QuickML Creator: Uma ferramenta para integração de redes neurais em aplicações web

Thanize Assunção Rodrigues [1] **, Daniel Welfer**[2]

[1,2]Universidade Federal de Santa Maria (UFSM)
Santa Maria – RS – Brasil

thanizerodrigues@hotmail.com, welfer@gmail.com

Abstract. *This article presents QuickML Creator, a tool that automates the creation of web applications integrated with pre-trained machine learning models. The tool was evaluated based on the ISO/IEC 25010 standard, showing positive results in terms of usability, efficiency, and reliability for users with varying levels of experience.*

Resumo. *Este artigo apresenta a QuickML Creator, ferramenta que automatiza a criação de aplicações web integradas a modelos de aprendizado de máquina pré-treinados. A ferramenta foi avaliada com base na norma ISO/IEC 25010:2023, apresentando resultados positivos quanto usabilidade, eficiência e confiabilidade para usuários com diferentes níveis de experiência.*

1. Introdução

A grande quantidade de volumes de dados em áreas como saúde, agricultura e educação promoveu o uso de aprendizado de máquina para tratar de problemas complexos [Oliveira 2023]. Apesar dos avanços, desafios permanecem no *deployment*, devido à interoperabilidade entre plataformas e à falta de conhecimento técnico de usuários finais [Braga 2023, Baier et al. 2019].

Frameworks como Streamlit, Dash e Gradio surgiram para simplificar o desenvolvimento de aplicações *web* integradas a modelos de aprendizado de máquina, oferecendo componentes e abstrações que facilitam a criação de interfaces interativas [Wirfs-Brock et al. 1990]. Entretanto, ainda apresentam limitações de flexibilidade, curva de aprendizado e portabilidade.

Para superar algumas limitações, as plataformas *low-code* têm se mostrado importantes para ampliar o acesso ao aprendizado de máquina, permitindo que usuários com diferentes níveis de experiência desenvolvam aplicações funcionais [de Freitas et al. 2025, Sundberg and Holmström 2023]. Estudos também destacam a necessidade de personalização e evolução contínua dessas ferramentas [dos Santos de Oliveira 2025], contexto que fundamenta o desenvolvimento do QuickML Creator.

Diante de tantas opções, torna-se necessário avaliar qual delas atende melhor às demandas de um projeto. Apresenta-se, portanto, a QuickML Creator, ferramenta que automatiza a criação de aplicações integradas a modelos de aprendizado de máquina, oferecendo solução acessível para usuários com diferentes níveis de conhecimento. A análise comparativa e a validação da ferramenta adotaram a ISO/IEC 25010:2023 [International Organization for Standardization 2023], considerando atributos como funcionalidade, usabilidade e eficiência.

2. Método

A metodologia deste estudo foi organizada em duas etapas principais. Na primeira, realizou-se uma análise comparativa de frameworks para *deployment* de modelos de aprendizado de máquina, utilizando modelos pré-treinados voltados à detecção de múltiplas doenças disponíveis em repositórios como Kaggle e GitHub. Foram selecionados três modelos de alta acurácia, representando diferentes técnicas de aprendizado.

Na segunda etapa, definiram-se os frameworks Streamlit, Gradio e Dash, considerando critérios como facilidade de uso, versatilidade, popularidade e integração com modelos treinados. As aplicações foram estruturadas em páginas independentes, permitindo upload de imagens ou entrada de dados, com diagnósticos exibidos diretamente abaixo das informações fornecidas. A comparação seguiu os critérios da ISO/IEC 25010:2023, incluindo funcionalidade, confiabilidade, usabilidade, manutenibilidade, portabilidade, segurança e compatibilidade, com padronização do número de páginas e modelos integrados para garantir justiça na avaliação.

A partir dessa análise, desenvolveu-se o QuickML Creator, com o objetivo de facilitar a criação de aplicações *web* integradas a modelos pré-treinados, oferecendo um ambiente *low-code* tanto para desenvolvedores experientes quanto para iniciantes. Implementada em Python com Flask no *back-end* e HTML, CSS e JavaScript no *front-end*, a ferramenta foi hospedada no PythonAnywhere. Para assegurar transparência e reprodutibilidade, o código-fonte está disponível publicamente no GitHub: `https://github.com/ThanizeR/QuickMLCreator`.

