QuickAutoML: Uma ferramenta para treinamento automatizado de modelos de aprendizado de máquina (versão estendida)*

Guilherme Siqueira[1], **Gustavo Rodrigues**[1,3], **Eduardo Feitosa**[2], **Diego Kreutz**[1]

[1]Laboratório de Estudos Avançados em Computação (LEA)
Programa de Pós-Graduação em Engenharia de Software (PPGES)
Universidade Federal do Pampa (Unipampa)

[2]Grupo de Pesquisa em Tecnologias Emergentes e Segurança de Sistemas (ETSS)
Instituto de Computação (IComp)
Universidade Federal do Amazonas (UFAM)

[3]Combate à Fraude

GuilhermeSiqueira.aluno@unipampa.edu.br
gustavo.rodrigues@combateafraude.com
efeitosa@icomp.ufam.br,diegokreutz@unipampa.edu.br

Resumo. *Com o aumento da popularidade dos modelos de aprendizado de máquina em diferentes domínios e contextos, aumentou também a necessidade por ferramentas e mecanismos capazes de facilitar a otimização e utilização prática desses modelos. Neste trabalho, apresentamos a QuickAutoML, uma ferramenta que automatiza os processos de ajuste de hiperparâmetros, criação, treinamento e validação de modelos. O principal objetivo da ferramenta é abstrair a complexidade desses processos para permitir que usuários com pouco conhecimento técnico possam criar modelos robustos em questão de minutos.*

1. Introdução

Aprendizado de máquina (ou *machine learning*) tem por objetivo melhorar o desempenho de uma solução computacional motivada por determinado problema (e.g., classificar *malwares*) através de aprendizado contínuo [Yao and Liu, 2014]). O desenvolvimento de soluções baseadas em aprendizado de máquina envolve diversos processos como tratamento e manipulação de dados, engenharia de características (ou *feature engineering*) e seleção de algoritmos. O fato de tais processos muitas vezes ocorrerem de forma *ad hoc*, aliado à necessidade de conhecimento do domínio do problema a ser solucionado, torna o desenvolvimento de modelos de aprendizado de máquina custosos em tempo, fortemente dependente de conhecimento técnico especializado e difícil de ter sua viabilidade avaliada.

Ferramentas como autosklearn [Feurer et al., 2015] e TPOT [Olson and Moore, 2016] automatizam esses processos, permitindo obter artefatos testáveis e com um custo de tempo menor em comparação com abordagens manuais. Em particular, essas ferramentas reduzem substancialmente a curva de aprendizagem, uma vez que eliminam a necessidade de diversos conhecimentos específicos da área de

*Este artigo é uma versão estendida do paper [Siqueira et al., 2021], de 6 páginas, originalmente publicado no WRSeg 2021 e convidado para publicação na ReABTIC.

aprendizado de máquina. Entretanto, o usuário ainda necessita realizar ajustes finos manualmente (*e.g.,* definição do espaço de busca, configurações de CPU e memória), o que elimina apenas parcialmente o problema da dependência de conhecimento especializado. Além disso, essas ferramentas consome tipicamente horas (ou até mesmo dias) para explorar as opções de otimização, seleção e treinamento de modelos, aumentando o custo computacional da validação inicial de um projeto de aprendizado de máquina.

Neste trabalho, propomos a QuickAutoML, que permite desenvolver modelos de aprendizado de máquina sem necessidade de conhecimento técnico especializados. A ferramenta identifica o algoritmo que melhor se ajusta a determinado problema e realiza a otimização desse algoritmo através de ajustes finos automatizados (i.e., seleciona automaticamente hiperparâmetros otimizados). Em uma análise experimental (Seção 3), demonstramos, utilizando 28 *datasets* distintos, que a QuickAutoML consegue atingir resultados semelhantes as ferramentas TPOT e autosklearn, em termos de acurácia, mas com um tempo de execução consideravelmente menor.

