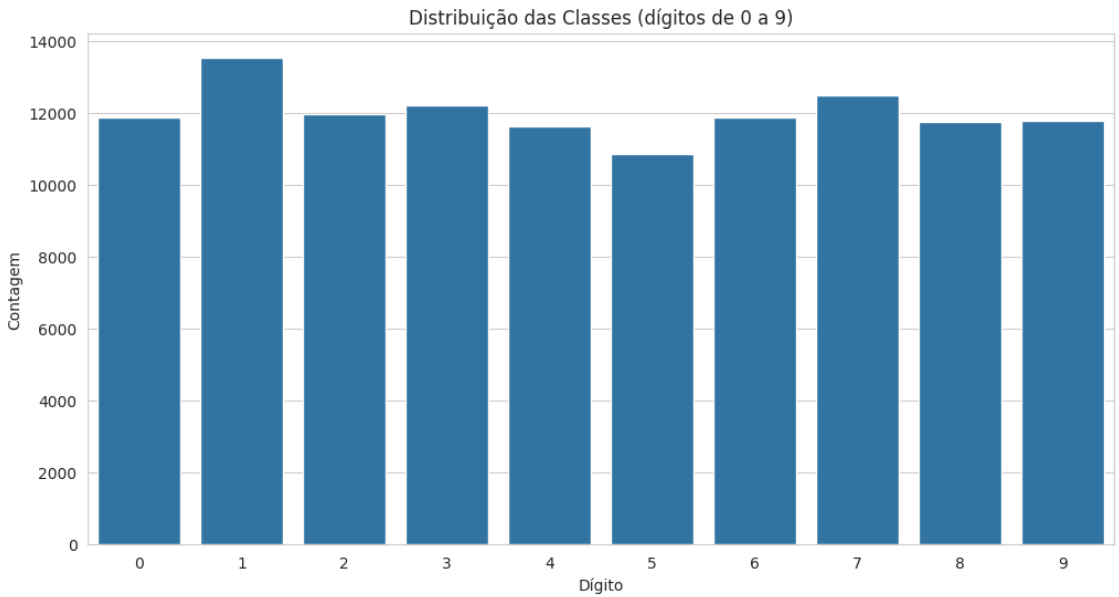


Fonte: Saída do Código.

Análise

O gráfico de barras mostrou que o *dataset* é bem balanceado, com cada dígito (0 a 9) tendo aproximadamente o mesmo número de amostras.
Justificativa: Isso é fundamental para garantir que o modelo não desenvolva um viés para as classes mais frequentes e que a métrica de acurácia seja confiável.

Figura 4: Distribuição das Classes.



Fonte: Saída do Código.

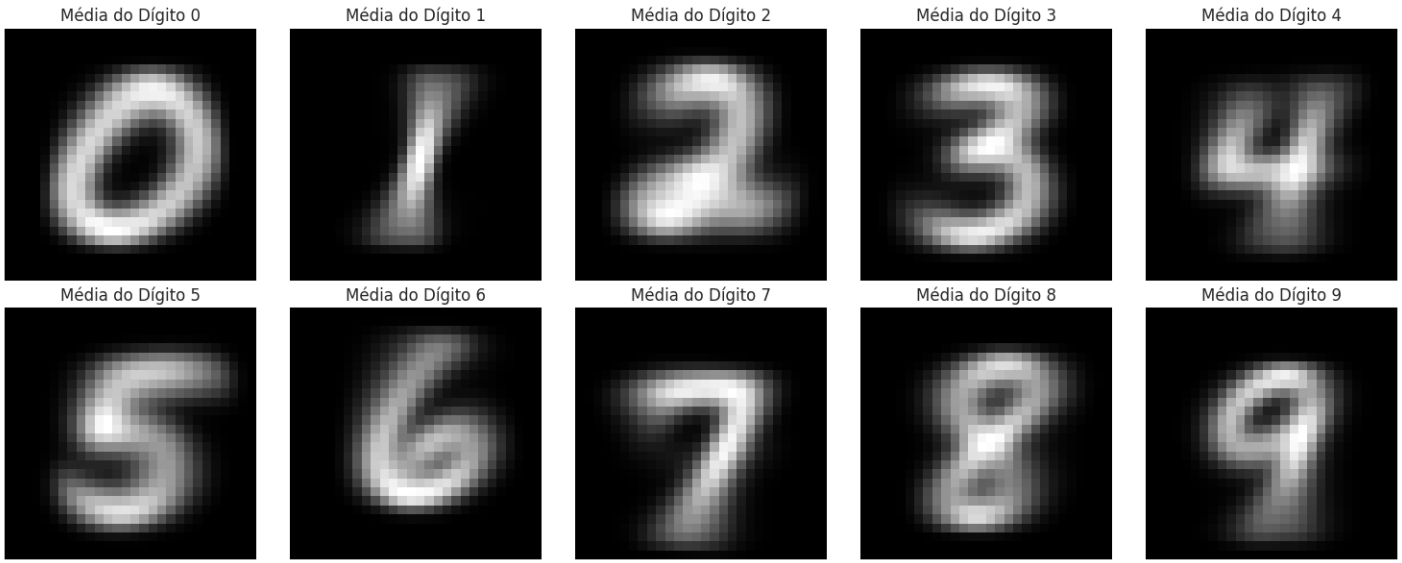
3.2 EDA: Projeto II: Imagem Média por Classe

