Universidade Federal do Paraná
Setor de Ciências Exatas
Departamento de Estatística
Programa de Especialização em *Data Science* e *Big Data*

Luiz Gabriel de Souza

Aprendizado de Máquina Automático (AutoML) para problemas de classificação: Análise comparativa do desempenho de frameworks privados e públicos

Curitiba
2022

Luiz Gabriel de Souza

Aprendizado de Máquina Automático (AutoML) para problemas de classificação: Análise comparativa do desempenho de frameworks privados e públicos

Monografia apresentada ao Programa de Especialização em Data Science e Big Data da Universidade Federal do Paraná como requisito parcial para a obtenção do grau de especialista.

Orientador: Ph.D. Luiz Eduardo S. Oliveira

Curitiba
2022

Especialização em Data Science e Big Data
Universidade Federal do Paraná
dsbd.leg.ufpr.br

Aprendizado de Máquina Automático (AutoML) para problemas de classificação: Análise comparativa do desempenho de frameworks privados e públicos

Luiz Gabriel de Souza[1]
Ph.D. Luiz Eduardo S. Oliveira[2]

Resumo

O crescente uso de soluções baseadas em aprendizado de máquina para resolução de problemas complexos em diversas áreas de aplicação, tem demandado soluções versáteis e de fácil utilização, que não demandem muitas habilidades técnincas para a sua execução. A partir de meados da década passada, começaram a surgir soluções voltadas para esse objetivo, denominadas *Automated Machine Learning* (AutoML), com iniciativas da comunidade de desevolvimento de códigos abertos e de grandes empresas privadas. Com base nessa realidade, diversos projetos tem buscado comparar a performance de *frameworks* de AutoML, testando diferentes bases de dados, arquiteturas e parametrizações aceitas por cada *framework*. Este projeto tem como objetivo comparar a performance de seis *frameworks* de *AutoML* escolhidos, sendo três deles pagos (GCP AutoML, Amazon Web Services Autopilot e Dataiku) e três de código aberto (Pycaret, MLJar e H20), para modelos de classificação binários, visando entender se há diferenças consideráveis entre a parformance dos *frameworks* pagos e gratuitos de AutoML com base na métrica ROC-AUC. O projeto fez uso de 10 bases de dados públicas, que foram processadas com o menor ajuste possíveis de parametrização permitidos por cada *framework*, buscando emular o uso de usuários com pouca ou nenhuma experiência em machine learning. Concluimos que o *framewor* GCP AutoML se destacou mas ao contrário dos demais *framework* não mostra quais algoritmos foram testados, seus parâmtros ou qualquer outra informação referente ao log dos experimentos, mostrando apenas as métricas de performance da solução final. Dentre os *framework* gratuitos o MLJar e H2O demonstraram alto nível de competitividade frente aos outros *framework* privados AWS Autopilot e Dataiku, tendo apenas como contra ponto a necessidade de conhecimentos de programação para execução dos processos enquanto as soluções privadas possuem interface interativo e não necessitam de códificação explicita.

Palavras-chave: Automate Machine Learning, AutoML, Pycaret, GCP, Autopilot.

Abstract

The growing use of solutions based on machine learning to solve complex problems in several application areas has demanded versatile and easy-to-use solutions that do not require many technical skills for their execution. From the middle of the last decade, solutions aimed at this objective began to emerge, called Automated Machine Learning (AutoML), with initiatives from the open source development community and large private companies. Based on this reality, several projects have sought to compare the performance of AutoML frameworks, testing different databases, architectures and parameterizations accepted by each framework. This project aims to compare the performance of six chosen frameworks of AutoML, three of which are paid (GCP AutoML, Amazon Web Services Autopilot and Dataiku) and three are open source (Pycaret, MLJar and H20) , for binary classification models, in order to understand if there are considerable differences between the performance of paid and free AutoML frameworks based on the ROC-AUC metric. The project made use of 10 public databases, which were processed with the smallest possible parameterization adjustment allowed by each framework, seeking to emulate the use of users with little or no experience in machine learning. We conclude that the framewor GCP AutoML stood out but unlike the other framework it does not show which algorithms were tested, their parameters or any other information regarding the log of the experiments, showing only the performance metrics of the final solution . Among the free framework, MLJar and H2O demonstrated a high level of competitiveness against other private framework AWS Autopilot and Dataiku, having only as a counterpoint the need for programming knowledge to execute the processes while the private solutions have an interactive interface and do not require explicit coding.