2. Implementação

A QuickAutoML é um arcabouço para os algoritmos de aprendizado de máquina fornecidos pelo scikit-learn, uma biblioteca que implementa diversos algoritmos para a linguagem Python [Pedregosa et al., 2011]. A Figura 1 ilustra o processo de treinamento de modelos na QuickAutoML.

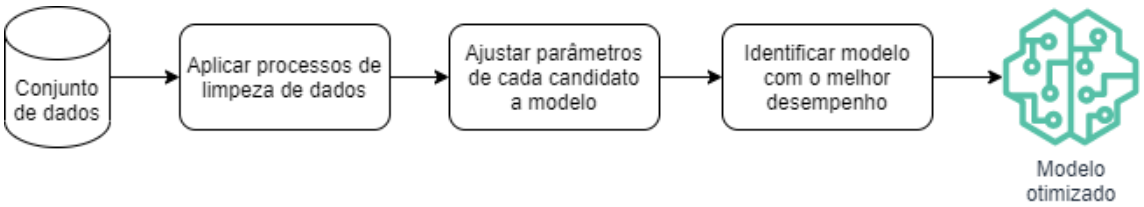

Figura 1. Processo de treinamento

O ponto de partida é o fornecimento de um conjunto de dados (i.e., *dataset*) que será utilizado para treinamento, separado em dados de entrada e dados de saída, de forma que cada amostra de entrada (variável preditora) esteja associada a uma amostra de saída (variável alvo). A próxima etapa é treinar os modelos a fim de obter o que melhor se ajusta aos dados; o parâmetro pelo qual obtemos o melhor modelo é definido na seção 2.2. Nesse treinamento, são criados e testados diversos candidatos com diferentes configurações (ver detalhes na seção 2.3).

2.1. Limpeza de dados

Limpeza de dados é o processo responsável pela identificação e remoção de inconsistências presentes nos dados, com o objetivo de aumentar a qualidade dos mesmos [Rahm and Do, 2000]. Tais inconsistências podem ser originadas por vários fatores, como dados ausentes, tipos de dados incorretos, dados redundantes ou simplesmente dados que foram preenchidos incorretamente. No escopo de aprendizado de máquina, o objetivo da aplicação de limpeza de dados é facilitar o aprendizado do algoritmo. A QuickAutoML automatiza parte desse processo, implementando a detecção e remoção de valores ausentes e registros duplicados.

2.2. Métricas de avaliação

Métricas de avaliação podem ser definidas como aspectos de desempenho de um modelo, tornando mensurável a diferença entre o real e o que foi predito [Botchkarev, 2018]. Para realizar a comparação entre diferentes modelos, é essencial definir uma métrica que servirá como parâmetro para a avaliação. A escolha entre diferentes métricas pode influenciar diretamente no resultado da comparação, pois diferentes modelos tendem a apresentar valores diferentes para um conjunto de métricas [Caruana and Niculescu-Mizil, 2006].

A escolha da métrica reflete diretamente os requisitos do problema. Por exemplo, em aplicações onde é mais crítico reduzir a taxa de falsos negativos gerados na classificação, uma alta revocação é mais significante para determinar a qualidade do modelo do que uma alta precisão. Por padrão, na QuickAutoML, a métrica utilizada para comparação dos modelos é a acurácia (*i.e.*, a porcentagem de predições corretas feitas pelo modelo), porém, o usuário pode utilizar outras métricas de classificação (precisão, revocação ou F1 *score*).

2.3. Treinamento e seleção de modelos

Para cada algoritmo, é fornecida uma combinação diferente de hiperparâmetros (*i.e,* valores que controlam o processo de aprendizado de determinado modelo [Wu et al., 2019]). Por exemplo, algoritmos baseados em árvores necessitam que sejam definidos valores para profundidade máxima, número máximo de nós-folha e critério para medir a qualidade de uma partição. A otimização manual desses hiperparâmetros é uma tarefa repetitiva, tediosa e com sucesso puramente baseado no acaso.