Análise

A visualização da imagem média de cada dígito ajuda a identificar a aparência "prototípica"da classe.

Justificativa: Permite notar possíveis sobreposições visuais que podem confundir o modelo, como entre os dígitos "4"e "9".

Figura 5: Média das Imagens por Classe.



Fonte: Saída do Código.

4. Engenharia de Características

4.1 Justificativa das Abordagens

Projeto I: Severidade do Câncer:

Para este modelo clássico, a engenharia de características consistiu em preparar os dados para a transformação:

- **Identificação:** Separação das *features* em numéricas e categóricas.
- **Transformação:** As *features* numéricas foram padronizadas (`StandardScaler`) e as categóricas foram convertidas com `OneHotEncoder` dentro de um *pipeline*.

Projeto II: MNIST - Abordagem Automática

Uma grande vantagem das CNNs é que elas realizam a engenharia de características automaticamente.

- As camadas convolucionais aprendem a extrair *features* relevantes (bordas, texturas, formas) diretamente dos *pixels*, eliminando a necessidade de criação manual de características.

UNILA

5. Divisão do *Dataset*

5.1 Divisão Treino-Teste

Metodologia

Para ambos os projetos, foi utilizada a função `train_test_split` do *Scikit-learn* com uma divisão de **80% para treino** e **20% para teste**, usando `random_state=42` para reprodutibilidade.

Projeto I: Severidade do Câncer

Divisão padrão, pois a variável alvo (regressão) é contínua e não necessita de estratificação.

Projeto II: MNIST

A divisão foi **estratificada** (`stratify=y`).

- **Justificativa:** Garante que a proporção de cada dígito seja a mesma nos conjuntos de treino e teste, o que é necessário para problemas de classificação.

6. Escolha e Arquitetura do Modelo

6.1.1 Justificativa da Escolha: Projeto I

Justificativa da Escolha do *RandomForestRegressor*

Para o problema de regressão com dados tabulares, o ***RandomForestRegressor*** foi selecionado por suas características principais:

- **Robustez:** Como um modelo de *ensemble*, ele combina os resultados de múltiplas árvores de decisão. Isso torna o modelo final mais estável e menos propenso a *overfitting* em comparação com uma única árvore.
- **Versatilidade:** Lida nativamente com dados tabulares que contêm uma mistura de *features* numéricas e categóricas, como é o caso deste projeto, simplificando a etapa de pré-processamento.

UNILA

6.1.2 Hiperparâmetros Padrão (*Baseline*)

Configurações Iniciais do Modelo

O modelo foi treinado com os seguintes parâmetros padrão da biblioteca *Scikit-learn*, que serviram como nossa linha de base (*baseline*) antes da otimização:

- `n_estimators:` `100` (Número de árvores na floresta).
- `max_depth:` `None` (As árvores crescem até a pureza máxima, o que pode levar a *overfitting*).
- `min_samples_split:` `2` (Número mínimo de amostras para que um nó possa ser dividido).
- `min_samples_leaf:` `1` (Número mínimo de amostras que uma folha final deve ter).

6.2 Arquitetura do *Pipeline*: Projeto I

Interpretabilidade e Estrutura

Interpretabilidade: Uma vantagem adicional do *Random Forest* é a capacidade de analisar a importância das *features*, ajudando a entender quais fatores mais influenciam a severidade.

Estrutura do *Pipeline* Completo

A solução final não é apenas o *regressor*, mas um *pipeline* que combina:

1. **Pré-processador:** Com `StandardScaler` para as *features* numéricas e `OneHotEncoder` para as categóricas.
2. **Estimador:** O próprio `RandomForestRegressor`.

Justificativa: Garante consistência e previne o vazamento de dados.

6.3 Escolha do Modelo: Projeto II (CNN)

Justificativa da Escolha

Para a tarefa de classificação de imagens do MNIST, uma **Rede Neural Convolucional (CNN)** foi a escolha natural.

