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. We performed a literature review of existing AutoML papers and conducted a survey on six popluar AutoML frameworks. 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. Our framework, SME, demonstrates the ability to generate a model with near equal accuracy to a "dumb" model that runs every model on every dataset, but in a fraction 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

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

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

Preprint. Under review.

Table 1: Survey of Existing AutoML Frameworks

Framework	EOU	Algs	FE	TM	HPT	Inter	Custom
H2O AutoML	Easy	10	Basic	Yes	Auto	SHAP	High
TPOT	Hard	10	Basic	No	Genetic	POJO	Moderate
MLJAR	Easy	10	Advanced	Yes	Auto	Basic	High
FLAML	Easy	10	Limited	Yes	Auto	SHAP	Low-Moderate
LightAutoML	Easy	10	Advanced	Yes	Auto	Basic	Moderate
GAMA	Hard	10	Basic	Yes	Auto	SHAP	Moderate

Abbreviations: EOU: Ease of Use, Algs: Algorithms, FE: Feature Engineering, TM: Time Management, HPT: Hyper-Parameter Tuning, Inter: Interface, Custom: Customization

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.

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

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. These extracted attributes are shown in Table 2.

We considered 28 different models, including all supervised and semi-supervised sklearn classification models and common boosting models such as XGBoost, Lightgbm, and Catboost.

To implement this model elimination process, we trained a boosting model on 268 kaggle datasets. We choose datasets that were medically related and focused our approach to binary classification. We extracted the attributes in Table 2 from each dataset, this is the input for the boosting model. We then ran a "dumb" framework that trained every model on each dataset. The accuracy of every model is

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
STD of IQR	$\sigma\{\text{IQR of column}\}$	Data Distribution
Mean Q1	$\mu\{\text{Q1 of column}\}$	Data Distribution
STD of Q1	$\sigma\{\text{Q1 of column}\}$	Data Distribution
Mean Q3	$\mu\{\text{Q3 of column}\}$	Data Distribution
STD 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
STD 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
STD of Correlation	$\sigma\{\text{Correlation Matrix}\}$	Data Linearity

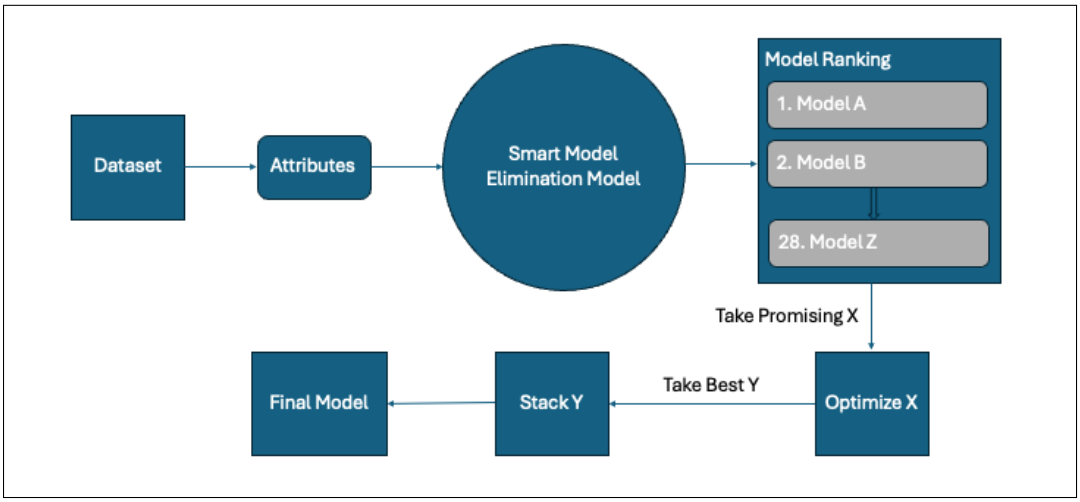

Figure 1: SMEML Process

ranked and every model is treated as a label with its value being its rank. Then we trained a multilabel regression to predict the rank of each model giving the extracted attributes. We produced a median spearman correlation of 0.7301 between the predicted rank and the actual rank and a median p-value of 1.032e-05. Furthermore, we guarantee that we can predict a top two model within the predicted top ten 90.12% of the time.

When a user inputs a dataset into SMEML, it first extracts the attributes from the dataset. It feeds these attributes into the boosting model to get the predicted rank of each model. Since we have determined that we can predict a top two model within the predicted top ten quite confidently we can eliminate the bottom 18 models from the list. We run bayseian optimization on the top ten models to get the best hyperparamters for each model. We select the top models that have significantly better performance than other models. These models are stacked together to form the final model. This process is shown in Figure 1.

Table 3: Experiment Results

Dataset	Accuracy		Time to Train (sec)		Size (RxC)
	SMEML	Dumb	SMEML	Dumb	
Chronic Kidney Disease Dataset	1.0	1.0	419.64	788.46	400x26
Brain Stroke Prediction Dataset	.9458	.9458	114.16	370.95	4892x11
Lung Cancer Prediction	.9839	.9839	143.07	317.04	309x16
Pima Indians Diabetes Dataset	.7857	.8052	161.99	374.01	768x8
Eye State Classification	.9536	.9549	312.57	696.51	14980x15

3 Experiments

3.1 Experiment Design

To evaluate our framework, we choose 7 different datasets of diverse size. We ran SMEML on each dataset for 20 iterations of tuning. We compared this against a "dumb" mode which simply runs every model on the dataset. We used the accuracy and time to train as our metrics.

All experiments are run on an Ubuntu server with Intel Xeon E5520 @ 2.27GHz and 24GB of RAM.

3.2 Results

4 Discussion

Acknowledgments and Disclosure of Funding

References