Keywrods: *Automate Machine Learning, AutoML, Pycaret, GCP, Autopilot.*

[1]Aluno do programa de Especialização em Data Science & Big Data, luiz.gabriel.souza@hotmail.com.

[2]Professor do Departamento de Estatística - DEST/UFPR, luiz.oliveira@ufpr.br.

1 Introdução e objetivo do projeto

As últimas décadas marcaram o crescimento do uso de soluções de aprendizado de máquina por grandes empresas e governos no mundo todo. Foram criadas soluções robustas para resolver problemas de visão computacional, reconhecimento de fala, jogos, classificação de elementos, agrupamentos, entre outras. É possível encontrar aplicações de ML na indústria com máquinas interconectadas que prevem problemas de produção [1]; em hospitais auxiliando no diagnósticos de paciêntes [2]; nas escolas e universidades personalizando o ensino de estudos [3]; no marketing das empresas, personalizando e maximizando os resultados de campanhas de comunicações [4]; etc.

Com essa popularização, o interesse de profissionais que buscam os benefícios dessa tecnologia também cresceu, fazendo com que a presença e o valor de soluções que exijam pouco ou nenhum conhecimento prévio sobre ML tenha se expandido. Empresas privadas e a comunidade de desenvolvedores de código aberto passou a criar soluções que automatizam o processo de experimentação que é feita normalmente por experiêntes profissionais das áreas de estatística e matemática. Grandes empresas de tecnologia como Amazon e Google, por exemplo, criaram soluções deste caráter, onde com poucos cliques em uma plataforma baseada em nuvem interativa, é possível treinar modelos complexos e extrair resultados que podem ser utilizados imediatamente, sem a necessidades de configuração prévia de todo o fluxo tradicional de construção de um modelo de ML. Ao mesmo tempo, iniciativas de código aberto criaram ferramentas como Pycaret [5] e MLJar [6], que apesar de não terem interface interativa como as citadas pelas empresas privadas acima, conseguem entregar soluções complexas e realizar diversos tipos de experimentos e transformação de dados com poucas linhas de código, de maneira intuitiva e rápida, sendo acessível para novos usuários interessados em modelos de ML.

Além de atendar aos interesses de profissionais com pouca experiência no desenvolvimento de soluções de ML, *frameworks* de *AutoML* podem contribuir facilidando o dia a dia de profissionais experientes, automatizando processos de experimentação e otimização de parâmetros, economizando tempo e aumentando a produtividade dos especialistas na área:

> "Put simply, AutoML can lead to improved performance while saving substantial amounts of time and money, as machine learning experts are both hard to find and expensive. As a result, commercial interest in AutoML has grown dramatically in recent years, and several major tech companies are now developing their own AutoML systems."[7]

Sendo assim, com base no crescimento do uso de soluções de AutoML, tanto privadas quanto gratuitas, este projeto tem como objetivo comparar a performance de *frameworks* de *AutoML* para problemas de classificação binários. Não serão realizados nenhum tipo de pré processamento ou engenharia de caracteríticas dos dados, visando testar as funcionalidades em cada *framework*. Além disso, tentaremos emular o uso dos *frameworks* por usuários com pouca experiência em ML, fazendo uso do maior uso de parâmetros padrão de cada framework