- **Estado da Arte:** É a arquitetura padrão-ouro para tarefas de visão computacional, pois é especialmente projetada para capturar padrões espaciais hierárquicos (linhas, formas, etc.).
- **Extração Automática de *Features*:** As camadas convolucionais eliminam a necessidade de engenharia de características manual, aprendendo os melhores filtros diretamente dos dados.

6.4 Arquitetura do Modelo: Projeto II (CNN)

Arquitetura Base Utilizada

O modelo foi construído sequencialmente com as seguintes camadas:

1. **Conv2D:** 32 filtros, kernel size 3x3, ativação ReLU.
2. **MaxPooling2D:** Janela 2x2 para downsampling.
3. **Conv2D:** 64 filtros, kernel size 3x3, ativação ReLU.
4. **MaxPooling2D:** Janela 2x2.
5. **Flatten:** Transforma os mapas 2D em um vetor 1D.
6. **Dense:** 100 neurônios, ativação ReLU.
7. **Dropout (0.5):** Camada para regularização e prevenção de *overfitting*.
8. **Dense (Saída):** 10 neurônios (um por classe), ativação **Softmax** para gerar as probabilidades de classificação.

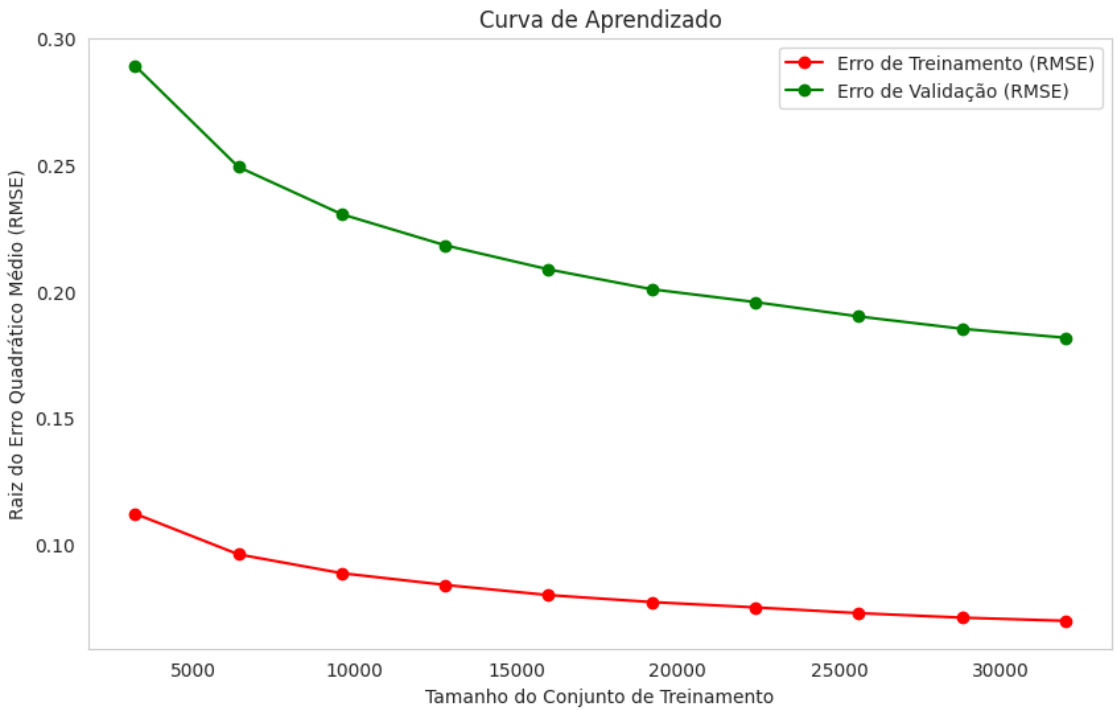
UNILA

7. Treinamento e Validação

Curva de Aprendizado

A validação da robustez do modelo base foi feita analisando sua curva de aprendizado, gerada com validação cruzada de 5 *folds*.

Figura 6: Curva de Aprendizado do Modelo Base.



Fonte: Saída do Código.

Validação Cruzada Estratificada (3-*Folds*)

Para obter uma estimativa mais confiável da performance, foi usada validação cruzada no conjunto de treino.

- **Método:** O conjunto de treino foi dividido em 3 "*folds*". O modelo foi treinado 3 vezes, cada vez usando 2 *folds* para treino e 1 para validação.
- **Resultado:** A acurácia média de validação foi de **99,19%** com um desvio padrão $\pm 0,0008$, demonstrando que o modelo é estável e performa bem em diferentes subconjuntos de dados.