O desenvolvimento seguiu o modelo sequencial clássico do ciclo de vida do software [Sommerville 2003], contemplando requisitos, projeto, implementação, integração, testes e manutenção. Entre as funcionalidades destacam-se autenticação de usuários, seleção de modelos (.h5 e .pkl) e geração automática de código. Foram disponibilizados templates em Streamlit e Gradio, alinhados à análise comparativa anterior. Um diagrama simplificado da arquitetura e exemplos de código gerado foram incluídos no repositório para apoiar a compreensão do fluxo de trabalho.

O QuickML Creator oferece um fluxo completo que inclui autenticação, geração de código e histórico de downloads, produzindo automaticamente arquivos essenciais para o *deployment*, como `app.py`, `README.md` e `requirements.txt`. Para validação, 15 usuários com diferentes níveis de experiência testaram a ferramenta. A avaliação, conduzida segundo a ISO/IEC 25010:2023, contemplou funcionalidade, usabilidade, eficiência, confiabilidade, satisfação e recomendação, enquanto aspectos como segurança, portabilidade e manutenibilidade ficaram fora do escopo.

3. Resultados

A comparação dos frameworks Streamlit, Gradio e Dash foi conduzida com base em dados de literatura e na norma ISO/IEC 25010:2023, assegurando objetividade e minimizando vieses. Foram avaliados atributos como funcionalidade, confiabilidade, usabilidade portabilidade critérios extras como *deployment* e comunidade entre outros. As interfaces desenvolvidas nos três frameworks mantiveram as mesmas funcionalidades, fluxos de navegação e integração com modelos pré-treinados, permitindo comparações justas.

Dentre os frameworks avaliados, o Gradio alcançou a maior pontuação geral (82),

destacando-se por sua interface intuitiva e facilidade no *deployment*. O Streamlit ficou em segundo lugar (79), oferecendo visualizações de qualidade, flexibilidade de layout e integração eficiente com modelos treinados. Já o Dash obteve a terceira posição (77), demonstrando integração robusta com modelos, mas apresentando limitações no *deployment*, principalmente relacionadas a falhas na instalação de dependências. Conforme apontado por [de Deus et al. 2025], a escolha de uma tecnologia deve considerar não apenas métricas de desempenho, mas também aspectos qualitativos como curva de aprendizado, estabilidade e custo de adoção. No presente estudo, isso evidencia que, embora Gradio se destaque em usabilidade e *deployment*, Streamlit e Dash possuem vantagens específicas dependendo do contexto e dos objetivos do projeto.

Em termos de confiabilidade, todos os frameworks apresentaram desempenho mediano, mantendo estabilidade e tolerância a falhas adequadas. Quanto à usabilidade, Streamlit e Dash tiveram pontuações razoáveis, e Gradio destacou-se por sua interface intuitiva. Em eficiência, todos demonstraram respostas rápidas e boa utilização de recursos. Na manutenibilidade e portabilidade, apresentaram arquitetura modular e boa adaptabilidade, enquanto em segurança Gradio se destacou em confidencialidade, com Streamlit apresentando pontuação inferior.

O *deployment* mostrou-se mais ágil no Gradio, mediano no Streamlit e limitado no Dash. A interoperabilidade com modelos treinados foi eficiente em Streamlit e Gradio, sendo menor no Dash devido à necessidade de maior configuração. Esses resultados evidenciam que, embora cada framework possua pontos fortes, há limitações que podem impactar o *deployment* e a integração de modelos, especialmente para usuários com diferentes níveis de experiência. Esse cenário e insights serviram de base para o desenvolvimento da QuickML Creator, buscando combinar os pontos positivos observados e minimizar as dificuldades encontradas nos frameworks avaliados.

