# Democratizing Policy Analytics with AutoML

Danilo Freire

MERCATUS WORKING PAPER

MERCATUS CENTER
George Mason University
3434 Washington Blvd., 4th Floor, Arlington, Virginia 22201
www.mercatus.org

*Danilo Freire, "Democratizing Policy Analytics with AutoML," Mercatus Working Paper, Mercatus Center at George Mason University, Arlington, VA, March 2021.*

**Abstract**

Machine learning methods have made significant inroads in the social sciences. Computer algorithms now help scholars design cost-effective public policies, predict rare social events, and improve the allocation of funds. However, building and evaluating machine learning algorithms remain labor-intensive, error-prone tasks. Thus, areas that could benefit from modern computer algorithms are often held back owing to implementation challenges or lack of technical expertise. In this paper, I show how scholars can use automated machine learning (AutoML) tools to preprocess their data and create powerful estimation methods with minimal human input. I demonstrate the functionalities of three open-source, easy-to-use AutoML algorithms, and I replicate a well-designed forecasting model to highlight how researchers can achieve similar results with only a few lines of code.

*JEL* Codes: C52, C53, C87

Keywords: automated machine learning, AutoML, computational social science, policy analysis, research methods, teaching

**Author Affiliation and Contact Information**

Danilo Freire
Independent researcher
https://danilofreire.github.io
danilofreire@gmail.com

# Democratizing Policy Analytics with AutoML

Danilo Freire

## 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 modeling 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 and Jones 2014). Supervised machine learning also helps governments 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 labor-intensive task. Practitioners apply several preprocessing steps just to prepare their data, and many modeling decisions, such as algorithm selection or parameter optimization, 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 owing 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 optimizes machine learning algorithms. More specifically, AutoML provides data-driven tools to minimize human effort in the machine learning workflow, automating steps such as 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 also makes state-of-the-art algorithms accessible to regular users. According to its proponents, AutoML promotes a true democratization of artificial intelligence (Hutter, Kotthoff, and Vanschoren 2019, ix; Shang et al. 2019). Also, AutoML approaches have been very successful in prediction challenges, and they consistently reach the top 5 percent in public machine learning competitions (Prasanna 2020; Lu 2019).

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 2021), 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. Although 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 coauthors and referees.

## 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 optimization, 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 nonexpert users, which are not only able to predict numeric data but are also able 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 that achieves the best performance in a given evaluation metric, such as the mean squared error or classification accuracy. Having selected the algorithm that minimizes 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, standardizing numeric covariates, and removing correlated features to avoid multicollinearity (He, Zhao, and Chu 2021; Truong et al. 2019). AutoML takes a data-driven approach here too and selects those data transformations that improve forecasting scores the most.

***AutoKeras***

AutoKeras is an AutoML algorithm based on Keras (Keras 2021), 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 optimize 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, *X* is the set of predictors and *y* 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 *y*. Users can also pass many parameters to the ak.StructuredDataClassifier() function, including the metric they want to minimize or maximize (such as accuracy or area under the ROC curve), set the number of networks that 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.

***H2O AutoML***

The second algorithm I discuss here is H2O AutoML. Developed by H2O.ai, a company based in

Silicon Valley, H2O AutoML 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 get started with the platform. H2O.ai offers its AutoML software for both R and Python, and the packages use the same functions and arguments in both of those languages. Users need only 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 the model's performance on the test dataset, and report model statistics. Example code for binary classification tasks in Python is as follows:

```
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,                # run 30 machine learning models
seed=1234)
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 functions with explanatory power. 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 obscurity has serious consequences in fields where decision mechanisms are relevant per se, such as judicial sentencing or healthcare allocation. H2O AutoML addresses this issue by offering explanation methods that describe how the general model performs and how it explains each individual observation.[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).

[1] Please visit http://docs.h2o.ai/h2o/latest-stable/h2o-docs/explain.html for more information on H2OAutoML’s model explainability functions.

## *TPOT*

The last algorithm I introduce in this section is TPOT, or the Tree-based Pipeline Optimization Tool. It is one of the oldest AutoML solutions for Python, and its authors have won several awards for their work.[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 until it reaches 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 following example code:

```
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 $X$ is a matrix of covariates and $y$ is the response variable. TPOT has an export function that is useful for those who need to export the optimized model and deploy it in other settings. Users can also customize TPOT's hyperparameters for classification and regression tasks. TPOT's documentation is available at http://epistasislab.github.io/tpot/.