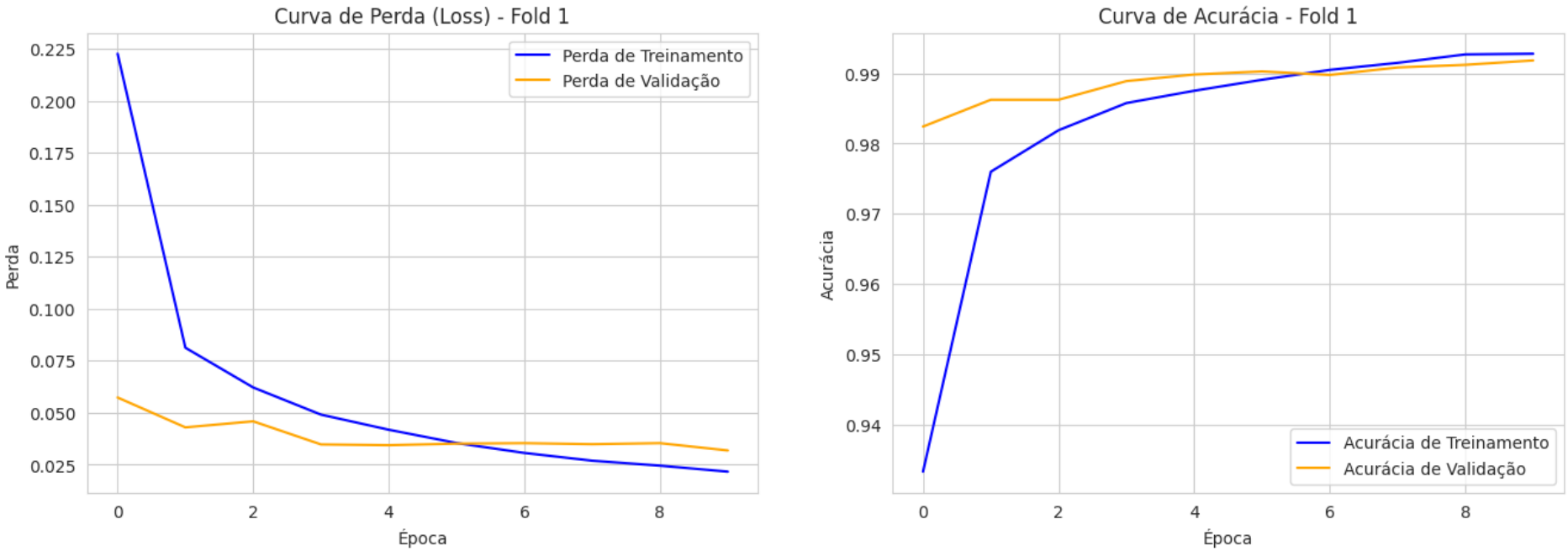
7.3 Análise das Curvas de Validação

Análise de Consistência do Treinamento

A consistência do treinamento do modelo foi verificada visualmente através das curvas de aprendizado de cada um dos 3 *folds* da validação cruzada. Como será visto nos próximos slides, todas as curvas apresentam um comportamento similar, com a acurácia de treinamento e validação convergindo para valores altos e próximos, indicando um modelo estável.