2 Dados

A fim de a garantir comparabilidade entre os resultados dos testes e ao mesmo tempo considerar as dificuldades de se lidar com dados categóricos e numéricos, selecionamos dez *datasets* disponibilizadas pelo projeto aberto OpenML [8], utilizadas para a criação do *AutoML Benchmark Framework* [9], idealizado pelo mesmo. Na documentação do projeto [9], a escolha dos *datasets* foi descrita como:

> "The benchmark aims to consist of datasets that represent real-world data science problems. This means we want to include datasets of all sizes (including big ones), of different problem domains and with various levels of difficulty".[9]

Dentre todas os *datasets* disponíveis, foram escolhidos os que possuiam diferentes volumes de atributos e instancias entre si, bem como diferentes tipos de dados como apenas categóricos, apenas numérico e numéricos e categóricos ao mesmo tempo. O nome e algumas características dos *datasets* escolhidos pode ser observado na tabela 1:

Tabela 1: Data-sets selecionados

Nome	Atributos	Instancias	Tipo_Atributos
kdd	231	50.000	Mix
jasmine	145	2.984	Mix
nomao	119	34.465	Mix
kr_vs_kp	37	3.196	Categórico
higgs	29	98.050	Numérico
kc1	22	2.109	Numérico
bank_marketing	17	45.211	Mix
adult	15	48.842	Mix
amazon_emp	10	32.769	Categórico
blood_transfusion	5	748.000	Numérico

Todos os *datasets* podem ser encontrado neste link

3 Experimentos

Foram testados seis *frameworks* de autoML, três deles privados e outros três de código aberto. Entre os privados, foram testados o Google Cloud AutoML (GCP AutoML) [10], Amazon Web Services AutoPilot (AWS AutoPilot) [11] e Dataiku [12]. Entre os de código aberto, foram testados PyCaret [5], MLJar [6] e H2O [13].

Para garantir a comparabilidade dos experimentos, algumas premissas foram tomadas em conta para todos os modelos processados:

- **Tempo de Processamento:** Limitado a 60 minutos o tempo de processamento para o treino de cada *datasets* em cada *framework*.
- **Ambiente de Processamento:** Para os *frameworks* privados, foram utilizados seus respectivos ambientes de processamento em nuvem sem personalização. Para os de código aberto, utilizamos a plataforma *Google Colab* com suas configuações padrões do pacote gratúito. Os scripts podem ser acessados aqui.
- **Métrica para otimizar:** Quando permitido pelo *framework*, optamos por otimizar os experimentos para a métrica **ROC-AUC**, escolhida como medida de comparação entre os resultados obtidos de cada *framework* - a mesma utilizada pelo projeto [9] para problema de classificação binária.
- **Paramêtros opcionais:** Optamos por utilizar a maior quantidade de parâmetros padrãos de cada *framework*, ou seja, sem utilizar parâmetros como balanceamento automático de variáveis, quantidade de modelos para treino, etc. Adicionamos, quando aplicável, a semente aleatória igual a 123, buscando reproducibilidade dos experimentos.
- **Outras limitações intencionais:** Quando possíveis de serem identificados, foram desconsirados resultados de modelos *Ensembles* e Redes Neurais, visto que nem todas os *frameworks* possuem tais *features*.

3.1 *Frameworks* Privados

Com o aumento da popularidade de soluções em cloud, que geralmente são pagas, as empresas investiram também em soluções de machine learning pré prontas para uso, como é o caso do Google Cloud Plataform (GCP), Amazon Web Services (AWS) e a Dataiko. Todas essas sem a necessidade de criação de scripts ou programação explicita, tudo sendo feito via interface interativa. Abaixo alguns comentário sobre nossa experiência de usuário sobre cada uma delas:

- **AWS Sagemaker AutoPilot:**
  - Experiência do Usuário: Se demonstrou ser fácil de se interagir. É necessário a criação de uma instância virtual *Sage Maker* e um *bucket S3* para salvar a base de treino e os resultados dos modelos treinados. Uma vez criado o ambiente, basta seguir uma série de passos na interface interativa da ferramenta, escolhendo a variável alvo, o tempo de processamento a até o *endpoint* do modelo de maneira opcional.
  - Funcionalidades: Uma das principais características deste framwork versus os demais é o fato de todos os modelos testados pelo framwork serem salvos em Jupyters Notebook, garantindo a reproducibilidade dos experimentos e possível personalização por parte do usuário.
- **Google Cloud Plattaform Auto Machine Learning**
  - Experiência do Usuário: Não é necessário a criação de instâncias virtuais. Só é necessário importar o data-set e seguir o passo a passo na interface interativa do módulo *Vertex AI* da plataforma, escolhendo a variável alvo, tempo de processamento e se deseja salvar os dados de treinamento e resultados no banco de dados *BigQuery*. Como toda a criação de instâncias virtuais para processamento é feita automaticamente pela plataforma, notamos que o tempo de processamento é maior do que nas demais plataformas. Se escolhermos treinar um dataset por 1 hora, por exemplo, só obterá os resultados do processo concluido cerca de 2 horas depois, pois a ferramenta realiza outros procedimentos antes e depois do treinamento dos modelos.
  - Funcionalidades: A ferramenta é completa ao que tange problemas de machine learning, sendo possível realizar o treinamento automático de dados estruturados (dados tabulados para classificação e regressão) e não estruturados (fotos, audios e textos). Como lado negativo, o *framework* não indica qual foi o melhor algoritmo escolhido, apenas mostra as métricas do modelo escolhido, fazendo com que não seja possível reproduzir os experimentos fora da plataforma.
- **Dataiku**
  - Experiência do Usuário: Possui interface muito clara e intuitiva. Com poucos cliques é possível importar o dataset de treino e criar um experimento de auto machine learning que traz os resultados e métricas para cada modelo treinado.
  - Funcionalidades: A principal característica diferente dos demais *frameworks* privados é o fato de ser possível escolher quais algoritmos serão testados. Apesar disso, não é possível escolher o tempo de treinamento, o que limita a quantidade de combinações experimentos com os hiperparametros a serem otimizados.

3.2 *Frameworks* Gratuitos

Dentro os *frameworks* gratuitos, um ponto que os difere dos *frameworks* privados, é o fato de precisarem de configuração via código, visto que são bibliotecas públicas em python, o que pode se tornar um empecilho para leigos em programação fazerem uso de tais soluções. Abaixo apresento alguns comentários sobre nossa esperiência de uso dos *framework* gratuitos:

- **Pycaret**
  - Experiência do Usuário: Se demonstrou uma biblioteca de fácil utilização, com muitos recursos ajustáveis e fácil acesso aos objetos criados durante os testes como as bases transformadas, métricas, entre outros. Um diferencial importante deste *framework* é que ele é construido com base nos modelos do *Sklearn*, que é o *framework* de ML mais utilizado por ciêntistas de dados. No entanto, não se limita aos algoritmos dispostos no *Sklearn*, pois possui compatibilidade para a adição de outros modelos, inclusive Redes Neurais, e suporte para processamento com GPUs.
  - Funcionalidades: Com poucos parâmetros nas funções é possível criar experimentos completos, testando uma série de algoritmos de machine learning e com resultados claros e comparativos. Para os usuários mais experientes, existem uma ampla gama de parâmetros adicionais não obrigatórios que podem automatizar transformações de dados. Além disso, possui funções de visualização de dados pré construidas que facilitam muito a análise dos resultados.
- **MLJar**
  - Experiência do Usuário: É necessário realizar a separação da base de treino e teste antes de passar os dados para o algoritmo. Uma vez realizada esse procedimento, com apenas uma função é possível dar inicio ao processamento dos experimentos.
  - Funcionalidades: Um diferencial deste *framework* é a geração de um arquivo HTML que consolida os resultados de todos os modelos testados. Com base nisso, é possível análisar as métricas, hiperparametros e gráficos dos resultados de cada modelo testado, incluindo o melhor entre eles.
- **H20 (Versão Gratúita)**
  - Experiência do Usuário: Dentre os *frameworks* gratuitos, apesar de ser o mais utilizado pela comunidade atualmente, foi o que mais apresentou a necessidade de configurações iniciais e maior conhecimento de programação por parte do usuário. Alguns exemplos dessas configurações é o fato do *framework* possuir estrutura de dados própria e necessitar de um módulo de java para funcionar.
  - Funcionalidades: Por ser o *framework* mais antigo dentre todos os testados, possui funcionalidades avançadas a possibilidade de criação de experimentos que incluem *deep learning* e complexos *ensembles* de modelos.

