
# MH-AutoML: Transparência, Interpretabilidade e Desempenho na Detecção de *Malware* Android

**Joner Assolin[1], Gabriel Canto[1], Diego Kreutz[2], Eduardo Feitosa[1]**

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)

[2]LEA, PPGES – Universidade Federal do Pampa (UNIPAMPA)

diegokreutz@unipampa.edu.br,
{joner.assolin,gabriel.canto,efeitosa}@icomp.ufam.edu.br

***Resumo.*** *A MH-AutoML é uma ferramenta de AutoML especializada na detecção de malware Android. Diferentemente de outras ferramentas de AutoML, a MH-AutoML incorpora recursos de transparência, interpretabilidade e depuração em todos os estágios do pipeline. A ferramenta também inclui métodos de seleção de características específicos para o domínio e otimizações de hiperparâmetros que geram bons resultados. Os resultados indicam que a MH-AutoML produz modelos preditivos competitivos (e.g., 95% de recall com baixo custo computacional) em comparação com modelos gerados por outras 7 ferramentas de AutoML.*

## 1. Introdução

AutoML (*Automated Machine Learning*) refere-se à automação, em larga escala, de um amplo espectro do processo de aprendizado de máquina além da criação de modelos tradicionais, como pré-processamento de dados, meta-aprendizagem, aprendizagem de características, pesquisa de modelos, otimização de hiperparâmetros, classificação/regressão, geração de fluxos de trabalho, aquisição de dados e relatórios [Guyon, I. et. al., 2015]. Em poucas palavras, o objetivo principal das ferramentas de AutoML é reduzir a quantidade e o esforço de especialistas na realização de tarefas de aprendizado de máquina, permitindo, ao mesmo tempo, a geração de modelos capazes de atingir bons resultados em termos de métricas qualitativas, por exemplo.

Existem várias ferramentas de AutoML, como auto-sklearn[1], TPOT[2] e Google AutoML[3]. Essas ferramentas geralmente permitem obter resultados testáveis com menos tempo comparado a abordagens manuais, que envolvem especialistas, cientistas de dados e processos empíricos extensivos. O principal atrativo do AutoML é a redução da curva de aprendizagem, eliminando a necessidade de conhecimentos específicos em aprendizado de máquina e domínio. No entanto, por serem de propósito geral, essas ferramentas podem não incluir recursos otimizados para domínios específicos, afetando o pré-processamento, engenharia de características e otimização de modelos.

Mais recentemente, diversas soluções de AutoML começaram a surgir para domínios específicos, como o reconhecimento facial [Yan, C. et. al., 2022] e veículos

[1]https://automl.github.io/auto-sklearn/master/
[2]http://epistasislab.github.io/tpot/
[3]https://cloud.google.com/automl

autônomos [Shi, X. et. al., 2021]. Entretanto, vale destacar que cada domínio específico apresenta desafios próprios, como a compreensão do problema, a operacionalização e a avaliação das previsões [Karmaker S. et. al., 2021], além da personalização [Xin, D. et. al., 2021]. Ferramentas especializadas em AutoML podem enfrentar esses desafios ao explorar as particularidades de cada domínio. No entanto, desafios transversais, como transparência e interpretabilidade [Xin, D. et. al., 2021], assim como recursos de depuração [Lee and Macke, 2020], ainda não são bem abordados pelas ferramentas de AutoML existentes.

Neste trabalho apresentamos a MH-AutoML[4], uma ferramenta de domínio específico que implementa um *pipeline* de AutoML completo, abstraindo a execução das etapas de limpeza de dados, engenharia de características, escolha de algoritmos, ajuste de hiperparâmetros e otimização. Adicionalmente, a MH-AutoML também implementa recursos de transparência, interpretabilidade, depuração e rastreamento de experimentos em todas as etapas do *pipeline*, o que é incomum em outras ferramentas de AutoML, que funcionam tipicamente como "caixas pretas" (e.g., apenas apresentação das métricas finais aos usuários).