7.3 Curvas de Acurácia por *Fold* (*Fold* 1)

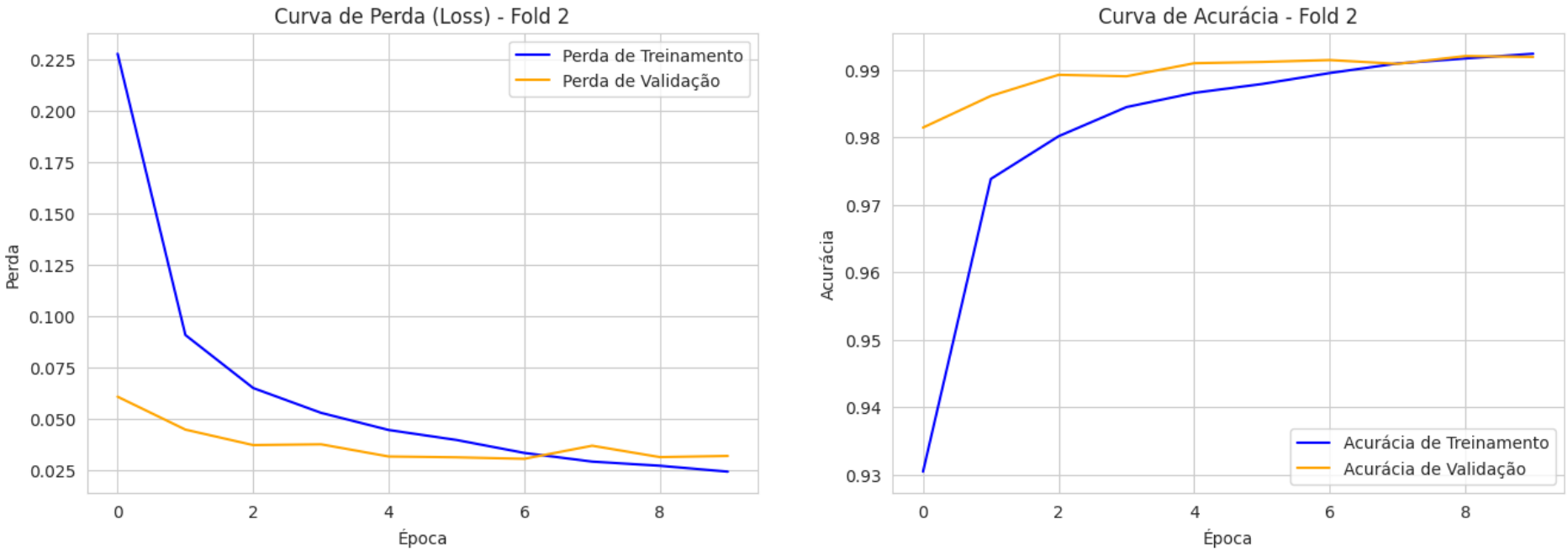
Figura 7: Curva de Acurácia e Perda - *Fold* 1



Fonte: Saída do Código.

7.3 Curvas de Acurácia por *Fold* (*Fold* 2)

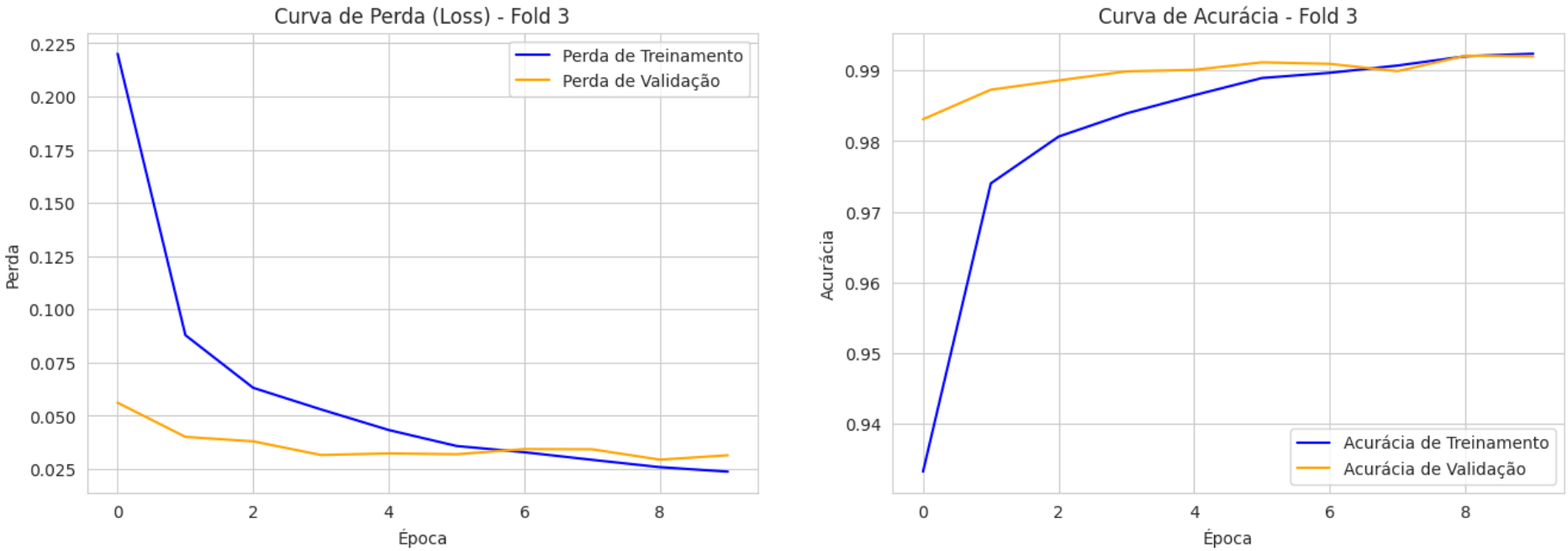
Figura 8: Curva de Acurácia e Perda - *Fold* 2



Fonte: Saída do Código.

7.3 Curvas de Acurácia por *Fold* (*Fold* 3)

Figura 9: Curva de Acurácia e Perda - *Fold* 3



Fonte: Saída do Código.