Na tabela 2 é possível observar algumas características comparáveis entre os *frameworks*.

4 Resultados e discussão

Na figura 1 é possível visualizar os resultados comparativos da métrica ROC-AUC para cada um dos *datasets* em cada *framework*. Vemos que há uma certa sobreposição de valores, dificultando a comparação da performance entre os *frameworks*. Isso acontece pois alguns *datasets* apresentam resultados semelhantes quando submetidos a cada *framework*, como por exemplo nos *datasets* jasmine, kr_vs_kp e nomao. Mesmo assim, em alguns *datasets* é possível observar que alguns *frameworks* se destacam, como é o caso dos *datasets* kdd, kc1, bank_marketing, amazon_emp e adult, onde é possível observar que o *framework* GCP Auto ML desponta dos demais, trazendo os melhores resultados. Por outro lado, é possível observar que o *framework* Pycaret tem os piores resultados nos *frameworks* kc1 e amazon_emp.

No entanto existe uma ressalva muito importante a ser feita sobre o *framework* GCP AutoML. Diferente de todos os demais *frameworks* analisados, ele é o único que não indica qual foi o algoritimo escolhido como melhor, se tornando uma espécie de caixa preta que não mostra sua metodologia, podendo fazer uso de quaisquer soluções não presentes em alguns outros *frameworks* como o uso de ensembles ou deep learning. Este fato é particularmente importante pois nos demais *frameworks* notamos uma tendência na escolha de modelos baseados em árvores (LightGBM, XGBoost, Random Forest e GBM), apesar de outros modelos lineares também terem sido usados em um numéro menor de casos, como SVM, LDA e GLM. Uma observação interessante é o fato de os datasets exclusivamente numéricos tiverem uma predominância do uso do modelo Random Forest enquanto que os demais tipo de *datasets* (Mix e Categóricos) tiveram predominância dos algoritimos LightGBM e XGBoost.

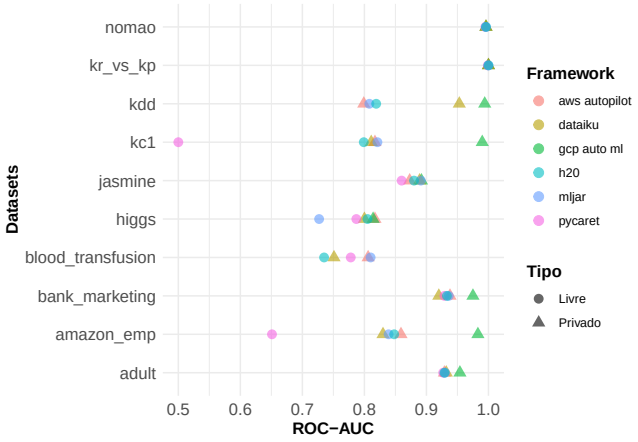

Figura 1: ROC-AUC Classificação Binária

Visando desenvolver uma métrica geral sobre performance de cada *framework*, criamos um ranking ordenado pela ROC-AUC de cada *dataset*, que vai de 1 a 6, onde o *framework* que obteve a maior ROC-AUC é o número 6 e o pior dataset (o último colocado) recebe o número 1.