## 2. A Ferramenta MH-AutoML

A Figura 1 ilustra o *pipeline* de AutoML da MH-AutoML, com suas macro tarefas necessárias para a execução do fluxo de aprendizado de máquina, incluindo etapas clássicas como pré-processamento dos dados, engenharia de características e seleção de modelo e ajuste de modelo.

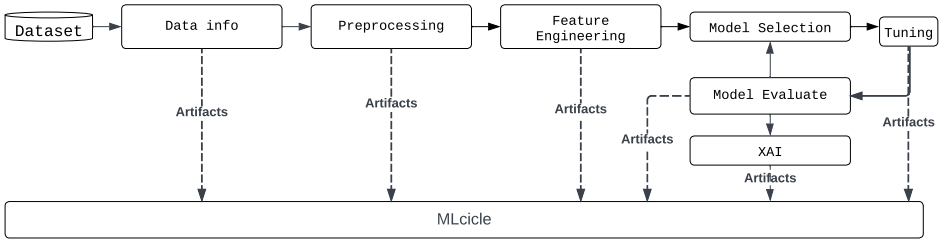


**Figura 1. *Pipeline* da MH-AutoML.**

A primeira etapa (*Data info*) é uma breve análise exploratória que fornece um resumo sobre o sistema e o *dataset* de entrada. Em termos de informações do sistema, a ferramenta coleta automaticamente detalhes do sistema operacional, incluindo métricas de memória RAM e disco (total, livre, utilizado), CPU (total, utilizado) e rede (número de bytes enviados e recebidos). Em relação ao *dataset*, são apresentados o número total e tipo de amostras e características, o nível de desbalanceamento entre as classes e outras informações de integridade dos dados.

Na etapa de *pré-processamento* (*Preprocessing*) dos dados ocorre a correção de inconsistências, normalização e transformação dos dados. Exemplos comuns de inconsistências incluem dados ausentes, tipos incorretos, dados redundantes e preenchidos incorretamente. Na versão atual, a ferramenta realiza a detecção e correção

[4]`https://github.com/SBSegSF24/MH-AutoML`

de erros através de várias sub-etapas, como a remoção de valores discrepantes (*outliers*), remoção de duplicatas, remoção de valores ausentes e a codificação *one-hot* das características se necessário. A ferramenta também gera e disponibiliza diferentes artefatos visuais, como gráficos de mapa de calor das "sujeiras" encontradas no *dataset*. Mais detalhes e exemplos desses artefatos visuais podem ser encontrados no GitHub [Assolin, J. et. al., 2024].

Na *engenharia de características* (*Feature Engineering*), a ferramenta incorpora diferentes métodos clássicos, como PCA (*Principal Component Analysis*), ANOVA (*Analysis of Variance*) e LASSO (*Least Absolute Shrinkage and Selection Operator*). Vale ressaltar que métodos como PCA e ANOVA estão entre os mais frequentemente utilizados por ferramentas de AutoML. Já o LASSO foi incorporado após realizarmos experimentos empíricos extensivos com mais de 20 métodos de seleção de características e uma dezena de conjuntos de dados significativamente heterogêneos. Os resultados indicaram que o LASSO apresenta uma capacidade de generalização superior aos demais métodos.

Na etapa de *seleção de modelo* (*Model Selection*), a MH-AutoML considera os algoritmos de aprendizado de máquina, o *dataset* e o problema de entrada. Utilizamos o método de votação `VotingClassifier` para combinar as previsões de múltiplos algoritmos, como *LightGBM*, *KNN*, *CatBoost*, *RandomForestClassifier*, *DecisionTreeClassifier* e *ExtraTreesClassifier*. O `VotingClassifier` aproveita as vantagens de diferentes modelos para oferecer uma solução mais robusta e precisa. Além disso, novos algoritmos (e.g., `SVM`) podem ser facilmente adicionados à lista de modelos considerados nesta etapa.

