Democratising Policy Analytics with AutoML

Danilo Freire[[1]](#footnote-0)

# 1 Introduction

Machine learning has made steady inroads into the social sciences. Although causal inference designs have become the standard methodology in economics and political science (Angrist and Pischke 2008), machine learning is increasingly used to tackle “prediction policy problems”, in which high forecasting accuracy is more important than unbiased regression coefficients (Kleinberg et al. 2015). For instance, scholars have employed algorithmic modelling to predict civil wars (Muchlinski et al. 2016; Ward, Greenhill, and Bakke 2010), mass killings (Freire and Uzonyi 2018; Ulfelder 2013), and state repression (Hill Jr and Jones 2014). Supervised machine learning also helps governments to devise local public policies, such as allocating fire inspection teams or directing patients for medical treatment (Athey 2017). Therefore, computer algorithms can improve social welfare by making state interventions more effective.

Despite the popularity of predictive analytics, building machine learning models remains a labour-intensive task. Practitioners apply several preprocessing steps just to prepare their data, and many modelling decisions, such as algorithm selection or parameter optimisation, are still largely based on trial and error (Elshawi, Maher, and Sakr 2019). As a result, areas that could benefit from predictive algorithms do not reach their full potential due to implementation challenges or lack of technical expertise (Amershi et al. 2019; Truong et al. 2019; Yang et al. 2018). In this regard, methods that simplify the machine learning pipeline can have significant academic and policy impacts (Ahmed, Mula, and Dhavala 2020; Healy, McInnes, and Weir 2017).

Automated machine learning (AutoML) aims to fill this gap. AutoML is an emerging framework that automatically chooses and optimises machine learning algorithms. More specifically, AutoML provides data-driven tools to minimise human effort in the machine learning workflow, automating steps like feature engineering, model selection, hyperparameter tuning, and model interpretation (Elshawi, Maher, and Sakr 2019). AutoML not only frees machine learning specialists from tedious and error-prone tasks, but makes state-of-the-art algorithms accessible to regular users. According to their proponents, AutoML promotes a true democratisation of artificial intelligence (Hutter, Kotthoff, and Vanschoren 2019, ix; Shang et al. 2019). AutoML approaches have also been very successful in prediction challenges, and they consistently reach the top 5% in public machine learning competitions (AWS Open Source Blog 2020; Google AI Blog 2020).

In this paper, I introduce three Python AutoML algorithms that policy analysts may consider in their work. In the following section, I describe the main functionalities of AutoKeras (Jin, Song, and Hu 2019), H2O AutoML (H2O.ai 2017), and TPOT (Olson and Moore 2016). All of the algorithms are open source, actively maintained, and easy to use. Then, I replicate two analyses that employ expert-coded machine learning models and show that AutoML can achieve comparable or better predictive performance with only a few lines of code. Lastly, I discuss how users can make their AutoML scalable and reproducible with Docker containers. Docker allows researchers to create an image of their complete working environment, thus all AutoML specifications and dependencies are automatically embedded in the Docker file. While Docker has been widely employed in business applications, its use in academia remains limited. I provide a simple tutorial so that readers can upload their AutoML setup to a website and share their Docker containers with co-authors and referees.

# 2 A Brief Introduction to AutoML Algorithms

Automated algorithms are a recent addition to the machine learning field. Thornton et al. (2013) proposed the first method to jointly address the problems of algorithm selection and parameter optimisation, and their results show that automated solutions often outperform baseline models. Since then, the literature has grown significantly. Today, there are a multitude of AutoML algorithms available for non-expert users, which are not only able to predict numeric data, but also to classify objects, translate text, annotate videos, and perform sentiment analysis in social media with few instructions (Liu et al. 2020).

The intuition behind AutoML algorithms is simple. First, the algorithm splits the original data into training and test datasets and applies different models to the training partition. Then the algorithm selects the model which achieves the best performance in a given evaluation metric, such as the mean squared error or classification accuracy. Having selected the algorithm that minimises the chosen metric, the next step is to find the set of hyperparameters that further improves the model’s predictive ability. The selection method here is the same. The algorithm tests many combinations of parameters and chooses the one that produces the best results according to the estimation metric. Finally, the results are compared against the test dataset to see how the model performs with new data. If necessary, users can add their own configurations to the AutoML algorithm or test the machine learning pipeline with other data splits.