Tabela 2: Resumo das Características dos Frameworks

Framework	aws.autopilot	gcp.auto.ml	dataiku	pycaret	mljar	h20
Pago ou Livre?	Pago	Pago	Pago	Livre	Livre	Livre
Realiza transformação de dados	S	S	S	S	S	N
Precisa de dados balanceados	N	N	N	N	N	N
Seleciona tarefa automaticamente (Reg vs Clas)	S	N	N	N	S	N
Permite escolher tempo de processamento	S	S	N	S	S	S
Experimentos reprodutiveis	N	N	N	S	S	S
Produz Ensembles	N	-	N	S	S	S
Produz Redes Neurais	N	-	S	N*	S	S
Forks	-	-	-	1300	271	1900
Versão	-	-	-	2.3.10	0.11.2	3.36.1.1

* Permite a adição manual de algoritmos de redes neurais externos (e qualquer outro tipo de modelo) desde que seja um objeto compatível com o Sklearn API

Em caso de empate, ambos recebem a mesma pontuação.

Quando somamos o ranking por *framework*, obtemos uma espécie de pontução comparativa entre os *frameworks*, conforme observado na figura 2. Vemos que, na comaparação geral, o *framework* GCP Auto ML possui o maior quantidade de pontos, seguido do AWS Autopilot, MLJar, Dataiku, H2O e por último o Pycaret. Concluimos então que os *frameworks* privados possuem melhor performance geral, visto que no top 3 melhores *frameworks* temos em primeiro e em segundo lugar *frameworks* privados. Apesar disso, concluimos também que os *frameworks* gratuitos MLJar e H20 não ficam muito distantes da performance dos *frameworks* privados AWS Autopilot e Dataiku, o que indica um auto grau de competitividade dos *frameworks* gratuitos, a depender do problema a ser resolvido.

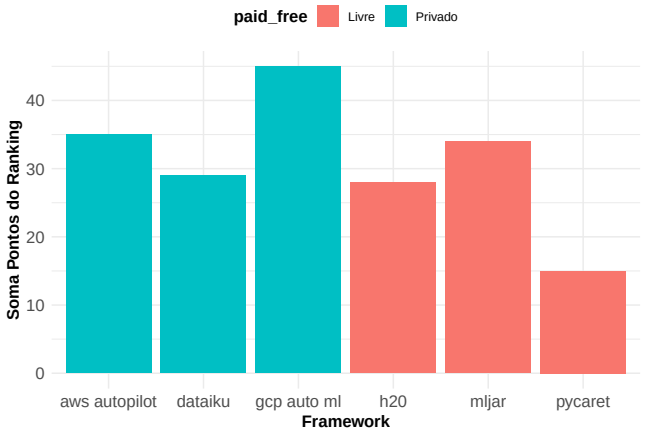

Figura 2: Soma de pontos por Framework

Se desconsiderarmos a dimensão *framework* e visualizarmos apenas a relação ranking versus gratuito ou privado, temos o gráfico 3. Nele observamos que no geral, os *frameworks* privados tiveram performance melhor do que os gratuitos, muito influênciado pela performance do *framework* GCP AutoML, como vimos no gráfico 2.