Vale destacar que a MH-AutoML conta com modelos de interpretabilidade intrínseca, isto é, com capacidade de compreensão nativa, o que elimina a necessidade de métodos adicionais posteriores para a explicação dos modelos. Modelos como *KNN*, *DecisionTreeClassifier*, *ExtraTreesClassifier* e *CatBoost* exibem diferentes níveis de interpretabilidade intrínseca. O *DecisionTreeClassifier*, por exemplo, é considerado intrinsecamento interpretável devido à sua estrutura baseada em regras simples, onde cada decisão é clara e compreensível.

O *ajuste do modelo* (*Tuning*) envolve a escolha e avaliação dos valores para os parâmetros de entrada do algoritmo selecionado anteriormente, com cada algoritmo exigindo uma combinação específica de hiperparâmetros que controlam o processo de aprendizado do modelo. A MH-AutoML utiliza a biblioteca *Optuna* para otimização de hiperparâmetros, realizando uma busca eficiente e automatizada dos melhores valores para melhorar o desempenho do modelo.

Na detecção de *malware* Android, a métrica mais relevante é o *recall*, que indica a porcentagem de amostras maliciosas corretamente classificadas. A MH-AutoML prioriza o *recall* como critério principal na seleção dos melhores hiperparâmetros, assegurando alta eficácia na identificação de *malware*.

É crucial observar ainda a utilização das técnicas LIME e SHAP na parte de explicabilidade (*XAI*), para a explicação e interpretabilidade complementar dos modelos, oferecendo uma compreensão mais profunda sobre o impacto de cada característica nos modelos preditivos. Além disso, a MH-AutoML incorpora também

a plataforma aberta de MLOps[5] denominada MLflow para o gerenciamento do ciclo de vida do *pipeline* de AutoML, viabilizando o versionamento de modelos, o armazenamento e gerenciamento de artefatos (arquivos, imagens, códigos), e a comparação de experimentos, garantindo uma rastreabilidade completa e facilitando a colaboração entre equipes, essas tarefas não são apresentadas em outras ferramentas, mesmo as que integram MLflow. Mais detalhes e exemplos sobre os artefatos (e.g., gráficos, comparações) gerados pela MH-AutoML em todos os estágios do *pipeline* de AutoML podem ser vistos no GitHub da ferramenta [Assolin, J. et. al., 2024].

## 3. Implementação

Para implementar a MH-AutoML, utilizamos a linguagem de programação Python (versão 3.9.7) e diversas bibliotecas de código aberto amplamente utilizadas em modelos de análise de dados e aprendizado de máquina. Para análise, limpeza e manipulação de dados, empregamos o `pandas` (versão 1.5.3) e o `numpy` (versão 1.24.0) para operações matemáticas. Utilizamos a biblioteca `scikit-learn` (versão 1.1.3) para implementar os algoritmos *RandomForestClassifier*, *DecisionTreeClassifier*, *ExtraTreesClassifier*, *KNN*, *LightGBM* e *CatBoost*. Para otimização automática de hiperparâmetros, adotamos a biblioteca `optuna` (versão 2.10.1). Utilizamos ainda a plataforma aberta de MLOps `MLflow` (versão 2.11.3) para o gerenciamento do ciclo de vida do *pipeline* de AutoML e as bibliotecas `LIME` (versão 0.2.0.1) e `SHAP` (versão 0.39.0) para a explicação e interpretabilidade dos modelos.

Para melhorar a usabilidade da ferramenta em termos de interpretabilidade e depuração, cada etapa da execução é acompanhada por registros de eventos `INFO` (informacional) e `CRITICAL` (problema de execução). As mensagens `INFO` permitem aos usuários monitorar o progresso da execução, ajudando a mitigar o problema da "caixa preta" comum em ferramentas de AutoML. Exemplos de saídas estão disponíveis no GitHub [Assolin, J. et. al., 2024], evidenciando como essas mensagens ajudam na compreensão do processo.