Many AutoML libraries also perform feature engineering tasks without human intervention. Feature engineering is the process of recoding variables to improve the performance of machine learning algorithms. Common tasks include creating dummy variables from categorical indicators, filling missing data, standardising numeric covariates, or removing correlated features to avoid multicollinearity (He, Zhao, and Chu 2020; Truong et al. 2019). AutoML takes a data-driven approach here too, and selects those data transformations that improve forecasting scores the most.

## 2.1AutoKeras

AutoKeras is an AutoML algorithm based on Keras (Chollet 2015), an interface for Google’s TensorFlow machine learning platform (Abadi et al. 2015). AutoKeras focuses exclusively on deep neural networks, and it performs classification and regression tasks on images, texts, and tabular data. Neural networks require extensive tuning to increase prediction accuracy, but AutoKeras uses neural architectural search (NAS) to automatically optimise the network hyperparameters. In this sense, users can train complex deep learning algorithms with little to no machine learning experience. One only needs to write four lines of code to run a classification task in AutoKeras:

import autokeras as ak *# load library*  
  
model **=** ak.StructuredDataClassifier() *# build model for tabular data*  
model.fit(X\_train, y\_train) *# fit model with training data split*  
predictions **=** model.predict(X\_test) *# predictions*

In the example above, is the set of predictors and is the response variable. Scholars just need to split the dataset into training and test partitions and separate the independent from the dependent variables. After that, AutoKeras will estimate a series of neural networks to predict . Users can also pass many parameters to the ak.StructuredDataClassifier() function, including the metric they want to minimise or maximise, such as accuracy or area under the ROC curve, set the number of networks the model will create, and limit the time reserved for each task. Please refer to <https://autokeras.com> to know more about AutoKeras’s model parameters and how to use the software for image or text classification and regression.

## 2.2 H2O AutoML

The second algorithm I discuss here is H2O AutoML. Developed by H2O.ai, a company based in Silicon Valley, H2OAutoML is a free and open source automated machine learning solution. Thus, individuals and firms can use it at no cost, and they can also inspect and modify the original code if they want to. Another advantage of H2O AutoML is that it provides a graphic interface that helps beginners to get started with the platform. H2O.ai offers their AutoML software for both R and Python, and the packages use the same functions and arguments in the two languages. Users only need to specify the dependent and independent variables, the training and test datasets, and the prediction task they want to run. The algorithm will automatically find the model that best fits the training data, evaluate its performance on the test dataset, and report model statistics. Example code for binary classification tasks in Python follows below:

import h2o *# load library*  
from h2o.automl import H2OAutoML *# load AutoML functions*  
h2o.init() *# start the module*  
  
train **=** h2o.import\_file("path/to/training\_data") *# load training data*  
test **=** h2o.import\_file("path/to/test\_data") *# load test data*  
  
x **=** train.columns *# independent variables*  
y **=** "dependent\_variable\_name" *# dependent variable*  
x.remove(y) *# remove dependent variable from matrix*  
  
model **=** H2OAutoML(max\_models**=**30, seed**=**1234) *# run 30 machine learning models*  
model.train(x**=**x, y**=**y, training\_frame**=**train) *# estimate model*  
predictions **=** model.predict(test) *# get predictions*

H2O AutoML also provides a large collection of model explainability functions. Critics have pointed out that many machine learning methods are “black boxes”, in the sense that they display little information about the estimation stage (Molnar 2020). This has serious consequences in fields where decision mechanisms are relevant *per se*, like judicial sentences or health care allocation. H2O AutoML addresses this issue by offering explanation methods that describe how the general model performs and how it explains each individual observation.[[2]](#footnote-1) The algorithm also shows the forecasting importance of every predictor (Grömping 2009), SHAP values (Lundberg et al. 2020), and partial dependence plots (Friedman and Meulman 2003).

## 2.3 TPOT

The last algorithm I introduce in this section is TPOT, or *Tree-based Pipeline Optimization Tool*. It is one of oldest AutoML solutions for Python, and its authors have won several awards for their work.[[3]](#footnote-2) TPOT uses a genetic search algorithm to find the best model for a given dataset (Olson and Moore 2016). The principle borrows ideas from evolutionary biology and consists of three steps. First, the algorithm estimates a baseline model. Then, it makes small random changes to the original computations. After that, it selects those variations that achieve high prediction scores. TPOT repeats this process until it cannot increase forecasting accuracy or after reaching the maximum computation time defined by the user.