O processo de treinamento é iniciado através da definição de uma lista de potenciais candidatos a modelos. Essa lista é populada em duas etapas: primeiramente, definimos as famílias de algoritmos, como métodos baseados em *bagging* ou em distância. Como demonstrado pelo algoritmo 1, a entrada do processo de ajuste de hiperparâmetros é iniciado através do fornecimento de dois conjuntos de dados (denotados por X, representando os dados de treinamento; e y, representando o vetor de dados de saída). Com esses dados, vários candidatos a modelo são treinados e avaliados considerando determinada métrica de desempenho. O número de modelos testados é igual ao número de algoritmos multiplicado pelo número de combinações de hiperparâmetros possíveis. Com isso, o modelo resultante que possuir o maior valor para tal métrica é selecionado como o modelo representativo para o conjunto de dados em questão.

Para evitar problemas pós-treinamento, como *overfitting* (*i.e.*, condição na qual o modelo se ajusta bem nos dados de treino, porém perde sua capacidade de generalização para dados novos), é utilizada uma abordagem chamada validação cruzada com k dobras. A utilização da validação cruzada no treinamento é descrita no algoritmo 1. Essa abordagem adiciona uma camada de treinamento extra ao particionar os dados k vezes, de acordo com a necessidade. Na ferramenta, utilizamos $k = 5$ por ser um valor tipicamente utilizado e apresentar tempo menor de execução em comparação a outras técnicas de validação (*e.g.*, LOOCV) [James et al., 2013]. Na prática, isso significa dividir o conjunto de dados em cinco subconjuntos e efetuar cinco rodadas de treino e teste. Em cada rodada, o modelo é treinado com quatro dessas partições e testado com a partição restante. Ao final, é agregada a acurácia média das avaliações. O modelo que possuir a maior acurácia média é escolhido.

Algorithm 1 Encontrar o melhor modelo

$X \leftarrow$ a matriz de features
$y \leftarrow$ o vetor de variáveis de saída
$scores \leftarrow$ um mapa vazio
for $model$ na lista de modelos **do**
    $metric\_values \leftarrow$ uma lista vazia
    for $train\_index, test\_index$ no kfold **do**
        $X\_train \leftarrow X[x\_train]$
        $y\_train \leftarrow y[y\_train]$
        Treinar o modelo com X_train e y_train
        $predictions \leftarrow$ Predições feitas pelo modelo
        Adicionar a $metric\_values$ o valor da métrica para essa instância de treino
        Atualizar $scores$ com $model$ como chave e a média de $metric\_values$ como valor
    end for
end for
$mean\_metrics \leftarrow$ uma lista vazia
for $values\_list$ nos valores de $scores$ **do**
    Adicionar a média de $values\_list$ para $mean\_metric$
end for
return O modelo que satisfaz $\text{argmax}(mean\_metrics)$

3. Avaliação

Para avaliar o desempenho da QuickAutoML, realizamos uma comparação com as ferramentas TPOT e autosklearn, utilizando como critérios de comparação a acurácia obtida pelo modelo resultante e o tempo de execução que cada ferramenta utilizou para encontrar um modelo otimizado. Na Seção 3.1 detalhamos o experimento realizado.

3.1. Metodologia

No experimento, utilizamos *datasets* específicos[1] para *benchmark* de algoritmos de aprendizado de máquina supervisionado [Olson and Moore, 2016]. Para chegar aos 28 *datasets*, primeiramente, aplicamos um filtro para selecionar aqueles destinados a tarefas de classificação. Num segundo momento, categorizamos os *datasets* segundo sua dimensionalidade (calculada através da multiplicação entre o número de linhas e o número de colunas do *dataset*). Distribuímos os *datasets* em 7 (sete) categorias de acordo com a dimensionalidade, representadas por letras de A a G, descritas na Tabela 1.