As mensagens `CRITICAL` são essenciais para identificar e tratar problemas, como erros de dados no pré-processamento ou falhas na seleção de características. Elas fornecem informações cruciais para a interpretabilidade e transparência, alertando os usuários sobre problemas específicos relacionados aos dados de entrada ou métodos utilizados. Isso é fundamental para garantir que a ferramenta evolua continuamente e minimize o risco de gerar modelos enviesados.

Diferentemente de outras ferramentas de AutoML, é importante destacarmos que a MH-AutoML exibe informações detalhadas sobre cada etapa do fluxo do *pipeline* de aprendizado de máquina. Mais detalhes sobre as saídas da execução em todos os estágios do *pipeline* e a implementação podem ser encontrados no repositório GitHub da ferramenta [Assolin, J. et. al., 2024].

## 4. Experimentação

A seguir apresentamos os *datasets* utilizados, o ambiente de experimentação, as métricas e os resultados de desempenho da ferramenta. Comparamos os resultados da MH-AutoML com outras 7 ferramentas de AutoML.

[5] `https://ml-ops.org`

### 4.1. *Datasets*

Para avaliar a MH-AutoML, utilizamos os 7 *datasets* apresentados na Tabela 1. Como pode ser observado, a quantidade e os tipos de características variam entre os conjuntos de dados. Todos os *datasets*, bem como referências aos repositórios originais, estão disponíveis no GitHub [Assolin, J. et. al., 2024].

**Tabela 1. *Datasets* utilizados nos experimentos.**

| Dataset | Características | | Amostras | | |
|---|---|---|---|---|---|
| | Qtde. | Tipos | Maliciosas | Benignas | Total |
| adroit | 166 | P | 3418 | 8058 | 11476 |
| androcrawl | 141 | A(26), I(8), P(84), O(23) | 10170 | 86574 | 96744 |
| android_permissions | 151 | P | 17787 | 9077 | 26864 |
| defensedroid_prs | 2877 | P(1489), I(1388) | 6000 | 5975 | 11975 |
| drebin | 215 | A(73), P(113), S(6), I(23) | 5555 | 9476 | 15031 |
| kronodroid_emulador | 276 | P(145), A(123), O(8) | 28745 | 35246 | 63991 |
| kronodroid_real | 286 | P(146), A(100), O(40) | 41382 | 36755 | 78137 |

**[P] Permissões, [A] Chamadas de API, [I] Intenções, [S] Comandos do Sistema, [O] Outros**

### 4.2. Ambiente e Métricas

As ferramentas de AutoML foram executadas em um computador com processador Intel(R) Core(TM) i7-1185G7 3.00GHz 11th 8 Cores, 32GB RAM, HD, com sistema operacional *host* MS-Windows 10 64 bit. Para a avaliação comparativa escolhemos as ferramentas de AutoML Auto-Sklearn, AutoPyTorch, LightAutoML, MLJar, AutoGloun, HyperGBM e TPOT por serem livres e de código aberto. Na comparação das ferramentas utilizamos as métricas de acurácia, precisão, pontuação F1 e *recall* e tempo de execução.

### 4.3. Resultados

A Figura 2 apresenta o mapa de calor com os resultados de desempenho das 8 ferramentas de AutoML para os 7 *datasets*. Os números representam a média das duas classes, isto é, benigna e maligna. Como podemos observar, a MH-AutoML obteve resultados competitivos quando comparada com ferramentas mais conhecidas e conceituadas na literatura, como a AutoGluon e a Auto-Sklearn, atingindo resultados de *recall* similares ou superiores para a maioria dos conjuntos de dados.

A MH-AutoML atingiu um *recall* superior a ferramentas como AutoGluon e Auto-Sklearn para os *datasets* como *kronodroid_emulador*, *drebin*, *defensedroid_prs* e *androcrawl*. No contexto geral, a exceção foi o conjunto de dados *drebin*, onde a TPOT se saiu ligeiramente melhor que as demais ferramentas. Já no caso da pontuação F1, a melhor ferramenta para o *drebin* foi a AutoGluon.