8. Avaliação do Modelo Final

8.1 Avaliação: Métricas de Desempenho (Projeto I)

Métricas de Regressão no Conjunto de Teste

O modelo base (não otimizado) foi avaliado no conjunto de teste, obtendo os seguintes resultados:

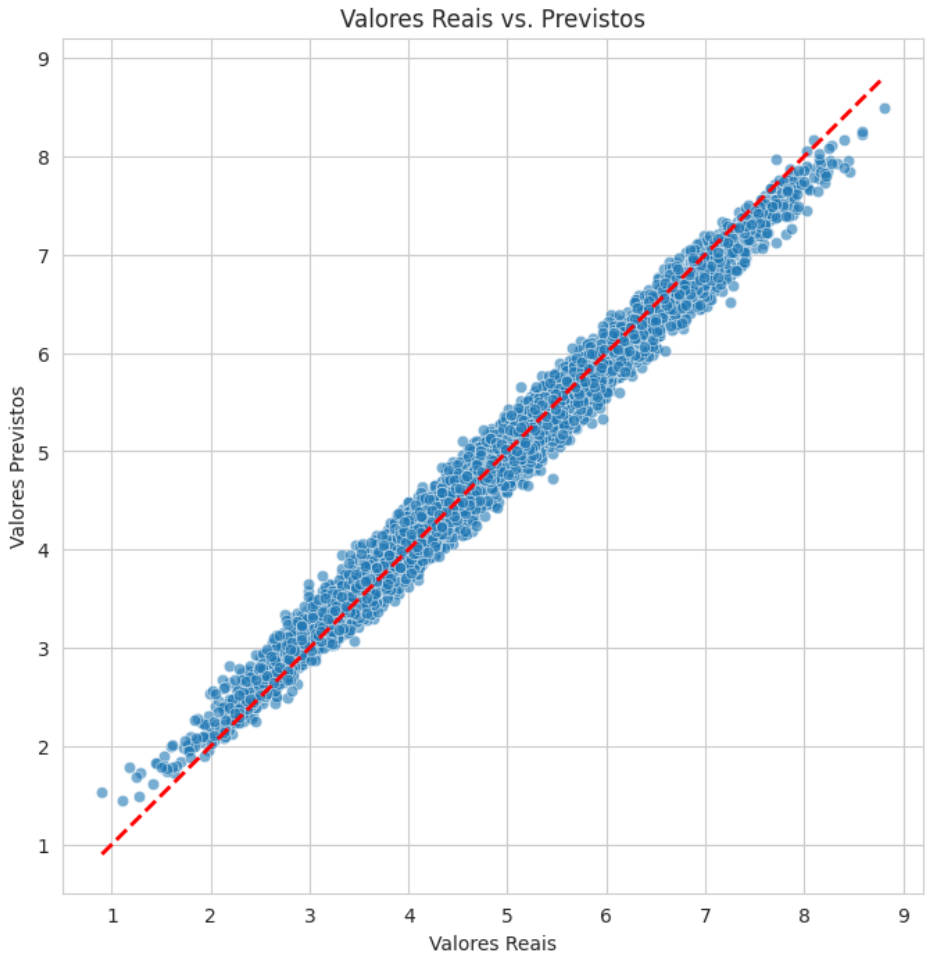
- *Mean Absolute Error* (MAE): 0,136
- *Root Mean Squared Error* (RMSE): 0,172
- ***R-squared* ($\mathbf{R^2}$): 0,979**

Justificativa

Um R^2 de 0,979 é um resultado excelente. Isso indica que o modelo consegue explicar **97,9%** da variância na pontuação de severidade com base nas *features* fornecidas.

8.2 Análise Visual: Reais vs. Previstos (Projeto I)

Figura 10: Dispersão: Valores Reais vs. Previstos.



Fonte: Saída do Código.

8.3 Avaliação: Métricas de Desempenho (Projeto II)

Resultados no Conjunto de Teste

O modelo final (não otimizado) foi treinado com todos os dados de treino e avaliado no conjunto de teste, alcançando:

- **Acurácia Geral: 99,20%**

Análise do Relatório de Classificação

O *classification report* demonstrou que as métricas de:

- ***Precision*, *Recall* e *F1-Score*** foram consistentemente altas (próximas de 0,99) para todas as 10 classes de dígitos.

Justificativa: Isso indica que o modelo não apenas é preciso no geral, mas também é equilibrado e não possui um viés significativo para nenhuma classe específica.