TPOT uses the scikit-learn (Pedregosa et al. 2011) Python library to estimate the models, but in contrast with the original package, it does so with minimal human input. TPOT supports GPU acceleration and has fast estimation times when compared to other tools. Users can create a classification model with the example code below:

from tpot import TPOTClassifier *# load library*  
  
model **=** TPOTClassifier() *# build model*  
model.fit(X\_train, y\_train) *# fit model*  
print(model.score(X\_test, y\_test)) *# print model evaluation*

Where is a matrix of covariates and is the response variable. TPOT has an export function that is useful for those who need to export the optimised model and deploy it in other settings. Users can also customise TPOT’s hyperparameters for classification and regression tasks. TPOT’s documentation is available at <http://epistasislab.github.io/tpot/>.

As we can see, the code shown in the three examples is almost identical, although the functions are running different processes in the background. However, AutoKeras, H2OAutoML, and TPOT can all quickly estimate regression or classification models for numeric data. Since these are the two tasks policy analysts most often do, the three algorithms presented above can be easily integrated into their machine learning workflow.

# 3 AutoML in Practice: Replication

How do AutoML models compare with expert-coded machine learning? AutoML algorithms have frequently appeared amongst the top performers in Kaggle competitions, yet they face unique challenges when tested with political or economic data. Datasets in these fields are often much smaller and have more measurement error than sales datasets, which are the standard data in machine learning tournaments. Therefore, data from the social sciences are usually hard to predict and computer algorithms may fare poorly when compared to experts.

Here I replicate two analyses that use machine learning to forecast rare events. Ward, Greenhill, and Bakke (2010) evaluate the out-of-sample predictive power of the models described in Fearon and Laitin (2003) and Collier and Hoeffler (2004), the two most widely-cited papers on the causes of civil war onset. The papers are suitable for our analysis because they describe a policy issue that is not only important, but also notably difficult to forecast. Civil war onset is a rare event, and the causal relationships amongst variables are not well-defined in the literature, so there is a good chance that many predictors are correlated or unnecessary.

In this exercise, I estimate one model per AutoML algorithm using the default configurations. Thus, the results below are a simple baseline which allows for modifications and extensions. The only data processing tasks I did were to create a training/test dataset split (75%/25%) prior to the estimation, as some libraries do not partition the data automatically, and add 5 cross-validation folds to test the models’ prediction accuracy. To save space, I did not include the code in this paper, but the replication materials are available at <https://github.com/danilofreire/mercatus-analytics-papers>.

Regarding the estimations, I use the area under the ROC curve as a score metric to make the results comparable with those by Ward, Greenhill, and Bakke (2010). I limit the running time to 10 minutes per model, so users can have a good idea of how AutoML algorithms perform within a small time window. For reproducibility, I run all models with the same seed number generated at [random.org](http://random.org) (8305).

I begin with the civil war data collected by Fearon and Laitin (2003). The data has 6402 country-year rows and 11 potential predictors of civil war onset. Ward, Greenhill, and Bakke (2010, 371) test the out-of-sample forecasting power of the model and find an area under the ROC curve of 0.738. The authors also assess the forecasting ability of Collier and Hoeffler’s (2004) main model. They have a different measurement for civil war onset, and their dataset has 688 country-years and 9 independent variables. According to Ward, Greenhill, and Bakke (2010), the area under the ROC curve in this model is 0.823. These are the two benchmarks for my AutoML models. The results follow below.

*Area under the ROC curve from AutoML learners.*

|  |  |  |
| --- | --- | --- |
| **Model** | Fearon and Laitin (2003) | Collier and Hoeffler (2004) |
| Ward, Greenhill, and Bakke (2010) | 0.738 | 0.823 |
| AutoKeras | 0.736 | 0.758 |
| H2OAutoML | **0.783** | 0.703 |
| TPOT | 0.715 | **0.825** |

Overall, the AutoML classifiers do a good job at predicting civil conflicts. All results are close to the original benchmark, and in each task one of the algorithms has a better predictive performance than the baseline model (in bold). Considering that civil conflicts are hard to predict and that the algorithms had limited modelling time, the results are an indicative of AutoML’s strong forecasting accuracy even in adverse conditions.