A análise do desempenho inferior da MH-AutoML no dataset *android_permissions* pode ser atribuída ao comportamento da ferramenta durante a fase de pré-processamento. Neste caso específico, a MH-AutoML conseguiu uma alta taxa de *recall* para amostras de *malware* (96%), mas apresentou um *recall* muito baixo para amostras benignas (apenas 12%). Esse desempenho se deve à eliminação de 1353 amostras pertencentes à classe benigna, que foram removidas por serem compostas exclusivamente por valores zero, indicando sua ausência nas

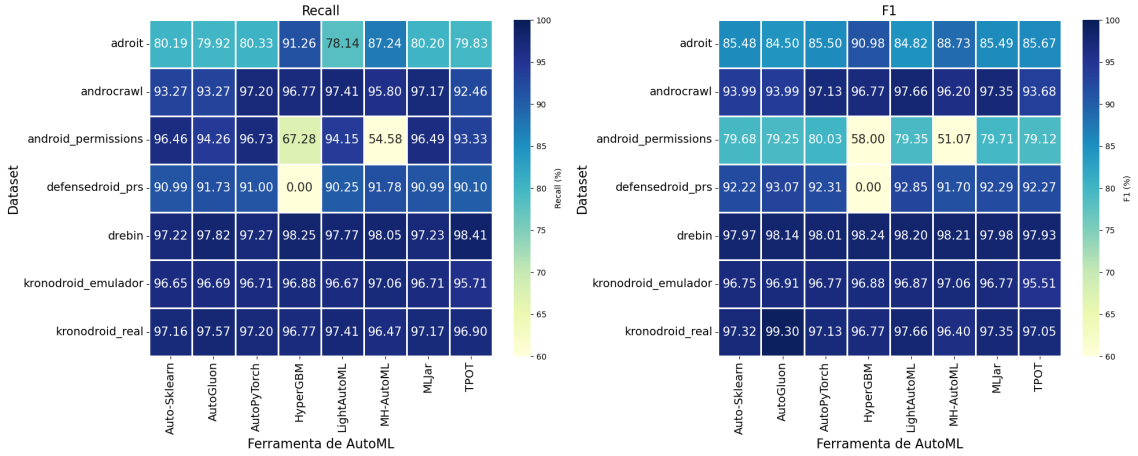


**Figura 2. Mapa de calor do *recall* e pontuação F1 das 8 ferramentas de AutoML.**

amostras. Ao contrário de outras ferramentas, a MH-AutoML oferece informações detalhadas por classe, permitindo análises mais precisas e assertivas.

Finalmente, a MH-AutoML demonstrou também um tempo de execução comparativamente muito bom, ficando entre as 4 ferramentas de melhor tempo para todos os *datasets*, como pode ser observado no mapa de calor da Figura 3. É interessante destacar que a MH-AutoML supera ferramentas conhecidas e frequentemente utilizadas, como Auto-Sklearn, AutoPyTorch e TPOT. Vale destacar também que ferramentas como AutoGloun e HyperGBM possuem a etapa de hiper-parâmetro simplificada, utilizando hiperparâmetros predefinidos, o que explica o seu menor tempo de execução. Entretanto, essa estratégica acaba comprometendo ligeiramente os resultados de desempenho, como pode ser observado na Figura 2. Podemos observar também que a ferramenta HyperGBM falhou durante a execução no *dataset* *defensedroid_prs*, essa falha ocorre devido a uma falha na ferramenta em processar caracteres especiais nos nomes das características.

