Smart Model Elimination for Efficient Automated Machine Learning

Eric Su Zhang[*]
St. Mark's School of Texas
10600 Preston Rd, Dallas, TX 75230
`ericspring08@gmail.com`

Benjamin Standefer
St. Mark's School of Texas
10600 Preston Rd, Dallas, TX 75230
`bjmstandefer@gmail.com`

Abstract

Automated Machine Learning or AutoML has emerged as a popular field of research. Many of these frameworks optimize a form of model selection, however, they all require most models to be run and evaluated. We propose a novel framework that automatically eliminates models that are unlikely to be performant by training a Boosting Model on hundreds of kaggle datasets. Out of the 30 model options, our framework can predict a top five model within its top eight 80% of the time.

1   Introduction

Machine learning (ML) is a type of artificial intelligence (AI) that uses a variety of algorithms to interpret data and extrapolate from it, making predictions based on observed trends. Individual models are trained on datasets to become smarter so as to make predictions based on new data. The development and optimization of these models is a time-consuming process involving statistical nuances and complex learning strategies. This has led to the emergence of Automated Machine Learning (AutoML) frameworks and research.

One sector of the global industry that has the most potential for beneficial integration of AutoML frameworks is healthcare, particularly in developing countries. These countries have severe shortages of healthcare workers and limited tools for diagnosis. For example, Africa has 2.3 healthcare workers per 1000 individuals, while the Americas have 24.8 healthcare workers per 1000. The World Health Organization (WHO) emphasized that this deficit is growing every year and will likely reach 18 million personnel by 2030. The use of AI in healthcare has been varied. Medical AI's typically automate repetitive tasks, making time consumption a primary concern. This coupled with a lack of adequate resources makes designing faster diagnostic systems for developing countries' medical sectors a pivotal issue. Most AutoML frameworks implement some form of model selection where a pool of models are filtered until a final model is selected. However, these frameworks require that every model to be run to filter out. In theory, this means that much energy is spent training models that are unlikely to be selected. We propose SMEML, a novel model selection algorithm that automatically eliminates models that it believes will not be performant.

1.1   Survey

---

[*]Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

Factor	H2O AutoML	TPOT	MLJAR
Ease of Use	Easy-to-use GUI and API	Python library, requires coding experience	Very user-fr…
Supported Algorithms	Linear, Tree, DNN, Clustering, etc.	Genetic Programming	Linear, Tree…
Feature Engineering	Basic	Basic	Advanced - …
Time Management	Time Constraints are Settable	Generations and population size	Time constra…
Ensemble Methods	Yes (stacking + blending)	Yes (stacking)	Yes (stackin…
Hyperparameter Tuning	Automated	Genetic algorithm	Automated a…
Model Interpretability	SHAP, LIME, PDPs, MOJO	POJO, etc.	Basic
Scalability	Good (distributed computing)	Limited	Good
Customization	High	Moderate	High

Table 1: Comparison of six AutoML frameworks based on commonmly evaluated criteria as well as more obscure features relevant to the medical diagnosis process

We systematically reviewed and compared six different AutoML frameworks, comparing every feature relevant to medical implementation. Of these six, we wanted to take a closer look at four: FLAML, H2O, MLJAR, and TPOT. By testing our new framework against these four in evaluating medical data, we hoped to delineate key weaknesses and strengths in our own design.

Taking a closer look at four AutoML standards, we can observe pros and cons in their theoretical application to medicine and research. FLAML uses blend search hyper-parameter optimization and focuses on lightweight models, making it quick and applicable to repetitive tasks in the medical sphere. It supports imputation techniques for missing values with user specification and has an intuitive API, making it moderately user-friendly for minimally trained healthcare workers. H2O has a wider variety of supported algorithms that may or may not be appropriate for medical diagnosis, making it slower than FLAML on Mean. In one trial, more than half of FLAML's performances in one minute were better than or equal to H20's performances in one hour. H2O has automatic and multi-faceted methods of dealing with missing values and flexible interfaces in R and Python, making it intuitive for users. MLJAR uses more advanced algorithms such as light gradient boosting and neural networks. It automatically deals with missing values and is known for its automated user interface. TPOT uses genetic programming. It builds pipelines over multiple generations, which can be time consuming. It has built-in pre-processing for missing values, but the genetic programming can require fine-tuning from users.