Após isso, foram selecionados, de forma aleatória, 4 (quatro) *datasets* de cada categoria para compôr o conjunto de avaliação, resultando em um total de 28 *datasets*. O conjunto de avaliação é demonstrado na Tabela 2.

Para ajustar a proporção de tempo de execução, considerando que tanto a TPOT quanto autosklearn possuem uma abordagem que visa a obtenção de modelos mais robustos em detrimento de um tempo maior de execução, configuramos os parâmetros de execução em: (a) `generations=5` e `population_size=20`, para o classificador da TPOT[2]; e (b) `time_left_for_this_task=120` e `per_run_time_limit=30`,

[1] `https://github.com/EpistasisLab/pmlb/tree/master/datasets`
[2] `http://epistasislab.github.io/tpot/using/`

Tabela 1. Categorias de *datasets* por dimensionalidade

Categoria	Faixa
A	501 - 1,000
B	1,001 - 5,000
C	5,001 - 10,000
D	10,001 - 25,000
E	25,001 - 50,000
F	50,001 - 100,000
G	100,001 - 200,000

para o classificador da autosklearn[3]. Esses valores foram utilizados seguindo as recomendações da documentação oficial de cada uma das ferramentas.

A configuração do ambiente utilizado para o experimento é: processador 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz, 32GB de memória RAM, Python 3.8.10, PyCharm IDE 212.5080.64, JRE 11.0.11 e Ubuntu 20.04.2 LTS.

3.2. Resultados

Podemos observar na Tabela 3 (com a abreviação AutoSk representando a autosklearn e abreviação Quick representando a QuickAutoML) que, em 24 dos conjuntos de dados de avaliação, tanto TPOT quanto autosklearn geraram modelos que obtiveram uma diferença menor do que 5% de acurácia em relação aos modelos gerados pela QuickAutoML. Contudo, com uma diferença consideravelmente maior em relação ao tempo de execução. Além disso, em 6 e 10 *datasets*, a QuickAutoML encontrou modelos com acurácia superior à TPOT autosklearn, respectivamente.

Os resultados são explicados pela abordagem que, tanto autosklearn quanto TPOT, implementam: ambas visam treinar um *pipeline* de dados completo, considerando processos como normalização de dados, *feature engineering*, redução de dimensionalidade, entre outros, através de técnicas como evolução de algoritmos genéticos (no caso da TPOT) e otimização bayesiana (no caso da autosklearn). Essas abordagens adicionam uma complexidade maior ao treinamento e que, em boa parte dos casos, não agrega um valor muito superior em termos de desempenho dos modelos resultantes.

Os piores casos (i.e. diferença maior do que 10 na acurácia entre a QuickAutoML e as outras ferramentas) apresentados na avaliação ocorreram nos *datasets GAMETES_Epistasis_3_Way_20atts_0.2H_EDM_1_1*, *GAMETES_Epistasis_2_Way_20atts_0.1H_EDM_1_1* e *GAMETES_Epistasis_2_Way_20atts_0.4H_EDM_1_1*. Nesses conjuntos de dados, é notável a presença de uma grande proporção de características do tipo categórico (representando

[3]https://automl.github.io/auto-sklearn/master/examples/20_basic/example_classification.html#sphx-glr-examples-20-basic-example-classification-py