[2] A list of the awards is available at http://automl.info/tpot/.

As one can see, the code shown in the three examples is almost identical, though the functions are running different processes in the background. However, AutoKeras, H2O AutoML, and TPOT can all quickly estimate regression or classification models for numeric data. Since these are the two tasks policy analysts do most often, the three algorithms presented above can be easily integrated into their machine learning workflow.

## AutoML in Practice: Replication

How do AutoML models compare with expert-coded machine learning? AutoML algorithms have frequently appeared among 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 my 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 among 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, my results are a simple baseline that allows for modifications and extensions. The only data processing tasks I do are create a training dataset–test dataset split (75 percent to 25 percent) before the estimation, as some libraries do not partition the data automatically, and add five cross-validation folds to test the models' prediction accuracy. To save space, I do 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 (8305).

I begin with the civil war data collected by Fearon and Laitin (2003). The data have 6,402 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. The latter authors have a different measurement for civil war onset, and their dataset has 688 country-years and nine 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 appear in table 1.

**Table 1. 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 |
| H2O AutoML | **0.783** | 0.703 |
| TPOT | 0.715 | **0.825** |

Note: Numbers in bold indicate better predictive performance than the baseline model.

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 (shown in bold type). Considering that civil conflicts are hard to predict and that the algorithms had limited modeling time, the results indicate AutoML's strong forecasting accuracy even in adverse conditions.

## Sharing AutoML Models with Docker

Once one has estimated AutoML models, how should he or she deploy or share them? My suggestion is to use Docker as a reproducibility tool.[3] Docker is a virtualization platform that allows users to build, test, and share their software in standardized 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, scholars can distribute their complete software environment to collaborators and reviewers, as is common practice in the social sciences. 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.

[3] Please find the Docker documentation files at https://docs.docker.com.

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 Docker Hub (https://hub.docker.com/signup), which is a cloud-based repository for Docker images. After that, one is 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 because it requires less coding.

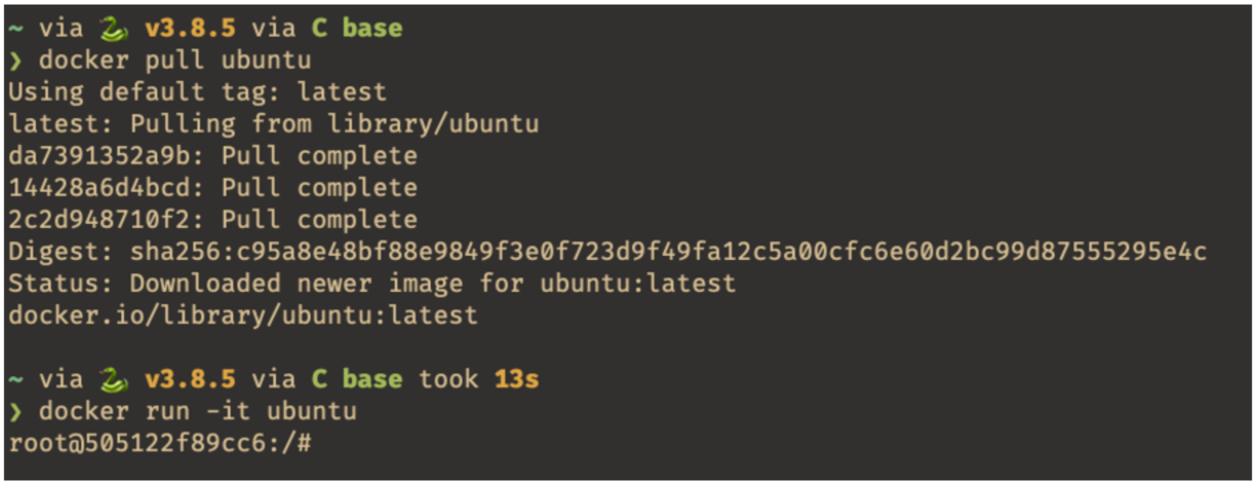
I start by pulling and running a prebuilt Ubuntu Linux image. Docker will start an Ubuntu session without changing any configuration in one's computer. To install the container, I run the following code in my terminal:

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

Upon doing so, I see a root session in the terminal (see figure 1). 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
```

**Figure 1. Docker Container Running Ubuntu Linux.**



