SmartML: A Meta Learning-Based Framework for Automated Selection and Hyperparameter Tuning for Machine Learning Algorithms

Mohamed Maher, Sherif Sakr
University of Tartu, Estonia
{mohamed.abdelrahman,sherif.sakr}@ut.ee

ABSTRACT

Due to the increasing success of machine learning techniques, nowadays, thay have been widely utilized in almost every domain such as financial applications, marketing, recommender systems and user behavior analytics, just to name a few. In practice, the machine learning model creation process is a highly iterative exploratory process. In particular, an effective machine learning modeling process requires solid knowledge and understanding of the different types of machine learning algorithms. In addition, all machine learning algorithms require user-defined inputs to achieve a balance between accuracy and generalizability. This task is referred to as *Hyperparameter Tuning*. Thus, in practice, data scientists work hard to find the best model or algorithm that meets the specifications of their problem. Such iterative and explorative nature of the modeling process is commonly tedious and time-consuming.

We demonstrate `SmartML`, a meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. Being meta learning-based, the framework is able to simulate the role of the machine learning expert. In particular, the framework is equipped with a continuously updated knowledge base that stores information about the meta-features of all processed datasets along with the associated performance of the different classifiers and their tuned parameters. Thus, for any new dataset, `SmartML` automatically extracts its meta features and searches its knowledge base for the best performing algorithm to start its optimization process. In addition, `SmartML` makes use of the new runs to continuously enrich its knowledge base to improve its performance and robustness for future runs. We will show how our approach outperforms the-state-of-the-art techniques in the domain of automated machine learning frameworks.

1 INTRODUCTION

Machine learning is the field of computer science that focuses on building algorithms that can automatically learn from data and automatically improve its performance without end-user instructions, influence or interference. In general, the effectiveness of machine learning techniques mainly rests on the availability of massive datasets, of that there can be no doubt. The more data that is available, the richer and the more robust the insights and the results that machine learning techniques can produce. Nowadays, we are witnessing a continuous growth in the size and availability of data in almost every aspects of our daily life. Thus, recently, we have been witnessing many leaps achieved by machine learning in wide range of fields [1, 9]. Consequently, there are growing demand to have increasing number of data scientists with strong knowledge and good experience with the various machine learning algorithms in order to be able to build models that can achieve the target performance and to keep up with exponential growing amounts of data which is produced daily.

In practice, the machine learning modeling process is a highly iterative exploratory process. In particular, there is no one-model-fits-all solution, i.e, there is no single model or algorithm which is well-known to achieve the highest accuracy for all data set varieties in a certain application domain. Hence, trying many machine learning algorithms with different parameter configurations is commonly considered an inefficient, tedious, and time consuming process. Therefore, there has been growing interest to automate the machine learning modeling process as it has been acknowledged that *data scientists do not scale*[1]. Therefore, recently, several frameworks have been designed to support automating the machine learning modeling process. For example `Auto-Weka` [8] is an automation framework for algorithm selection and hyper-parameter optimization which is based on Bayesian optimization using sequential model-based algorithm configuration (SMAC) and tree-structured parzen estimator (TPE). `Auto-Sklearn` [3] is a framework that has been implemented on top of the popular python `scikit-learn` machine learning package that automatically considers the past performance on similar datasets for its automation decision. Other tools include `Google Vizier` which is based on grid or random search [5] and `TPOT` which is based on genetic programming [10].

In this demonstration, we present `SmartML`, a meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms (using 15 classifiers). In our framework, the meta-learning feature is emulating the role of the domain expert in the field of machine learning [4, 11]. In particular, we exploit the knowledge and experience from previous runs by storing a set of data meta-features along with their performance. In addition, our knowledge base is continuously updated after running each task over `SmartML` which contributes to improving framework performance over the time. Our meta-learning mechanism is mainly used for the algorithm selection process in order to reduce the parameter-tuning search space which is conducted using SMAC Bayesian optimization [7]. This is different from other tools [3, 8] which

[1]https://hbr.org/2015/05/data-scientists-dont-scale

	SmartML	Auto-Weka	AutoSklearn	TPOT
Language	R	Java	Python	Python
API	Yes	No	No	Yes
Optimization Procedure	Bayesian Optimization (SMAC)	Bayesian Optimization (SMAC and TPE)	Bayesian Optimization (SMAC)	Genetic Programming, and Pareto Optimization
Number of Algorithms	15 classifiers on top of R	27 classifiers on top of WEKA	15 classifiers on top of scikit learn	15 classifiers on top of scikit learn
Support Ensembling	Yes	Yes	Yes	No
Use Meta-Learning	Yes (incrementally updated KB)	No	Yes (Static)	No
Feature preprocessing	Yes	Yes	Yes	No
Model Interpretability	Yes	No	No	No

Table 1: Comparison between State-of-the-art Automated Machine Learning Frameworks

deal with algorithm selection as one of the parameters to be tuned.