**4 Sharing AutoML Models with Docker**

Now that we have estimated our AutoML models, how to deploy or share them? My suggestion is to use [Docker](https://www.docker.com/) as a reproducibility tool.[[4]](#footnote-3) Docker is a virtualisation platform that allows users to build, test, and share their software in standardised packages called containers. Each container has a lightweight version of an operating system, usually Linux, and users can add any other software or folders to the base Docker image. Instead of sharing just data and code, as it is common practice in the social sciences, scholars can distribute their complete software environment to collaborators and reviewers. Thus, Docker guarantees that all computer libraries are identical to the ones in the original analysis, which ensures complete reproducibility and easy deployment to other machines.

Docker is available for all major operating systems and requires only a few commands to work. In this section, I show how researchers can create a custom Docker container within minutes. First, download Docker Desktop at <https://www.docker.com/products/docker-desktop> and install it. Docker Desktop includes all necessary files to build and run Docker containers. Second, create a free account at DockerHub (<https://hub.docker.com/signup>), which is a cloud-based repository for Docker images. After that, we are ready to use Docker.

One can create a Docker container in two ways, either by writing a Dockerfile, a configuration file with instructions on which packages to download and run in the Docker image, or by modifying an existing Docker container. I recommend the second method as it requires less coding.

I start by pulling and running a pre-built Ubuntu Linux image. Docker will start a Ubuntu session without changing any configuration in your computer. To install the container, run the following code on your terminal:

docker pull ubuntu # download the image from DockerHub  
docker run -it ubuntu *# run the image; -it to start the Docker container*

You will see a root session in your terminal:

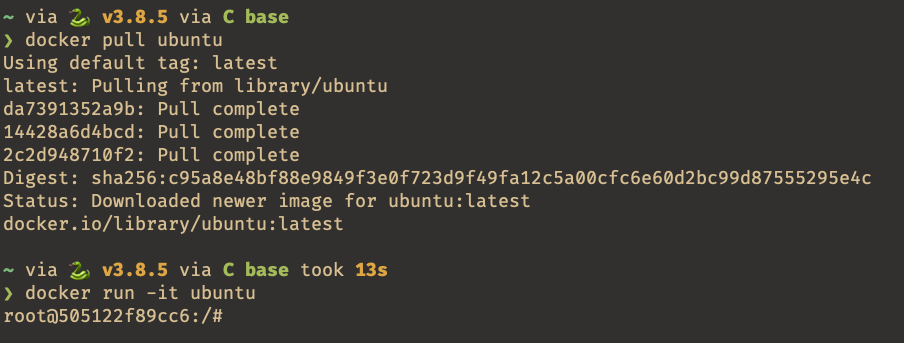


Figure 1: Docker container running Ubuntu Linux.

Then, I install Python, R, and the required AutoML libraries.

apt update -y # update the system  
apt install python3 python3-pip r-base default-jre *# required files*  
pip3 install autokeras # AutoKeras  
pip3 install h2o # H2O AutoML  
pip3 install tpot # TPOT

The pip3 command installs the Python libraries and their dependencies, so we already have all the software we need to estimate our models. The next step is to add the data and scripts to Docker. To do so, we close the connection with the container with exit and find the container ID with docker ps -a, which lists all active Docker containers. We then copy the files with the docker cp command.

*# In the Docker container:*  
exit # stop the container  
*# In your regular terminal:*  
docker ps -a *# list all available containers*

When you exit the Docker image and type docker ps -a, you will see something like:



Figure 2: List of available Docker containers.

Where the first column indicates the container ID. In this case, it starts with 5051. To copy the files to that specific container, just write the following lines in your terminal:

docker cp ~/path/to/file/automl.Rmd 5051:/automl.Rmd # copy script  
docker cp ~/path/to/file/fl\_data.csv 5051:/fl\_data.csv *# copy data*  
docker cp ~/path/to/file/ch\_data.csv 5051:/ch\_data.csv *# copy data*

Lastly, we need to save the changes we have made to the container and upload it to DockerHub. We use docker commit [container\_ID] [new\_name] to commit the changes, where [container\_ID] is ID value we found above (5051) and [new\_name] is the name we want to give to the modified container. Check if the container has been saved with docker image.

docker commit 5051 mercatus-automl  
docker images

This is the terminal output:

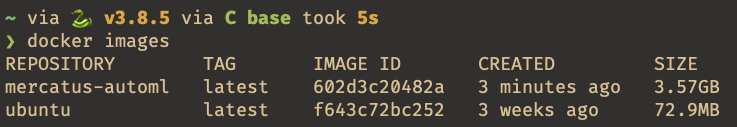


Figure 3: Docker images.