```
~ via 🐍 v3.8.5 via C base
❯ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
Digest: sha256:c95a8e48bf88e9849f3e0f723d9f49fa12c5a00cfc6e60d2bc99d87555295e4c
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

~ via 🐍 v3.8.5 via C base took 13s
❯ docker run -it ubuntu
root@505122f89cc6:/#
```

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

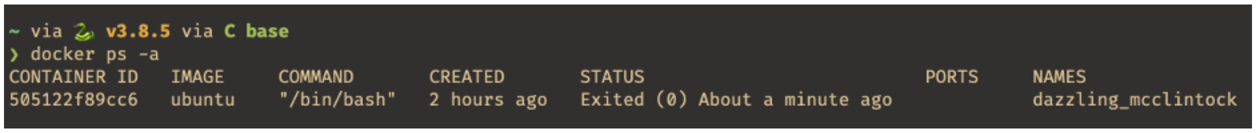
```
                                          # In the Docker container:
exit                                          # stop the container
                                          # In your regular terminal:
docker ps -a                            # list all available containers
```

When I exit the Docker image and type docker ps -a, I see something like what is shown in figure 2.

The first column indicates the container ID. In this case, it starts with 5051. To copy the files to that specific container, I just write the following lines in my 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
```

**Figure 2. List of Available Docker Containers**



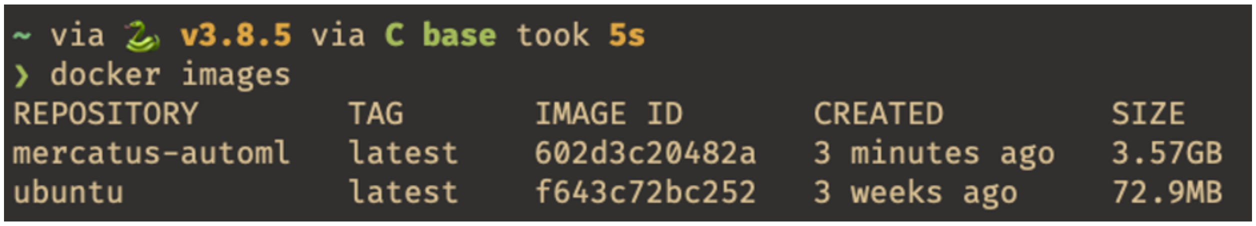
```
~ via 🐍 v3.8.5 via C base
❯ docker ps -a
CONTAINER ID   IMAGE     COMMAND       CREATED       STATUS                         PORTS     NAMES
505122f89cc6   ubuntu    "/bin/bash"   2 hours ago   Exited (0) About a minute ago            dazzling_mcclintock
```

Lastly, I need to save the changes I have made to the container and upload it to Docker Hub. I use *docker commit [container_ID] [new_name]* to commit the changes, where *[container_ID]* is the ID value given earlier (5051) and *[new_name]* is the name I want to give to the modified container. I also check if the container has been saved with docker image.

```
docker commit 5051 mercatus-automl
docker images
```

The terminal output is shown in figure 3.

**Figure 3. Docker Images**



```
~ via 🐍 v3.8.5 via C base took 5s
❯ docker images
REPOSITORY        TAG       IMAGE ID       CREATED          SIZE
mercatus-automl   latest    602d3c20482a   3 minutes ago    3.57GB
ubuntu            latest    f643c72bc252   3 weeks ago      72.9MB
```