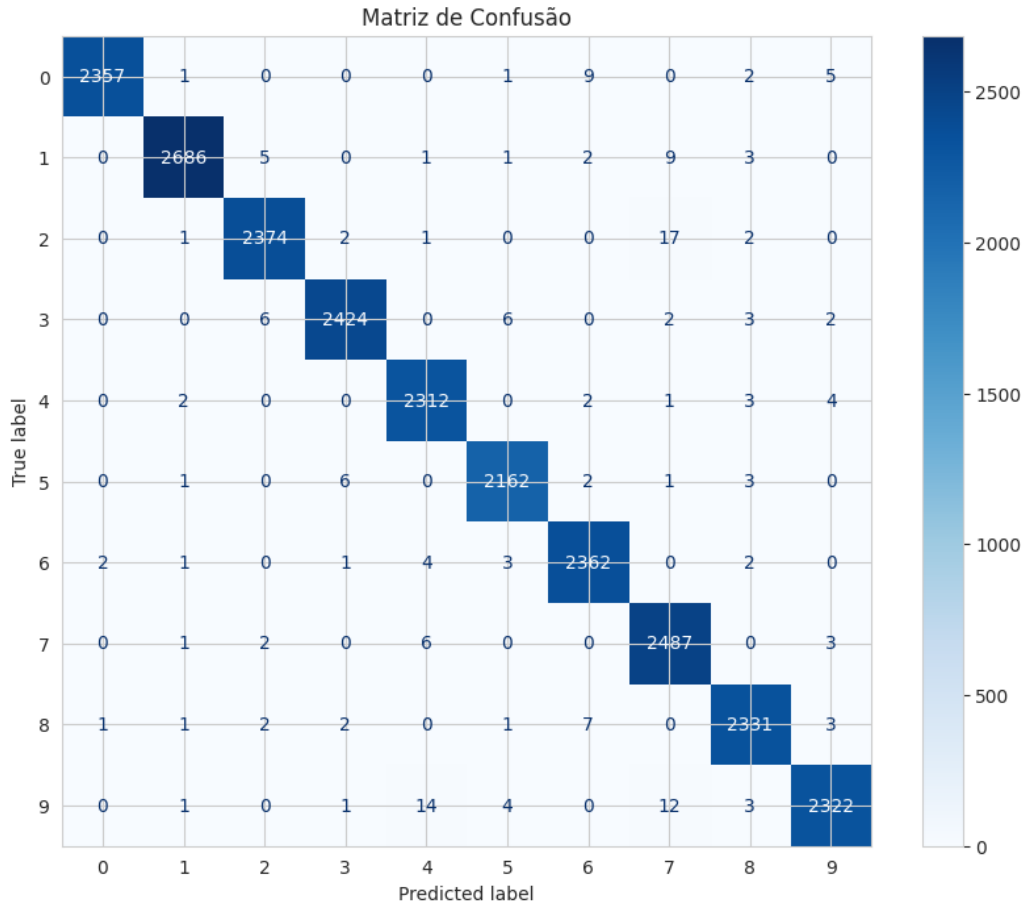
UNILA

8.4 Análise Visual: Matriz de Confusão (Projeto II)

Análise da Matriz

A matriz de confusão visualiza o desempenho detalhado do modelo. A diagonal principal mostra o número de previsões corretas para cada classe.
Observação: Os valores fora da diagonal são muito baixos, indicando pouquíssimos erros de classificação e confirmando a alta performance do modelo.

Figura 11: Matriz de Confusão no Conjunto de Teste.



Fonte: Saída do Código.

9. Otimização de Hiperparâmetros

9.1 Otimização: Projeto I: Severidade do Câncer

Método e Resultado

Foi utilizada a técnica ***RandomizedSearchCV*** para testar 20 combinações aleatórias de hiperparâmetros com validação cruzada de 5 *folds*.

- **Melhores Parâmetros:** n_estimators: 300, min_samples_split: 2, min_samples_leaf: 1, max_depth: None.

Análise

O modelo otimizado teve um desempenho quase idêntico ao modelo base (R^2 permaneceu em 0,979).

- **Justificativa:** Isso indica que os hiperparâmetros padrão do *RandomForestRegressor* já eram muito eficazes para este *dataset*, deixando pouca margem para melhorias.

UNILA

9.2 Otimização: Projeto II: MNIST

Método e Resultado

Foi utilizado o **KerasTuner com RandomSearch** para testar 5 combinações de hiperparâmetros da CNN em um subconjunto dos dados.

- **Melhores Parâmetros Encontrados:** taxa de aprendizado de 0,001, 48 filtros na primeira camada e 100 neurônios na camada densa.

Análise

O modelo otimizado alcançou uma acurácia de **98,83%** no conjunto de teste.