Now we need to push the image to DockerHub. Go to <https://hub.docker.com/repositories> and create a new repository. Then, add your DockerHub credentials to your local machine with docker login --username=your\_username and create a tag for your container. Note that the container ID has been updated (602d). Finally, type docker push your\_username/repository\_name to push your image to DockerHub. Example code:

docker login --username=danilofreire # add credentials  
Password: # type your password  
  
docker tag 602d danilofreire/mercatus-automl:first *# add tag*  
docker push danilofreire/mercatus-automl:first # push image to DockerHub

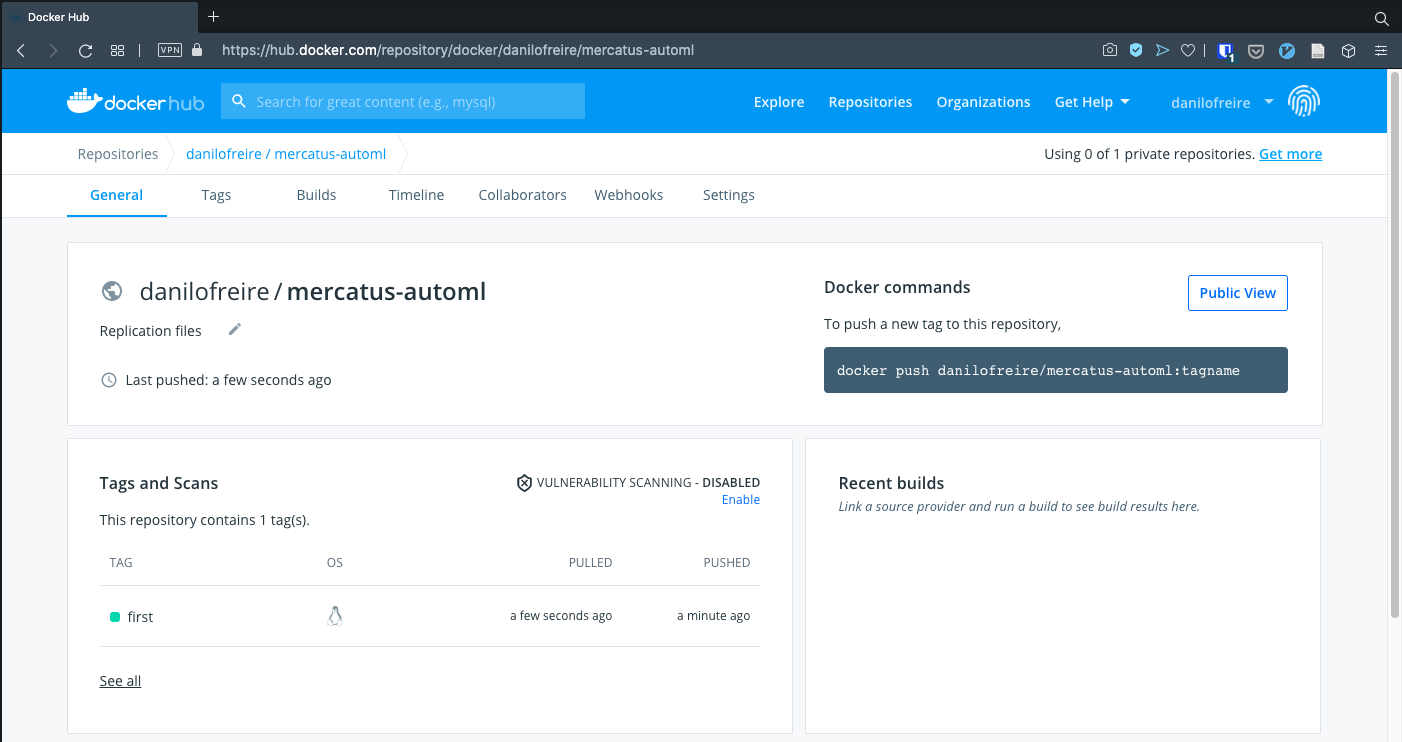


Figure 4: Container uploaded to DockerHub.

