**Comparing AutoML Tools in Titanic Dataset — Machine Learning**

[[](https://medium.com/@leomaurodesenv?source=post_page-----87ae3b23228a--------------------------------)](https://medium.com/@leomaurodesenv?source=post_page-----87ae3b23228a--------------------------------)

[[Sinch Blog](https://medium.com/wearesinch?source=post_page-----87ae3b23228a--------------------------------)](https://medium.com/wearesinch?source=post_page-----87ae3b23228a--------------------------------)

[Leonardo Mauro P. Moraes](https://medium.com/@leomaurodesenv?source=post_page-----87ae3b23228a--------------------------------)

·

Follow

Published in

[Sinch Blog](https://medium.com/wearesinch?source=post_page-----87ae3b23228a--------------------------------)

·

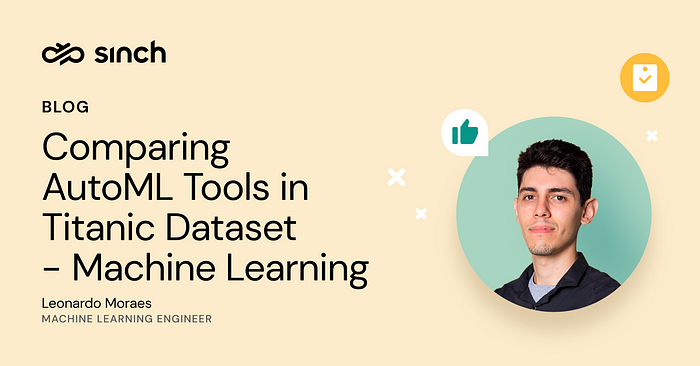
6 min read

·

Dec 5, 2022

135

1



**What is AutoML?** Automated Machine Learning (AutoML) is the process of automating the tasks of applying machine learning to problems. Machine learning lifecycle is composed of several steps including, but not limited to, data selection, data preprocessing, data mining, model evaluation, model fine-tuning, and deployment [Adapted from Fayyad, 96]. Thus, AutoML is the process of automatizing one or more steps from the machine learning lifecycle. In this way, we can improve the efficiency of machine learning experiments and accelerate research on the problem.

**How can we use AutoML?** Okay, let’s use AutoML. The problem is that there are multiple Python packages, so how can I choose the right package for my project? It is not very clear how to do that; and it is a time-consuming experience for all of them. In this blog post, we are going to explore a set of 11 (eleven) well-known Python packages for AutoML.

**Experimentation Setup**

For this AutoML benchmark, we used the dataset from [Titanic — Machine Learning from Disaster](https://www.kaggle.com/competitions/titanic) competition, simply called the Titanic dataset. This is a competition to use machine learning to create a classification model that predicts which passengers survived the sinking of the Titanic. The titanic dataset is composed of twelve columns of categorical and numerical data, such as “Pclass”, “Name”, “Sex”, “Fare”, “Cabin”, “Embarked”, etc. Since many of the AutoML packages do not process categorical data, I performed a simple preprocessing to remove and transform the columns.

The train set counts just 891 samples. We are going to use this set to train and validate our AutoML packages. The results of the test will then be sent to the competition. We are going to normalize the train set using an Encoder and Imputer (for missing data). I encoded the columns “Sex”, “Embarked” to numerical columns, and executed the Imputer into the columns “Pclass”, “Sex”, “SibSp”, “Parch”, “Fare”, “Age”, “Embarked”. You can access the Jupyter Notebook code from the public link at the end of the blog post.

*Note, a few codes are written in markdown in the Jupyter Notebook, because it uses incompatible packages for the other packages. But they all have the installation and run code.*

**# Lazy Predict**

[[GitHub](https://github.com/shankarpandala/lazypredict), [Documentation](https://lazypredict.readthedocs.io/)] Lazy Predict helps build a lot of basic models (from scikit-learn and similar libraries) without much code, and helps understand which models work better without any parameter tuning. In other words, Lazy Predict is a good library to test multiple solutions at once quickly. It only includes preprocessing and model training; thus, it does not perform any fine-tuning.