Tabela 2. *Datasets* utilizados para avaliação da ferramenta

Nome	N. Amostras	N. Features	N. Classes	Dimensionalidade	Grupo
haberman	306	3	2	918	A
tae	151	5	3	755	A
iris	150	4	3	600	A
mux6	128	6	2	768	A
penguins	344	7	3	2408	B
prnn_crabs	200	7	2	1400	B
prnn_fglass	205	9	5	1845	B
analcatdata_dmft	797	4	6	3188	B
spect	267	22	2	5874	C
breast	699	10	2	6990	C
diabetes	768	8	2	6144	C
horse_colic	368	22	2	8096	C
led7	3200	7	10	22400	D
soybean	675	35	18	23625	D
sonar	208	60	2	12480	D
german	1000	20	2	20000	D
GAMETES_Epistasis_2_Way _20atts_0.4H_EDM_1_1	1600	20	2	32000	E
GAMETES_Epistasis_2_Way _20atts_0.1H_EDM_1_1	1600	20	2	32000	E
GAMETES_Epistasis_3_Way _20atts_0.2H_EDM_1_1	1600	20	2	32000	E
phoneme	5404	5	2	27020	E
mfeat_zernike	2000	47	10	94000	F
wine_quality_white	4898	11	7	53878	F
hypothyroid	3163	25	2	79075	F
churn	5000	20	2	100000	F
kr_vs_kp	3196	36	2	115056	G
allbp	3772	29	3	109388	G
agaricus_lepiota	8145	22	2	179190	G
ring	7400	20	2	148000	G

grupos ou classificações), o que indica que a QuickAutoML pode apresentar resultados inferiores em comparação a ferramentas mais robustas quando os conjuntos de dados fornecidos apresentam dados nesses formatos.

Por outro lado, nos *datasets prnn_fglass*, *haberman*, *analcatdata_dmft*, *diabetes* e *hypothyroid*, a QuickAutoML obteve resultados superiores aos obtidos pelas demais ferramentas analisadas. Isso indica que os processos de pré-processamento, seleção e construção de características implementados pela TPOT e autosklearn resultam em uma *overengineering*, ou seja, acabam por não agregar valor ao modelo resultante em alguns casos.

Por fim, nos *datasets penguins*, *phoneme*, *agaricus_lepiota* e *mfeat_zernike*, a QuickAutoML gerou um modelo que obteve os mesmos resultados, em termos de acurácia, que um modelo gerado por uma das outras ferramentas. Isso indica que, mesmo com a complexidade maior inerente à autosklearn e TPOT, o resultado é equivalente e com modelos resultantes potencialmente idênticos.

É interessante destacar que apesar de a QuickAutoML ser uma ferramenta de mesma finalidade que as outras duas, suas abordagens e casos de uso ideais são diferentes. Por exemplo, por mais que autosklearn e TPOT gerem modelos melhores na maior parte dos casos, seus processos de treinamento demandam ambientes computacionais potentes e bastante tempo. Em casos de uso onde é necessário que os modelos sejam constantemente atualizados, a QuickAutoML pode ser uma solução melhor.

4. Trabalhos relacionados

Trabalhos recentes (*e.g.*, auto-sklearn [Feurer et al., 2015] e TPOT [Olson and Moore, 2016]) propõem soluções para automatizar processos de *aprendizado de máquina*, como seleção de características, processamento de dados, teste e otimização de modelos, resultando na implementação de um conceito conhecido AutoML. A utilização de AutoML reduz a demanda por cientistas de dados e permite que usuários desenvolvam soluções baseados em *aprendizado de máquina* sem conhecimento técnicos aprofundados [Zöller and Huber, 2021]. Diferentemente destas, o objetivo da QuickAutoML é fornecer bons resultados com um custo computacional significativamente menor através da redução do número de etapas de pré-processamento de dados.

Diversos trabalhos investem recursos nas etapas de avaliação e validação dos resultados obtidos pelos modelos. Ferramentas como ModelTracker [Amershi et al., 2015], Crosscheck [Arendt et al., 2020] e Gestalt [Patel et al., 2010] implementam soluções baseadas em interfaces gráficas que objetivam tornar a avaliação dos modelos mais intuitiva e eficiente. Similarmente, atráves da QuickAutoML o usuário pode gerar relatórios de desempenho dos modelos treinados em formato de gráficos ou planilhas, o que facilita as análises dos resultados.