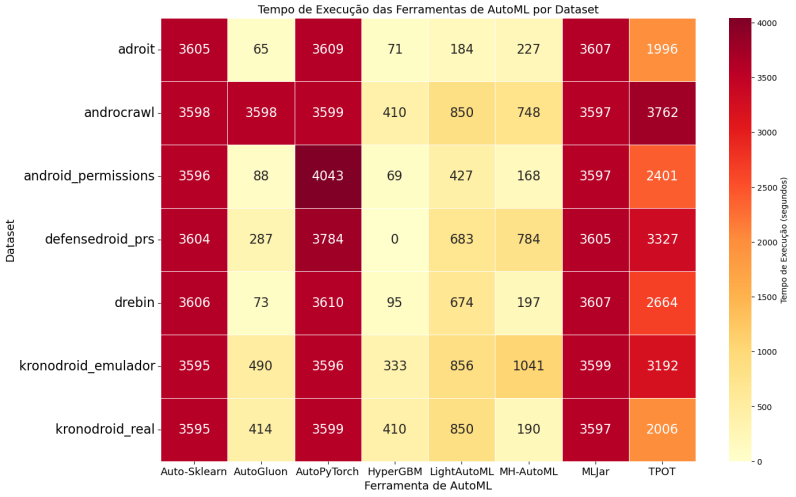


**Figura 3. Dados de tempo de execução das ferramentas de AutoML.**

## 5. Trabalhos Relacionados

A Tabela 2 apresenta uma relação das principais ferramentas de AutoML de código aberto e livres. Como pode ser observado, todas as ferramentas incluem tarefas de classificação, que é o foco atual da MH-AutoML. Entretanto, nenhum das outras ferramentas suporta versionamento de modelos e artefatos do *pipeline* de AutoML e poucas coletam informações e métricas de sistema durante a execução.

**Tabela 2. Trabalhos Relacionados**

| Referências | Ano | Ferramenta | Tarefas | Explicabilidade | Rastreamento | Versionamento | Métricas de sistema | Transparência e Depuração |
|---|---|---|---|---|---|---|---|---|
| [Jin et al., 2023] | 2023 | AutoKeras | Classificação, Regressão | - | - | Não | Não | T1, T2, D1, D2 |
| [Nasimian, A. et. al., 2024] | 2023 | AlphaML | Classificação, Regressão, Agrupamento, *Forecasting* | Feature importance, Permutation importance, SHAP, LIME | - | Não | Não | T1, T2, D1 |
| [LeDell and Poirier, 2020] | 2020 | H2O AutoML | Classificação, Regressão | Feature importance, Partial Dependence, TreeSHAP, ICE | MLflow, H2O Flow | Não | Sim | T1, T2, D1, D2 |
| [Erickson, N. et. al., 2020] | 2020 | AutoGluon | Classificação, regressão | Feature importance | - | Não | Sim | T1, T2, D1, D2 |
| [Molino P. et. al., 2019] | 2019 | LUDWIG | classificação, regressão | - | MLflow, Comet, wandb | Não | Não | T1, T2, D1, D2 |
| [Zimmer et al., 2000] | 2020 | Auto-PyTorch | Classificação, Regressão, *Forecasting* | - | - | Não | Não | T1, T2, D1, D2 |
| [Olson and Moore, 2016] | 2020 | TPOT | Classificação, Regressão | - | - | Não | Não | T1, T2, D1, D2 |
| Este trabalho | 2024 | MH-AutoML | Classificação | SHAP, LIME, Feature Importance | MLflow | Sim | Sim | T1, T2, D1, D2 |

Em termos de explicabilidade, a utilização de *frameworks* de IA explicável, como SHAP e LIME, aparece apenas em duas ferramentas, a AlphaML e a MH-AutoML. Já com relação à transparência e depuração, temos as características T1, T2, D1 e D2. Na transparência, T1 indica que a ferramenta apresenta a relação de algoritmos utilizados no *pipeline*, enquanto T2 indica que os hiperparâmetros dos modelos são exibidos. Na depuração, D1 indica que a ferramenta apresenta *logs* durante o treinamento, validação e testes, enquanto que D2 indica que são exibidos *logs* específicos sobre erros e avisos.