And that completes our tutorial. The image has been successfully uploaded to DockerHub and researchers can download the file with docker pull danilofreire/mercatus-automl. As we have seen, Docker offers a flexible and fully reproducible method to share machine learning models or other statistical analyses. It goes beyond current academic reproducibility practices and certifies the exact replication of the findings.

# 5 Conclusion

Computer scientists have applied machine learning to predict a myriad of outcomes, from shopping habits to kidney diseases. Algorithms now power email filters, translation software, satellite farming, self-driving cars, and many other devices. In the past years, social scientists have also adopted machine learning tools to forecast political events and improve public policies. AutoML is a new class of algorithms that facilitates machine learning tasks and allows non-experts to use sophisticated computer estimations in their work. Here I have provided a simple introduction to three Python AutoML libraries and shown that their prediction accuracy is on par with those achieved by area experts. Moreover, I have suggested that users should adopt Docker to share their machine learning models and have fully reproducible pipelines.

AutoML is a dynamic field that is still in its infancy. The growing support from big technology firms such as Google, Amazon, and Microsoft indicates that we should expect a large number of new algorithms in the future. Fortunately, most AutoML tools remain free to use, so individuals are likely to benefit from these advances. While non-experts can use AutoML tools to produce good estimates with minimal coding experience, practitioners are able to automate labour-intensive tasks and focus on improving the predictive ability of their models. In sum, I hope AutoML becomes an important part of the machine learning toolkit, and that automated models help policy analysts to answer some of their most pressing questions.

# 6 References

Abadi, Martı'n, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.” <http://tensorflow.org/>.

Ahmed, Shakkeel, Ravi S Mula, and Soma S Dhavala. 2020. “A Framework for Democratizing AI.” *arXiv Preprint arXiv:2001.00818*.

Amershi, Saleema, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. “Software Engineering for Machine Learning: A Case Study.” In *2019 Ieee/Acm 41st International Conference on Software Engineering: Software Engineering in Practice (Icse-Seip)*, 291–300. IEEE.

Angrist, Joshua D, and Jörn-Steffen Pischke. 2008. *Mostly Harmless Econometrics: An Empiricist’s Companion*. Princeton, NJ: Princeton University Press.

Athey, Susan. 2017. “Beyond Prediction: Using Big Data for Policy Problems.” *Science* 355 (6324). American Association for the Advancement of Science: 483–85.

AWS Open Source Blog. 2020. “Machine learning with AutoGluon, an open source AutoML library.” <https://aws.amazon.com/blogs/opensource/machine-learning-with-autogluon-an-open-source-automl-library/>.

Chollet, François. 2015. “Keras.” <https://keras.io>.

Collier, Paul, and Anke Hoeffler. 2004. “Greed and Grievance in Civil War.” *Oxford Economic Papers* 56 (4). Oxford University Press: 563–95.

Elshawi, Radwa, Mohamed Maher, and Sherif Sakr. 2019. “Automated Machine Learning: State-of-the-Art and Open Challenges.” *arXiv Preprint arXiv:1906.02287*.

Fearon, James, and David Laitin. 2003. “Ethnicity, Insurgency, and Civil War.” *American Political Science Review* 97 (1). Cambridge University Press: 75–90.

Freire, Danilo, and Gary Uzonyi. 2018. “What Drives State-Sponsored Violence?: Evidence from Extreme Bounds Analysis and Ensemble Learning Models.” *SocArXiv* 15: 1–15.

Friedman, Jerome H, and Jacqueline J Meulman. 2003. “Multiple Additive Regression Trees with Application in Epidemiology.” *Statistics in Medicine* 22 (9). Wiley Online Library: 1365–81.

Google AI Blog. 2020. “An End-to-End AutoML Solution for Tabular Data at KaggleDays.” <https://ai.googleblog.com/2019/05/an-end-to-end-automl-solution-for.html>.

Grömping, Ulrike. 2009. “Variable Importance Assessment in Regression: Linear Regression versus Random Forest.” *The American Statistician* 63 (4). Taylor & Francis: 308–19.

H2O.ai. 2017. *H2O Automl*. <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html>.