Outras ferramentas, como Runway [Tsay et al., 2018], ModelHub [Miao et al., 2016] e ModelDB [Vartak et al., 2016], atuam no gerenciamento de modelos após as etapas de treinamento, facilitando a rastreabilidade, compreensão dos resultados, comparação entre os modelos e a criação de *pipelines* de dados. Isto é importante, considerando o cenário dinâmico onde os modelos de *aprendizado de*

Tabela 3. Resultados obtidos na comparação entre modelos

Nome	Acurácia (TPOT)	Acurácia (AutoSk)	Acurácia (Quick)	Tempo (TPOT)	Tempo (AutoSk)	Tempo (Quick)	Diferença máxima em acurácia
GAMETES_Epistasis_3_Way_20atts_0.2H_EDM_1_1	0.555	0.672	0.5	89.348	115.115	24.382	0.172
GAMETES_Epistasis_2_Way_20atts_0.1H_EDM_1_1	0.667	0.652	0.535	179.472	115.228	24.52	0.132
GAMETES_Epistasis_2_Way_20atts_0.4H_EDM_1_1	0.77	0.787	0.677	196.705	115.282	23.385	0.11
sonar	0.885	0.827	0.808	56.204	114.741	28.002	0.077
soybean	0.905	0.917	0.876	158.066	114.325	22.618	0.041
prnn_crabs	1.0	1.0	0.96	28.821	117.824	22.388	0.04
horse_colic	0.815	0.815	0.783	22.921	114.84	17.937	0.032
mux6	1.0	1.0	0.969	39.104	116.98	18.725	0.031
tae	0.553	0.526	0.526	32.351	115.83	21.234	0.027
iris	1.0	0.947	0.974	33.693	113.843	19.336	0.026
wine_quality_white	0.677	0.639	0.655	853.296	115.016	59.621	0.022
spect	0.851	0.806	0.836	41.528	116.104	11.181	0.015
kr_vs_kp	0.994	0.981	0.981	235.377	114.709	59.966	0.013
german	0.732	0.724	0.72	123.014	117.293	20.538	0.012
ring	0.977	0.978	0.966	1005.92	114.088	276.965	0.012
churn	0.958	0.954	0.948	596.834	117.527	118.372	0.01
breast	0.971	0.971	0.966	43.887	114.321	20.837	0.005
led7	0.725	0.728	0.724	121.48	114.484	23.97	0.004
allbp	0.965	0.971	0.97	425.677	115.514	78.726	0.001
mfeat_zernike	0.832	0.826	0.832	399.574	115.067	112.599	0.0
agaricus_lepiota	1.0	1.0	1.0	383.928	119.05	133.675	0.0
phoneme	0.905	0.896	0.905	285.346	114.952	66.067	0.0
penguins	1.0	1.0	1.0	53.938	114.467	18.766	0.0
hypothyroid	0.981	0.977	0.982	214.244	115.95	36.195	-0.001
diabetes	0.745	0.75	0.766	35.461	116.433	16.83	-0.016
analcatdata_dmft	0.175	0.195	0.225	38.6	116.007	13.349	-0.03
haberman	0.727	0.74	0.792	33.483	115.656	14.708	-0.052
prnn_fglass	0.731	0.731	0.808	34.827	115.128	22.433	-0.077

máquina podem se tornar ineficientes à medida que os dados ou requisitos mudam [Paleyes et al., 2020]. Futuramente, a QuickAutoML fornecerá suporte para o gerenciamento dos modelos criados, isto é, dará um maior suporte à manutenibilidade e a rastreabilidade destes ao decorrer do tempo.

Na Tabela 4 resumimos informações sobre ferramentas para automatização do processo de treinamento, avaliação e manutenção de modelos de *aprendizado de máquina*. As ferramentas foram agrupadas em três categorias, de acordo com a finalidade de sua utilização: AutoML, gerenciamento de modelos e avaliação de modelos.