O QuickML Creator demonstrou ser uma ferramenta prática para geração automatizada de interfaces de modelos de aprendizado de máquina, com fluxo contínuo desde a escolha do modelo até o download do código. A aplicação permite seleção do modelo, configuração do projeto e download de arquivos essenciais, como app.py, README.md e requirements.txt. Durante a validação, participaram 16 usuários com diferentes níveis de experiência: 60% iniciantes, 26,7% intermediários e 13,3% avançados. A avaliação indicou 80% de aprovação em funcionalidade, 86,7% em confiabilidade, média de 95,6% em usabilidade e 100% em eficiência de desempenho. Em satisfação geral e recomendação obtiveram 86,6% e as médias foram 4,31 e 4,6, respectivamente, evidenciando boa aceitação e destacando seu potencial educacional e para prototipagem ágil.

4. Considerações Finais

Este estudo comparou os frameworks Streamlit, Gradio e Dash para o desenvolvimento de aplicações *web* integradas a modelos de aprendizado de máquina e, a partir dessa análise, resultou na criação do QuickML Creator, ferramenta que automatiza a geração de aplicações de forma simples e rápida, avaliada quanto a desempenho, usabilidade, confiabilidade e qualidade de uso.

O Gradio destacou-se pela interface intuitiva e agilidade no *deployment*, seguido pelo Streamlit, que apresentou boa flexibilidade visual e integração. Já o Dash revelou limitações técnicas. Esses resultados reforçam que a escolha do framework deve consid-

erar o contexto, os objetivos do projeto e a experiência do usuário.

O QuickML Creator demonstrou eficiência na automatização de projetos, com altos índices de usabilidade, eficiência e confiabilidade, além de oferecer interface clara e fluxo estável. Os *feedbacks* apontaram potencial educacional para iniciantes e possibilidades de expansão, como suporte a novos frameworks e maior escalabilidade. Assim, o estudo cumpriu seus objetivos, evidenciando que o QuickML Creator representa um passo inicial relevante para a automatização de projetos de aprendizado de máquina, equilibrando desempenho, qualidade de software e acessibilidade. A ferramenta também confirma o potencial das plataformas *low-code/no-code* para o desenvolvimento ágil e o aprendizado prático, sugerindo caminhos futuros de evolução e integração com múltiplos frameworks.

References

Baier, L., Jöhren, F., and Seebacher, S. (2019). Challenges in the deployment and operation of machine learning in practice. In *ECIS*.

Braga, P. H. S. (2023). Implantação de modelos de aprendizado de máquina no formato onnx utilizando diferentes frameworks.

de Deus, M., Matos Junior, R. d. S., Júnior, G. L., and Quirino, G. d. S. (2025). Avaliação de desempenho de frameworks web java entre reatividade e threads virtuais. In *Anais do Workshop em Desempenho de Sistemas Computacionais e de Comunicação (WPERFORMANCE), 24*, pages 61–72.

de Freitas, D. S., da Silva, M. B., dos Santos, J. R., and Vicentini, R. M. (2025). Integração de inteligência artificial na indústria 4.0 utilizando ferramentas low-code para visão computacional. *Revista Brasileira De Mecatrônica — Brazilian Journal of Mechatronics*, 7(2):43–60.

dos Santos de Oliveira, M. (2025). Desenvolvimento ágil de aplicações web com plataforma low-code/no-code. Trabalho de conclusão de curso (tcc), Universidade do Extremo Sul Catarinense (UNESC). Orientador: Marcel Campos Inocencio.

International Organization for Standardization (2023). ISO/IEC 25010:2023 - Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. `https://www.iso.org/standard/78176.html`.

Oliveira, J. P. M. (2023). Análise comparativa da implantação de um modelo de classificação com aprendizado de máquina em diferentes níveis de mlops. Trabalho de Conclusão de Curso, Universidade Federal do Rio Grande do Norte.

Sommerville, I. (2003). *Engenharia de Software*. Addison Wesley, São Paulo, 6ª edition.

Sundberg, L. and Holmström, J. (2023). Democratizing artificial intelligence: How no-code ai can leverage machine learning operations. *Business Horizons*, 66(6):777–788.

Wirfs-Brock, A., Vissades, J., Cunningham, W., Johnson, R., and Bollette, L. (1990). Designing reusable designs (panel session) experiences designing object-oriented frameworks.