# Lazy Predict  
from lazypredict.Supervised import LazyClassifier  
  
automl = LazyClassifier()  
results, \_ = automl.fit(X\_train, X\_train, y\_train, y\_train)

**# hyperopt-sklearn**

[[GitHub](https://github.com/hyperopt/hyperopt-sklearn), [Documentation](http://hyperopt.github.io/hyperopt-sklearn/)] hyperopt-sklearn (hyperparameter optimization for scikit-learn) is hyperopt-based model selection among machine learning algorithms in scikit-learn. In contrast with Lazy Predict, hyperopt-sklearn can perform hyperparameter tuning in the models, but it does not perform data preprocessing automatically.

# hyperopt-sklearn  
from hpsklearn import HyperoptEstimator  
  
automl = HyperoptEstimator()  
automl.fit(X\_train, y\_train)

**# auto-sklearn**

[[GitHub](https://github.com/automl/auto-sklearn), [Documentation](https://automl.github.io/auto-sklearn/master/)] auto-sklearn is a toolkit on top of scikit-learn estimator. In summary, it combines data preprocessing, feature preprocessing and classifier evaluation. Note, it does not train simple models, auto-sklearn ensemble models to get better performance. It is good for competitions, but not for production models.

# auto-sklearn  
from autosklearn.classification import AutoSklearnClassifier  
  
automl = AutoSklearnClassifier()  
automl.fit(X\_train, y\_train)

**# TPOT**

[[GitHub](https://github.com/EpistasisLab/tpot), [Documentation](http://epistasislab.github.io/tpot/)] TPOT stands for Tree-based Pipeline Optimization Tool. It optimizes machine learning pipelines using genetic programming. It looks like a combination of hyperopt-sklearn fine-tuning and auto-sklearn data preprocessing, however, it does not ensemble the models — it keeps models simple and interpretable. Nice tool for high performance and simple modeling.

# TPOT  
from tpot import TPOTClassifier  
  
automl = TPOTClassifier()  
automl.fit(X\_train, y\_train)

**# MLJAR**

[[GitHub](https://github.com/mljar/mljar-supervised), [Documentation](https://supervised.mljar.com/)] mljar-supervised abstracts the common way to preprocess the data, construct the machine learning models, and perform hyperparameter tuning to find the best model. Also, it supports explainability and automatic exploratory data analysis. It contains more features than TPOT, but it is more complex too. Also, it can support textual data, but it was developed for tabular data only.

# MLJAR  
from supervised.automl import AutoML  
  
automl = AutoML(mode="Compete")  
automl.fit(X\_train, y\_train)

**# FLAML**

[[GitHub](https://github.com/microsoft/FLAML), [Documentation](https://microsoft.github.io/FLAML/)] FLAML is a package that finds accurate machine learning models automatically. It frees users from selecting learners and hyperparameters for each learner. It can also be used to tune generic hyperparameters for MLOps workflows, models, algorithms, computing experiments, software configurations and so on. It contains many features as MLJAR has, without explainability. Also, it can process textual data and can work with online learning as well.

# FLAML  
from flaml import AutoML  
  
automl = AutoML()  
automl.fit(X\_train, y\_train, task="classification")

**# AutoGluon**

[[GitHub](https://github.com/awslabs/autogluon), [Documentation](https://auto.gluon.ai/)] AutoGluon automates machine learning tasks enabling you to easily achieve strong predictive performance in your applications. This package is focused on training, evaluation, fine-tuning, and deployment. It can works with textual data, tabular data, image, time series, and even multi-modal data. It is the most robust and generic AutoML tool so far.

# AutoGluon  
from autogluon.tabular import TabularPredictor  
  
automl = TabularPredictor(label="Survived", path="titanic-autogluon")  
automl.fit(train) # pd.DataFrame as parameter

**# H2O**

[[GitHub](https://github.com/h2oai/h2o-3), [Documentation](https://h2o.ai/)] H2O is an in-memory platform for distributed, scalable machine learning. It provides implementations of many popular algorithms; but it didn’t use scikit-learn implementations, it contains its own implementations. It can reach nice results, but it only works with its own pre-built algorithms and models.