He, Xin, Kaiyong Zhao, and Xiaowen Chu. 2020. “AutoML: A Survey of the State-of-the-Art.” *Knowledge-Based Systems* 212. Elsevier: 1–27.

Healy, John, Leland McInnes, and Colin Weir. 2017. “Bridging the Cyber-Analysis Gap: The Democratization of Data Science.” *The Cyber Defense Review* 2 (1). JSTOR: 109–18.

Hill Jr, Daniel W, and Zachary M Jones. 2014. “An Empirical Evaluation of Explanations for State Repression.” *American Political Science Review*. JSTOR, 661–87.

Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren. 2019. “Automated Machine Learning: Methods, Systems, Challenges.” In. Cham: Springer Nature.

Jin, Haifeng, Qingquan Song, and Xia Hu. 2019. “Auto-Keras: An Efficient Neural Architecture Search System.” In *Proceedings of the 25th Acm Sigkdd International Conference on Knowledge Discovery & Data Mining*, 1946–56. ACM.

Kleinberg, Jon, Jens Ludwig, Sendhil Mullainathan, and Ziad Obermeyer. 2015. “Prediction Policy Problems.” *American Economic Review* 105 (5): 491–95.

Liu, Zhengying, Adrien Pavao, Zhen Xu, Sergio Escalera, Isabelle Guyon, Julio CS Jacques Junior, Meysam Madadi, and Sebastien Treguer. 2020. “How Far Are We from True AutoML: Reflection from Winning Solutions and Results of AutoDL Challenge.” In *7th Icml Workshop on Automated Machine Learning*, 1–12.

Lundberg, Scott M, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. “From Local Explanations to Global Understanding with Explainable AI for Trees.” *Nature Machine Intelligence* 2 (1): 2522–5839.

Molnar, Christoph. 2020. *Interpretable Machine Learning*. Victoria: Leanpub.

Muchlinski, David, David Siroky, Jingrui He, and Matthew Kocher. 2016. “Comparing Random Forest with Logistic Regression for Predicting Class-Imbalanced Civil War Onset Data.” *Political Analysis*. JSTOR, 87–103.

Olson, Randal S, and Jason H Moore. 2016. “TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning.” In *Workshop on Automatic Machine Learning*, 66–74. PMLR.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–30.

Shang, Zeyuan, Emanuel Zgraggen, Benedetto Buratti, Ferdinand Kossmann, Philipp Eichmann, Yeounoh Chung, Carsten Binnig, Eli Upfal, and Tim Kraska. 2019. “Democratizing Data Science through Interactive Curation of ML Pipelines.” In *Proceedings of the 2019 International Conference on Management of Data*, 1171–88.

Thornton, Chris, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2013. “Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms.” In *Proceedings of the 19th Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 847–55.

Truong, Anh, Austin Walters, Jeremy Goodsitt, Keegan Hines, C Bayan Bruss, and Reza Farivar. 2019. “Towards Automated Machine Learning: Evaluation and Comparison of AutoML Approaches and Tools.” In *2019 Ieee 31st International Conference on Tools with Artificial Intelligence (Ictai)*, 1471–9. IEEE.

Ulfelder, Jay. 2013. “A Multimodel Ensemble for Forecasting Onsets of State-Sponsored Mass Killing.” In *APSA 2013 Annual Meeting Paper*, 1–23.

Ward, Michael D, Brian D Greenhill, and Kristin M Bakke. 2010. “The Perils of Policy by P-Value: Predicting Civil Conflicts.” *Journal of Peace Research* 47 (4). Sage Publications Sage UK: London, England: 363–75.

Yang, Qian, Jina Suh, Nan-Chen Chen, and Gonzalo Ramos. 2018. “Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models.” In *Proceedings of the 2018 Designing Interactive Systems Conference*, 573–84.

1. Independent researcher, [danilofreire@gmail.com](mailto:danilofreire@gmail.com), <https://danilofreire.github.io>. [↑](#footnote-ref-0)
2. Please visit <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/explain.html> for more information on H2OAutoML’s model explainability functions. [↑](#footnote-ref-1)
3. A list of the awards is available at <http://automl.info/tpot/>. [↑](#footnote-ref-2)
4. Please find the Docker documentation files at <https://docs.docker.com>. [↑](#footnote-ref-3)