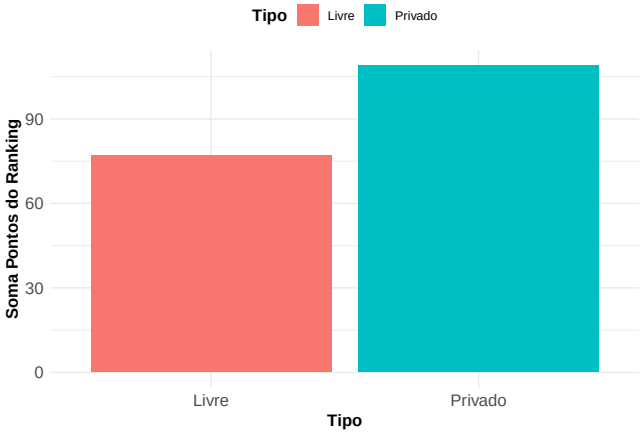

Figura 3: Soma de pontos por tipo de Framework

5 Conclusão

O presente projeto conclui que há apenas um ligeiro ganho de performance dos *frameworks* privados frente aos gratuitos, com exceção do *framework* GCP Auto ML que desponta dos demais. Isso nos diz que a depender do seu problema de negócio a ser resolvido, de suas limitações de orçamento e dos seus conhecimentos em programação, soluções gratuitas podem ser bastante competitivas contra soluções privadas. Dentre os *frameworks* gratuitos, é notável a performance geral do MLJar, muito próximo da performance do *framework* privado AWS Autopilot.

Uma ressalva importante sobre o presente projeto é que não foi escopo do projeto analisar a presença de overfitting ou qualquer outro problema de generalização que os resultados possam ter.

Como próximos passos acredito ser relevante estruturar uma maneira efetiva e comparável de coletar dados de custos financeiros de processamento em ambiente virtual em nuvem para os *frameworks* privados. Além disso, é possível extender a potência máxima de cada *framework*, incluindo experimentos com *deep learning* e *ensembles* de modelos, quando o *framework* possuir tal funcionalidade. Por fim, para se obter conclusões mais abrangentes, seria importante aumentar o número de

datasets e o escopo para problemas de regressão e classificações multi-classes.

Referências

[1] Marina Paolanti, Luca Romeo, Andrea Felicetti, Adriano Mancini, Emanuele Frontoni, and Jelena Loncarski. Machine learning approach for predictive maintenance in industry 4.0. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6, 2018.

[2] Niharika G. Maity and Sreerupa Das. Machine learning for improved diagnosis and prognosis in healthcare. In *2017 IEEE Aerospace Conference*, pages 1–9, 2017.

[3] Lijia Chen, Pingping Chen, and Zhijian Lin. Artificial intelligence in education: A review. *IEEE Access*, 8:75264–75278, 2020.

[4] Pål Sundsøy, Johannes Bjelland, Asif M. Iqbal, Alex "Sandy" Pentland, and Yves-Alexandre de Montjoye. Big data-driven marketing: How machine learning outperforms marketers' gut-feeling. In William G. Kennedy, Nitin Agarwal, and Shanchieh Jay Yang, editors, *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 367–374, Cham, 2014. Springer International Publishing.

[5] Moez Ali. *PyCaret: An open source, low-code machine learning library in Python*, April 2020. PyCaret version 1.0.0.

[6] Aleksandra Płońska and Piotr Płoński. Mljar: State-of-the-art automated machine learning framework for tabular data. version 0.10.3, 2021.

[7] Joaquin Vanschoren Frank Hutter, Lars Kotthoff. *Automated Machine Learning*. 2019.

[8] Bernd Bischl Joaquin Vanschoren, Jan N van Rijn. OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[9] P. Gijsbers, E. LeDell, S. Poirier, J. Thomas, B. Bischl, and J. Vanschoren. An Open Source AutoML Benchmark. *arXiv preprint arXiv:1907.00909 [cs.LG]*, 2019. Accepted at AutoML Workshop at ICML 2019.

[10] Google Cloud. *Vertex AI para usu?rios do AutoML*.

[11] Amazon Web Services. *Amazon SageMaker Developer Guide*.

[12] Dataiku. *Dataiku Data Science Studio*.

[13] Erin LeDell and Sebastien Poirier. H2O AutoML: Scalable automatic machine learning. *7th ICML Workshop on Automated Machine Learning (AutoML)*, July 2020.