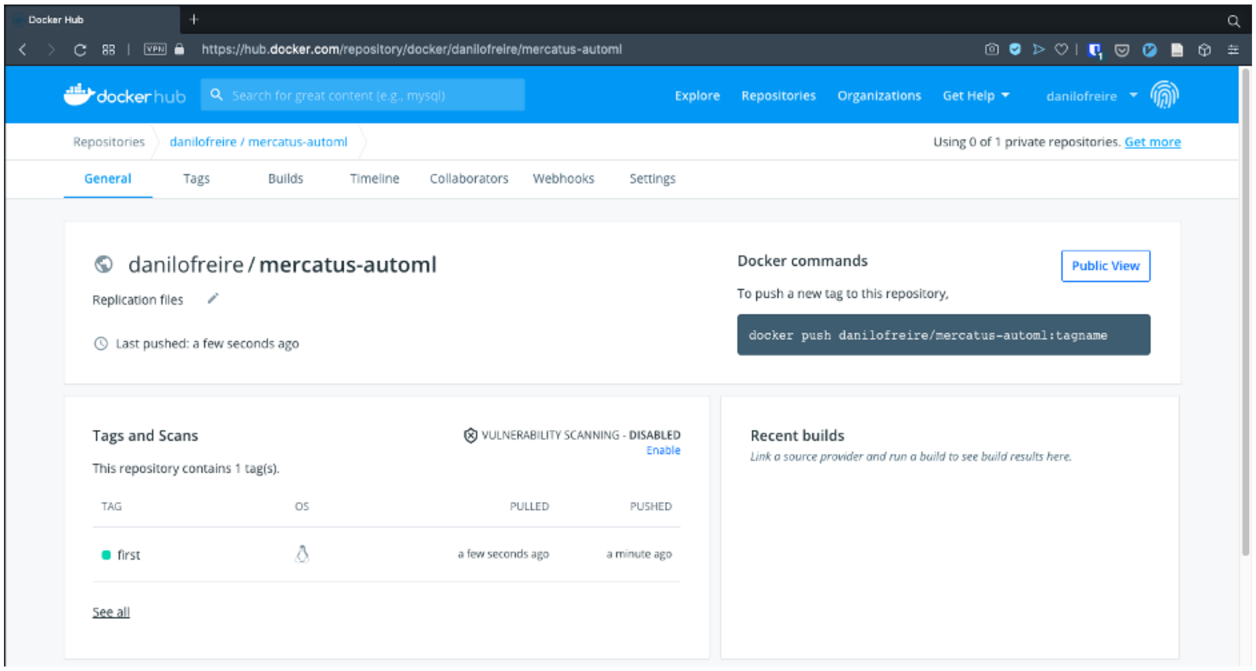
Now I need to push the image to Docker Hub. I go to https://hub.docker.com /repositories and create a new repository. Then I add my Docker Hub credentials to my local machine with *docker login --username=your_username* and create a tag for my container. Note that the container ID has been updated (602d). Finally, I type *docker push my_username/repository_name* to push my image to Docker Hub. Here's some 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 Docker Hub
```

Upon a successful upload, I see what is shown in figure 4.

**Figure 4. Container Uploaded to Docker Hub.**



Using 0 of 1 private repositories. Get more

danilofreire / mercatus-automl

Replication files

Last pushed: a few seconds ago

Docker commands

To push a new tag to this repository,