Tabela 4. Principais características de trabalhos relacionados

Nome	Open Source	Ano	Dados estruturados (e.g. arquivos .csv)	Dados não estruturados (e.g. imagens, texto)	Supervisionado	Não supervisionado	Categoria
QuickAutoML	Sim	2021	Sim	Não	Sim	Não	AutoML
Auto-sklearn	Sim	2015	Sim	Não	Sim	Não	
TPOT	Sim	2016	Sim	Não	Sim	Não	
Runway	Não	2018	Sim	Não	Sim	Não	Gerenciamento de modelos
TFX	Sim	2017	Sim	Sim	Sim	Não	
Gestalt	Não	2010	Sim	Sim	Sim	Não	
ModelDB	Sim	2016	Sim	Sim	Sim	Não	
ModelHub	Sim	2016	Sim	Sim	Sim	Não	
Foolbox	Sim	2017	Sim	Sim	Sim	Não	Avaliação de modelos
DroidAPIMiner	Sim	2017	Não	Sim	Sim	Não	
ModelTracker	Não	2015	Sim	Sim	Sim	Não	
CrossCheck	Sim	2020	Sim	Não	Sim	Não	

5. Considerações Finais

Neste trabalho, apresentamos a QuickAutoML, uma ferramenta para automatização do processo de criação e treinamento de modelos de aprendizado de máquina. Como avaliação dos resultados, comparamos os modelos gerados pela ferramenta com duas ferramentas populares de *AutoML*, em 28 conjuntos de dados de avaliação. Esse experimento demonstrou que a QuickAutoML consegue encontrar modelos satisfatórios para determinado problema sem necessidade de ajustes manuais.

Para dar continuidade ao desenvolvimento da ferramenta, podemos elencar os seguintes trabalhos futuros:

1. fornecer suporte a outras abordagens de aprendizado de máquina, como métodos de regressão e *deep learning*;
2. implementar etapas de limpeza de dados, criação, transformação e seleção de *features*;
3. fornecer as funcionalidades através de uma interface gráfica de usuário.

Agradecimentos

Esta pesquisa foi parcialmente 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. O presente trabalho foi realizado também

com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Referências

Amershi, S., Chickering, M., Drucker, S. M., Lee, B., Simard, P., and Suh, J. (2015). Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 337–346.

Arendt, D., Huang, Z., Shrestha, P., Ayton, E., Glenski, M., and Volkova, S. (2020). Crosscheck: Rapid, reproducible, and interpretable model evaluation. *arXiv preprint arXiv:2004.07993*.

Botchkarev, A. (2018). Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. *arXiv preprint arXiv:1809.03006*.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Miao, H., Li, A., Davis, L. S., and Deshpande, A. (2016). Modelhub: Towards unified data and lifecycle management for deep learning. *arXiv preprint arXiv:1611.06224*.

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*, pages 66–74. PMLR.

Paleyes, A., Urma, R.-G., and Lawrence, N. D. (2020). Challenges in deploying machine learning: a survey of case studies. *arXiv preprint arXiv:2011.09926*.

Patel, K., Bancroft, N., Drucker, S. M., Fogarty, J., Ko, A. J., and Landay, J. (2010). Gestalt: integrated support for implementation and analysis in machine learning. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 37–46.

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.

Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13.

Siqueira, G., Rodrigues, G., Kreutz, D., and Feitosa, E. (2021). Quickautoml: Uma ferramenta para treinamento automatizado de modelos de aprendizado de máquina. In *XV Workshop de Trabalhos de Iniciação Científica e de Graduação (WTICG)*.

Tsay, J., Mummert, T., Bobroff, N., Braz, A., Westerink, P., and Hirzel, M. (2018). Runway: machine learning model experiment management tool. In *Conference on Systems and Machine Learning (SysML)*.

Vartak, M., Subramanyam, H., Lee, W.-E., Viswanathan, S., Husnoo, S., Madden, S., and Zaharia, M. (2016). Modeldb: a system for machine learning model management. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*, pages 1–3.

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., and Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40.

Yao, X. and Liu, Y. (2014). Machine learning. In *Search Methodologies*, pages 477–517. Springer.

Zöller, M.-A. and Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. *Journal of Artificial Intelligence Research*, 70:409–472.