Keeping the factors we have laid out in mind, namely expeditious running, accuracy, and interpretability, we also uncovered a key continuity in framework design: frameworks are forced to run and train the majority of their available models to maximize accuracy at the expense of efficiency. Given the restricted nature of tabular data, there are some models that, in general, are less likely to perform well, thus running and training these models is frequently wasteful. Cutting these models completely, however, is not optimal either, as they are the best option in a minority of cases and give frameworks versatility. We hypothesized that using a combination of layered machine learning implementation in which we train a model to predict a basic framework's outputs and meta-feature extraction in which we accelerate the training process by reducing complicated data matrices to their measurable attributes would yield an equally versatile system that minimizes time spent on prediction while still maximizing accuracy and straightforward implementation in a medical context.

2   Methodology

2.1   Observation

When considering a specific dataset, various aspects regarding the distribution and relationship within the data can give us clues as to which models are likely to perform best. For example, linear models tend to work better on datasets where there is a linear relationship between features. As a result, we can develop a boosting model to predict the best models based off extracted attributes from a dataset.

2.2   Data Extraction

To train our boosting model, we collected over a thousand datasets from kaggle, a popular website for hosting public datasets. After used various medical terms to filter down datasets and limiting

Table 2: Attributes extracted from a dataset

Attribute	Formula	Purpose
# of Rows	$\lvert\mathbf{A}\rvert_r$	Data Size
# of Columns	$\lvert\mathbf{A}\rvert_c$	Data Size
Target Distribution	$\frac{\max(\text{count}_1,\text{count}_0)}{\lvert\mathbf{A}\rvert_r}$	Data Type
Numerical Columns	$\frac{\lvert\mathbf{A}_{num}\rvert}{\lvert\mathbf{A}\rvert_c}$	Data Type
Binary Categorical Columns	$\frac{\lvert\mathbf{A}_{bin}\rvert}{\lvert\mathbf{A}\rvert_c}$	Data Type
Mean # of Distinct Values	$\mu\{\text{\# of distinct values in column}\}$	Data Type
Mean IQR	$\mu\{\text{IQR of column}\}$	Data Distribution
Standard Deviation of IQR	$\sigma\{\text{IQR of column}\}$	Data Distribution
Mean Q1	$\mu\{\text{Q1 of column}\}$	Data Distribution
Standard Deviation of Q1	$\sigma\{\text{Q1 of column}\}$	Data Distribution
Mean Q3	$\mu\{\text{Q3 of column}\}$	Data Distribution
Standard Deviation of Q3	$\sigma\{\text{Q3 of column}\}$	Data Distribution
Mean Percent of Z-Score Outliers	$\mu\left\{\frac{\text{\# of Z-Score Outliers}}{\lvert\mathbf{A}\rvert_r}\right\}$	Outlier Data Points
Standard Deviation of Percent of Z-Score Outliers	$\sigma\left\{\frac{\text{\# of Z-Score Outliers}}{\lvert\mathbf{A}\rvert_r}\right\}$	Outlier Data Points
Mean Correlation	$\mu\{\text{Correlation Matrix}\}$	Data linearity
Standard Deviation of Correlation	$\sigma\{\text{Correlation Matrix}\}$	Data Linearity

our search to tabular, binary-classification datasets, we were left with 268 datasets. We extracted the attributes in Table 2. from each dataset and ran every model on each dataset to get the performace of each model. We trained a multilabel regression model on these attributes to predict the rank of every model on a specific dataset.

2.3 Experiments

3 Discussion

Acknowledgments and Disclosure of Funding

References