`SmartML` can be used as a package in `R language`, one of the most popular languages in the data science domain, or as a Web application[2]. It is also designed to be programming language agnostic so that it can be embedded in any programming language using its available REST APIs. Table 1 shows a feature comparison between our framework and other state-of-the-art frameworks. In our demonstration, we will show that `SmartML` can outperform other tools especially at small running time budgets by reaching better parameter configurations faster. In addition, `SmartML` has the advantage that its performance can be continuously improved over time by running more tasks which makes `SmartML` *smarter* by getting more experience based on the growing knowledge base.

2 SMARTML ARCHITECTURE

Figure 1 illustrates the framework architecture of `SmartML`. In the *input definition* phase, the user uploads the dataset, choose the required options for features selection and preprocessing, specify which features of the dataset should be included in the modeling process, specify the target column which represents the classes of labels of the instances in the dataset and specify the `time budget` constraint for the framework for conducting the hyper-parameter tuning process. `SmartML` accepts `csv` and `arff` (attribute relation file format developed with the Weka machine learning software) file formats.

In the *preprocessing* phase, `SmartML` starts by performing the feature preprocessing operations specified by the selected features. Table 2 lists the feature preprocessing operations supported by the `SmartML` framework. In this phase, the dataset is randomly split into *training* and *validation* partitions where the former is used in algorithm selection and hyper-parameter tuning while the later is used for evaluating the selected configurations during parameter tuning. In addition, a list of 25 meta-features are extracted from the training split describing the dataset characteristics. Examples of these features include *number of instances*, *number of classes*, *skewness* and *kurtosis of numerical features*, and *symbols of categorical features.*

Currently, `SmartML` supports 15 different classifiers (Table 3). In the *algorithm selection* phase, the meta features of the input dataset at hand, which is extracted during the *preprocessing* phase, are compared with the meta features of the datasets that are stored in the knowledge base in order to identify the *similar* datasets, using a nearest neighbor approach. The dataset similarity detection process follows a weighted mechanism between two different factors. The first factors is the *Euclidean distance* between the meta-features of the dataset at hand and meta-features of all datasets stored in the knowledge base. The second factor is the magnitude of the best performing algorithms on the similar dataset. For example, it may be better to select the top n top performing algorithms on a single very similar dataset than selecting the first outperforming algorithm for n similar datasets. We use the retrieved results of the best performing algorithms on similar dataset(s) to nominate the candidate algorithms for the dataset at hand.

In the *hyper-parameter tuning* phase, `SmartML` attempts to tune the selected classifiers hyper-parameters for achieving the best performance. In particular, the knowledge base contains information about the best parameter configurations for each algorithm on each dataset. The configurations of the nominated best performing algorithms are used to initialize the hyper-parameter tuning process for the selected algorithms. The time budget constraint specified by the end user represents the time used in hyper parameter tuning of the selected classifiers. In particular, this budget is divided among all the selected algorithms according to the number of hyper-parameters to tune in each algorithm (Table 3). `SmartML` applies the SMAC technique for hyper-parameter optimization [7]. In particular, SMAC attempts to draw the relation between the algorithm performance and a given set of hyper-parameters by estimating the predictive mean and variance of their performance along the trees of the random forest model. The main advantage of using SMAC is its robustness by having the ability to discard low performance parameter configurations quickly after the evaluation on low number of folds of the dataset [7].

Finally, the results obtained from the hyper-parameter tuning process of the different nominated algorithms are compared with each other to recommend the best performing algorithm to the end user. In addition, a weighted ensembling [2] output of the top performing algorithms can be recommended to the end user based on their choice. In addition, we have integrated the `Interpretable Machine Learning (iml)` package[3] in order to explain for the user the most important features that have been used by the