- **Observação:** Embora excelente, o valor foi ligeiramente inferior à acurácia do modelo base treinado no *dataset* completo (99,20%). Isso pode ocorrer porque a busca foi feita em um subconjunto de dados para economizar tempo.

10. Implantação e Monitoramento

10.1 Implantação: Etapa Final do *Pipeline*

Objetivo da Implantação

Após o treinamento e a validação, o passo final do ciclo de vida de um modelo de *Machine Learning* é a sua implantação.

O objetivo é salvar o artefato treinado (o modelo ou o *pipeline* completo) de forma que ele possa ser carregado e utilizado em uma aplicação real, como uma API web ou um sistema de diagnóstico, para fazer previsões em novos dados sem a necessidade de retreinamento a cada uso.

10.2 Implantação: Métodos de Salvamento

Projeto I: Severidade do Câncer

- **Método:** O *pipeline* completo (pré-processador + modelo) foi salvo usando a função `joblib.dump()`.
- **Arquivo:** `modelo_..._v1.joblib`.
- **Justificativa:** `joblib` é eficiente para salvar objetos *Python* que contêm grandes arrays NumPy, como os modelos do *Scikit-learn*.

Projeto II: MNIST

- **Método:** O modelo Keras/TensorFlow foi salvo usando a função nativa `model.save()`.
- **Arquivo:** `..._otimizado.keras`.
- **Justificativa:** Este método salva não apenas os pesos, mas toda a arquitetura do modelo e a configuração do otimizador.

UNILA

10.3 Próximos Passos: Monitoramento

Monitoramento Contínuo

A implantação não é o fim do processo. O desempenho do modelo em produção deve ser monitorado continuamente para detectar a **degradação de performance** (também conhecida como *model drift*).

O que é o "Drift"?

- Ocorre quando os novos dados do mundo real começam a ter características estatísticas diferentes dos dados com os quais o modelo foi treinado.
- **Ação:** A detecção de um *drift* significativo é o gatilho que indica a necessidade de um retreinamento do modelo com dados mais recentes para manter sua acurácia e relevância.

UNILA

11. Referências

Referências I


Breiman, L. (2001).
Random forests.
Machine Learning, 45(1):5–32.


Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020).
An image is worth 16x16 words: Transformers for image recognition at scale.
arXiv preprint arXiv:2010.11929.


Friedman, J. H. (2001).
Greedy function approximation: A gradient boosting machine.
Annals of Statistics, 29(5):1189–1232.

Referências II


Hastie, T., Tibshirani, R., and Friedman, J. (2009).
The Elements of Statistical Learning: Data Mining, Inference, and Prediction.
Springer, New York, 2 edition.


He, K., Zhang, X., Ren, S., and Sun, J. (2016).
Deep residual learning for image recognition.
In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.


Ioffe, S. and Szegedy, C. (2015).
Batch normalization: Accelerating deep network training by reducing internal covariate shift.
arXiv preprint arXiv:1502.03167.

UNILA

Referências III

Kaggle Community (2024).
Mnist-120k handwritten digits dataset.
Disponível em plataformas de datasets como o Kaggle. Acesso em: 14 jun. 2025.

Kingma, D. P. and Ba, J. (2014).
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012).
Imagenet classification with deep convolutional neural networks.
In *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.

UNILA

Referências IV


Mughal, Z. (2025).
Global cancer patients (2015-2024).
https://www.kaggle.com/datasets/zahidmughal2343/global-cancer-patients-2015-2024.
Acesso em: 13 jun. 2025.


O'Malley, T. et al. (2019).
Kerastuner.
https://github.com/keras-team/keras-tuner.


Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011).
Scikit-learn: Machine learning in python.
Journal of Machine Learning Research, 12:2825–2830.

Referências V


Sabour, S., Frosst, N., and Hinton, G. E. (2017).
Dynamic routing between capsules.
In *Advances in Neural Information Processing Systems 30*, pages 3856–3866.


Simonyan, K. and Zisserman, A. (2014).
Very deep convolutional networks for large-scale image recognition.
arXiv preprint arXiv:1409.1556.


Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014).
Dropout: A simple way to prevent neural networks from overfitting.

Journal of Machine Learning Research, 15:1929–1958.

UNILA

Referências VI

Sung, H., Ferlay, J., Siegel, R. L., Laversanne, M., Soerjomataram, I., Jemal, A., and Bray, F. (2024).
Global cancer statistics 2022: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries.
CA: A Cancer Journal for Clinicians, 74(3):209–249.

Yadav, C. and Bottou, L. (2019).
Cold case: The lost mnist digits.
In *Advances in Neural Information Processing Systems*, volume 32.

Obrigado!

fjz.batistti.2017@aluno.unila.edu.br