```
docker push danilofreire/mercatus-automl:tagname
```

Tags and Scans

VULNERABILITY SCANNING - DISABLED

This repository contains 1 tag(s).

| TAG | OS | PULLED | PUSHED |
|---|---|---|---|
| first | | a few seconds ago | a minute ago |

Recent builds

*Link a source provider and run a build to see build results here.*

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

## Conclusion

Computer scientists have applied machine learning to predict a myriad of outcomes, such as shopping habits and kidney diseases. Algorithms now power email filters, translation software, satellite farming, self-driving cars, and many other devices. In the past few 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 nonexperts 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 that achieved by area experts. Moreover, I have suggested that users should adopt Docker to share their machine learning models and achieve 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 one 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. Whereas nonexperts can use AutoML tools to produce good estimates with minimal coding experience, practitioners are able to automate labor-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 answer some of their most pressing questions.

## References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, et al. 2015. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems." Preliminary white paper, November 9. https://arxiv.org/pdf/1603.04467.pdf.

Ahmed, Shakeel, Ravi S. Mula, and Soma S. Dhavala. 2020. "A Framework for Democratizing AI." Preprint, submitted January 6, 2020. https://arxiv.org/pdf/2001.00818.pdf.

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." Paper presented at the 41st

International Conference on Software Engineering: Software Engineering in Practice, Montréal, Canada, May. https://www.microsoft.com/en-us/research/uploads/prod/2019/03/amershi-icse-2019_Software_Engineering_for_Machine_Learning.pdf.

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, no. 6324 (February): 483–85.

Collier, Paul, and Anke Hoeffler. 2004. "Greed and Grievance in Civil War." *Oxford Economic Papers* 56, no. 4 (October): 563–95.

Elshawi, Radwa, Mohamed Maher, and Sherif Sakr. 2019. "Automated Machine Learning: State-of-the-Art and Open Challenges." Unpublished manuscript, last revised June 11. PDF file. https://arxiv.org/pdf/1906.02287.pdf.

Fearon, James D., and David D. Laitin. 2003. "Ethnicity, Insurgency, and Civil War." *American Political Science Review* 97, no. 1 (February): 75–90.

Freire, Danilo, and Gary Uzonyi. 2018. "What Drives State-Sponsored Violence?: Evidence from Extreme Bounds Analysis and Ensemble Learning Models." Unpublished manuscript, last edited November 27. PDF file. https://osf.io/preprints/socarxiv/pzx3q/download.

Friedman, Jerome H., and Jacqueline J. Meulman. 2003. "Multiple Additive Regression Trees with Application in Epidemiology." *Statistics in Medicine* 22, no. 9 (May): 1365–81.

Grömping, Ulrike. 2009. "Variable Importance Assessment in Regression: Linear Regression versus Random Forest." *American Statistician* 63, no. 4 (November): 308–19.

H2O.ai. Last updated February 21, 2021. "AutoML: Automatic Machine Learning." http://docs.h2o.ai/h2o/latest-stable/h2o-docs/automl.html.

He, Xin, Kaiyong Zhao, and Xiaowen Chu. 2021. "AutoML: A Survey of the State-of-the-Art." *Knowledge-Based Systems* 212 (January): 1–27.

Healy, John, Leland McInnes, and Colin Weir. 2017. "Bridging the Cyber-Analysis Gap: The Democratization of Data Science." *Cyber Defense Review* 2, no. 1 (winter): 109–18.

Hill Jr., Daniel W., and Zachary M. Jones. 2014. "An Empirical Evaluation of Explanations for State Repression." *American Political Science Review* 108, no. 3 (August): 661–87.

Hutter, Frank, Lars Kotthoff, and Joaquin Vanschoren, ed. 2019. *Automated Machine Learning: Methods, Systems, Challenges*. Cham, Switzerland: Springer Nature Switzerland AG.

Jin, Haifeng, Qingquan Song, and Xia Hu. 2019. "Auto-Keras: An Efficient Neural Architecture Search System." In KDD '19: Proceedings of the 25th ACM SIGKDD International

Conference on Knowledge Discovery & Data Mining, 1946–56. New York: Association for Computing Machinery.

Keras (website). n.d. Accessed February 24, 2021. https://keras.io.

Kleinberg, Jon, Jens Ludwig, Sendhil Mullainathan, and Ziad Obermeyer. 2015. "Prediction Policy Problems." *American Economic Review* 105, no. 5 (May): 491–95.

Liu, Zhengying, Adrien Pavao, Zhen Xu, Sergio Escalera, Isabelle Guyon, Julio C. S. Jacques Junior, Meysam Madadi, and Sebastien Treguer. 2020. "How Far Are We from True AutoML: Reflection from Winning Solutions and Results of AutoDL Challenge." Paper presented at the 7th ICML Workshop on Automated Machine Learning, virtual, July 18.

Lu, Yifeng. 2019. "An End-to-End AutoML Solution for Tabular Data at KaggleDays." Google AI Blog. May 9. https://ai.googleblog.com/2019/05/an-end-to-end-automl-solution-for.html.

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): 56–67.

Molnar, Christoph. 2020. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.* Victoria, British Columbia: 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* 24, no. 1 (winter): 87–103.

Olson, Randal S., and Jason H. Moore. 2016. "TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning." *Proceedings of Machine Learning Research* 64: 66–74.

Pedregosa, Fabian, Gaël Varoquaux, Aalexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Prasanna, Shashank. 2020. "Machine Learning with AutoGluon, an Open Source AutoML Library." AWS Open Source Blog. March 31.

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 *SIGMOD '19: Proceedings of the 2019 International Conference on Management of Data*, 1171–88. New York: Association for Computing Machinery.

Thornton, Chris, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms." In *KDD '13: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge*

*Discovery and Data Mining*, edited by Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, and Jingrui He, 847–55. New York: Association for Computing Machinery.

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)*, 1471–79. Portland, OR: Institute of Electrical and Electronics Engineers.

Ulfelder, Jay. 2013. “A Multimodel Ensemble for Forecasting Onsets of State-Sponsored Mass Killing.” Paper presented at the 2013 APSA Annual Meeting and Exhibition, Chicago, IL, August 1.

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): 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 *DIS ’18: Proceedings of the 2018 Designing Interactive Systems Conference*, 573–84. New York: Association for Computing Machinery.