[2]https://bigdata.cs.ut.ee/smartml/index.html

[3]https://cran.r-project.org/web/packages/iml/index.html

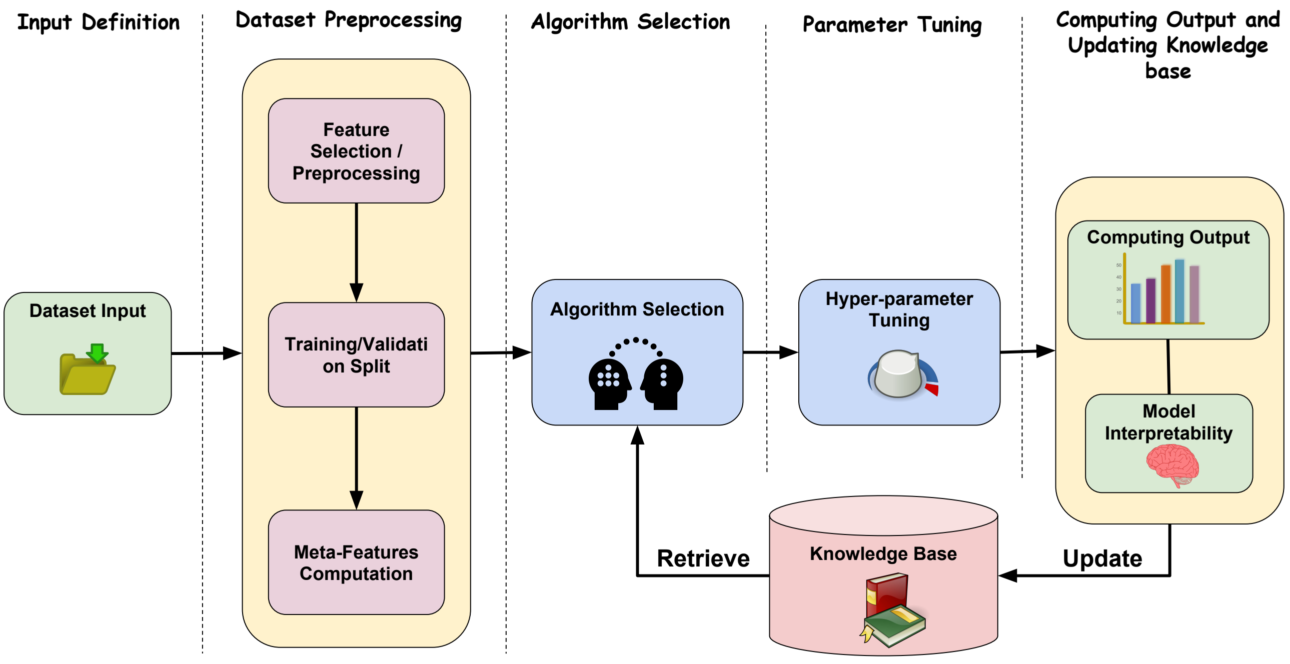


Figure 1: `SmartML` : Framework Architecture

center	subtract mean from values
scale	divide values by standard deviation
range	values normalization
zv	remove attributes with zero variance
boxcox	apply box-cox transform to non-zero positive values
yeojohnson	apply Yeo-Johnson transform to all values
pca	transform data to the principal components
ica	transform data to their independent components

Table 2: Integrated Feature Preprocessing Algorithms

Classification Algorithm	Categorical parameters	Numerical parameters	Package
SVM	1	4	e1071
NaiveBayes	0	2	klaR
KNN	0	1	FNN
Bagging	0	5	ipred
part	1	2	RWeka
J48	1	2	RWeka
RandomForest	0	3	randomForest
c50	3	2	C50
rpart	0	4	rpart
LDA	1	1	MASS
PLSDA	1	1	caret
LMT	0	1	RWeka
RDA	0	2	klaR
NeuralNet	0	1	nnet
DeepBoost	1	4	deepboost

Table 3: Integrated Classifier Algorithms

selected model for directing its prediction process [6]. The interactive interface of our system has been designed using the `Shiny` R Package[4].

[4]https://shiny.rstudio.com/

Dataset	# Att.	# Classes	# Instances	Auto-Weka Accuracy	SmartML Accuracy
abalone	9	2	8192	25.14	**27.13**
amazon	10000	49	1500	57.56	**58.89**
cifar10small	3072	10	20000	30.25	**37.02**
gisette	5000	2	2800	93.71	**96.48**
madelon	500	2	2600	55.64	**73.84**
mnist Basic	784	10	62000	89.72	**94.91**
semeion	256	10	1593	89.32	**94.13**
yeast	8	10	1484	51.80	**66.23**
Occupancy	5	2	20560	93.99	**95.55**
kin8nm	8	2	8192	93.99	**96.42**