Com relação ao rastreamento de experimentos, a MH-AutoML se destaca por sua integração nativa com a MLflow, fornecendo métricas detalhadas do modelo e do sistema, além de informações completas sobre o *pipeline* de execução. Ademais, a MH-AutoML também organiza os artefatos de todos os estágios do *pipeline*, o que contribui significativamente para a transparência e depuração. Por exemplo, a ferramenta fornece métricas e relatórios de desempenho para cada estágio individual do *pipeline*, permitindo uma análise de desempenho mais aprofundada e qualitativa.

## 6. Demonstração e Considerações Finais

**Demonstração**. A ferramenta será apresentada através de uma máquina virtual em ambiente hospedado em dispositivo próprio dos autores. As funcionalidades e saídas da MH-AutoML serão demonstradas e discutidas através de: (i) apresentação dos parâmetros de configuração e execução; (ii) apresentação das etapas do *pipeline* de AutoML; (iii) detalhamento dos *datasets* de entrada; (iv) analise dos artefatos gerados pela MH-AutoML;

**Considerações Finais**.

A MH-AutoML oferece recursos avançados de explicabilidade, transparência e depuração em comparação com outras ferramentas de AutoML. Ela inclui métodos de seleção de características e modelos específicos para a classificação de *malware* Android e recursos de versionamento e rastreamento integrados com o MLflow, permitindo análises detalhadas do *pipeline* de AutoML. Os resultados iniciais mostram que a MH-AutoML pode gerar modelos competitivos com ferramentas de AutoML livres e de código aberto, como AutoGluon e H2O AutoML. Assim, sua versão atual já demonstra potencial para competir com ferramentas existentes e contribuir para pesquisas futuras na detecção de *malware* Android.

**Agradecimentos**. Esta pesquisa foi financiada, conforme previsto nos Arts. 21 e 22 do decreto no. 10.521/2020, nos termos da Lei Federal no. 8.387/1991, através do convênio no. 003/2021, firmado entre ICOMP/UFAM, Flextronics da Amazônia Ltda. e Motorola Mobility Comércio de Produtos Eletrônicos Ltda. Este trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Código de Financiamento 001 e parcialmente financiado pela FAPEAM – por meio do projeto POSGRAD 2024/2025 e pela FAPERGS, através dos editais 08/2023 e 09/2023.

## Referências

Assolin, J. et. al. (2024). MH-AutoML. `https://github.com/SBSegSF24/MH-AutoML`.

Erickson, N. et. al. (2020). Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*.

Guyon, I. et. al. (2015). Design of the 2015 chalearn automl challenge. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Jin, H., Chollet, F., Song, Q., and Hu, X. (2023). Autokeras: An automl library for deep learning. *Journal of machine Learning research*, 24(6):1–6.

Karmaker S. et. al. (2021). Automl to date and beyond: Challenges and opportunities. *ACM Computing Surveys*, 54(8).

LeDell, E. and Poirier, S. (2020). H2o automl: Scalable automatic machine learning. In *Proceedings of the AutoML Workshop at ICML*, volume 2020.

Lee, D. J.-L. and Macke, S. (2020). A human-in-the-loop perspective on automl: Milestones and the road ahead. *IEEE Data Engineering Bulletin*.

Molino P. et. al. (2019). Ludwig: a type-based declarative deep learning toolbox.

Nasimian, A. et. al. (2024). Alphaml: A clear, legible, explainable, transparent, and elucidative binary classification platform for tabular data. *Patterns*, 5(1).

Olson, R. S. and Moore, J. H. (2016). TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning*.

Shi, X. et. al. (2021). An automated machine learning (automl) method of risk prediction for decision-making of autonomous vehicles. *IEEE TITS*, 22(11):7145.

Xin, D. et. al. (2021). Whither automl? understanding the role of automation in machine learning workflows. In *Proceedings of the CHI*.

Yan, C. et. al. (2022). Privacy-preserving online automl for domain-specific face detection. In *IEEE CVF*, pages 4134–4144.

Zimmer, L., Lindauer, M., and Hutter, F. (2000). Auto-pytorch tabular: Multi-fidelity metalearning for efficient and robust autodll. arxiv 2020. *arXiv preprint arXiv:2006.13799*.