# H2O  
import h2o  
  
# Start the H2O cluster (locally)  
h2o.init()  
# Get H2O data frame  
hf\_train = h2o.H2OFrame(train)  
  
# Run the AutoML  
automl = H2OAutoML()  
automl.train(x=x\_columns, y="Survived", training\_frame=hf\_train)

**# AutoKeras**

[[GitHub](https://github.com/keras-team/autokeras), [Documentation](https://autokeras.com/)] AutoKeras is a package based on Keras, a Deep Learning library. AutoKeras can perform AutoML into different types of data, such as tabular, image or even textual data. Deep Learning models tend to be good with real data.

# AutoKeras  
import autokeras as ak  
  
automl = ak.StructuredDataClassifier()  
automl.fit(X\_train, y\_train, epochs=10)

**# MLBox**

[[GitHub](https://github.com/AxeldeRomblay/MLBox), [Documentation](https://mlbox.readthedocs.io/en/latest/)] MLBox provides data preprocessing, feature selection, hyperparameter optimization and model evaluation. It contains several state-of-the-art competition models, such as Deep Learning, Stacking, LightGBM. Also, a few interpretation functions to analyze our results. It works only with tabular data.

# MLBox  
# Init MLBox  
from mlbox.preprocessing import \*  
from mlbox.optimisation import \*  
from mlbox.prediction import \*  
  
# Reading data  
rd = Reader(sep = ",")  
df = rd.train\_test\_split(paths, "Survived")  
  
# AutoML  
opt = Optimiser(scoring = "accuracy", n\_folds = 5)  
params = opt.optimise(space, df, 15)  
  
# Making Predictions  
prd = Predictor()  
prd.fit\_predict(params, df)

**# PyCaret**

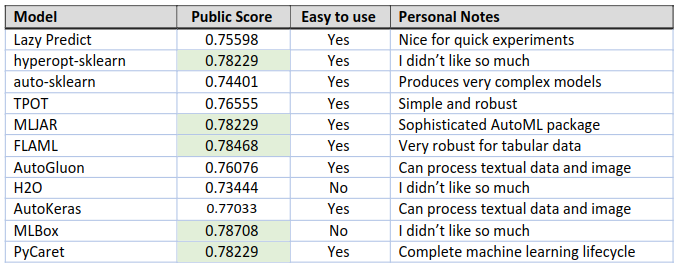
[[GitHub](https://github.com/pycaret/pycaret), [Documentation](https://pycaret.org/)] PyCaret is a low-code machine learning library that automates machine learning lifecycle. It is an end-to-end machine learning and model management tool that makes you more productive and speeds up the experiment cycle exponentially; AutoML is just one of the features. This is the most complete package on this list.

# PyCaret  
from pycaret import classification  
  
s = classification.setup(train, target = "Survived")  
best = classification.compare\_models()  
classification.plot\_model(best)

**Conclusion**

In this blog post, I presented and roughly evaluated eleven well-known Python packages for AutoML. Most of them work with tabular data, but few stand out due to their ability to work with textual data, images, time series, and even multi-modal projects.

It is important to note that I have not fully explored all the packages, so we could use more of their features to achieve better results. Anyway, they all achieved good results, averaging 76% F1-score.



AutoML Tools — Packages evaluation

In summary, I liked most Lazy Predict, TPOT, MLJAR, FLAML, AutoGluon, AutoKeras and PyCaret. Of course, each of them has their own advantages and disadvantages. *See the complete code in the****Jupyter Notebook****:*

**[AutoML Benchmark - Titanic Disaster](https://www.kaggle.com/code/leomauro/automl-benchmark-titanic-disaster?source=post_page-----87ae3b23228a--------------------------------" \t "_blank)**

[Explore and run machine learning code with Kaggle Notebooks | Using data from Titanic - Machine Learning from Disaster](https://www.kaggle.com/code/leomauro/automl-benchmark-titanic-disaster?source=post_page-----87ae3b23228a--------------------------------" \t "_blank)

[www.kaggle.com](https://www.kaggle.com/code/leomauro/automl-benchmark-titanic-disaster?source=post_page-----87ae3b23228a--------------------------------" \t "_blank)

**Reference**

Fayyad, Usama, Gregory Piatetsky-Shapiro, and Padhraic Smyth. [From data mining to knowledge discovery in databases](https://ojs.aaai.org/index.php/aimagazine/article/view/1230). AI magazine 17.3 (1996): 37–37.