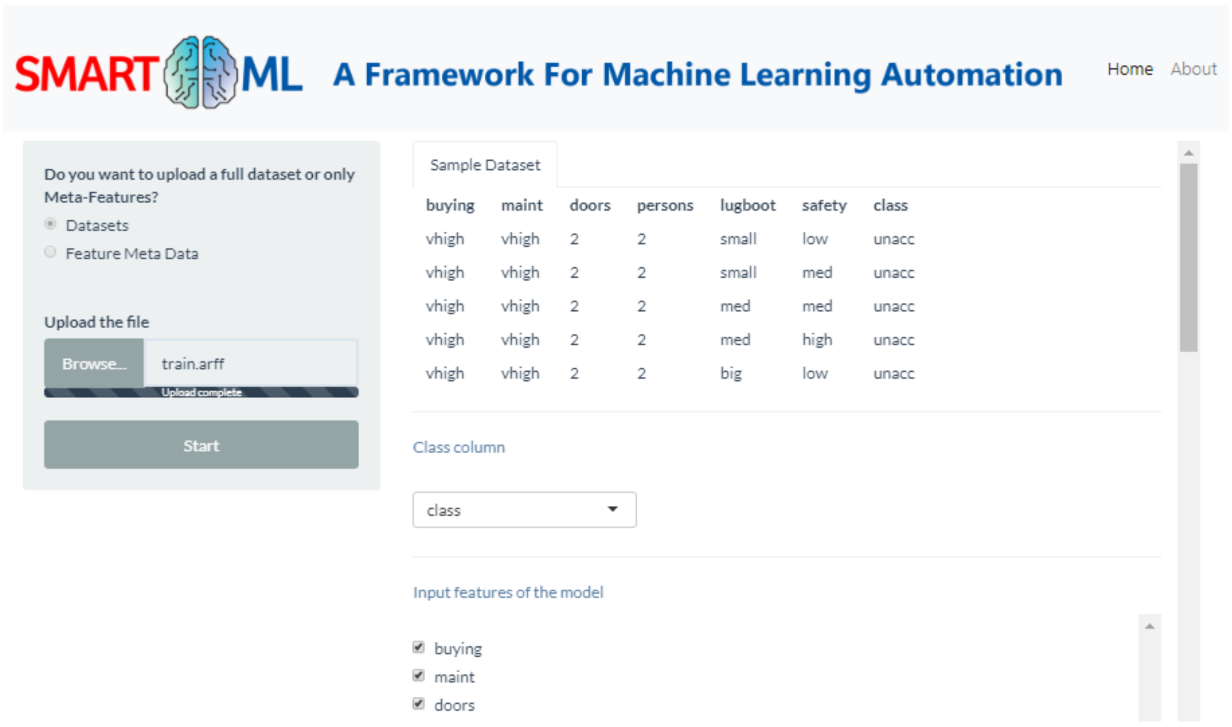
Table 4: Performance Comparison: `SmartML` VS `Auto-Weka`

3 DEMO SCENARIO

`SmartML` is available both as a Web application as well as RESTful APIs[5]. In this demonstration[6], we will present to the audience the workflow of the `SmartML` framework (Figure 1). In particular, we will show that how our approach can help non-expert machine learning users to effectively identify the machine learning algorithms and their associated hyperparameter settings that can achieve optimal or near-optimal accuracy for their datasets with little effort.

We start by introducing to the audience the challenges we tackle, the main goal and the functionalities of our framework. Then, we take the audience through the automated algorithm selection and hyper-parameter tuning process for sample datasets. We start by showing different features which is provided for the end-user (Figure 2). For example, the user can upload either a dataset file or a direct URL for the dataset. In addition, the user can choose either to perform both algorithm selection and hyper-parameter

[5]The source code of the `SmartML` framework is available on https://github.com/DataSystemsGroupUT/Auto-Machine-Learning
[6]A demonstration screencast is available on https://www.youtube.com/watch?v=m5sbV1P8oqU&feature=youtu.be



Do you want to upload a full dataset or only Meta-Features?
[✓ Datasets]
Feature Meta Data

Upload the file
train.arff
Upload complete

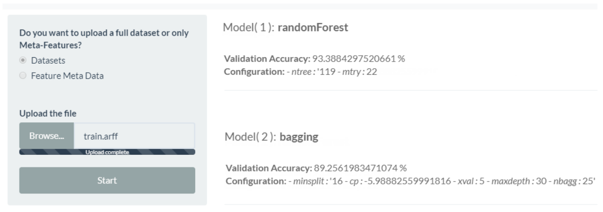
Sample Dataset

buying	maint	doors	persons	lugboot	safety	class
vhigh	vhigh	2	2	small	low	unacc
vhigh	vhigh	2	2	small	med	unacc
vhigh	vhigh	2	2	med	med	unacc
vhigh	vhigh	2	2	med	high	unacc
vhigh	vhigh	2	2	big	low	unacc

Class column: class

Input features of the model
- [x] buying
- [x] maint
- [x] doors

Figure 2: Screenshot: Configuring an experiment for a dataset


Do you want to upload a full dataset or only Meta-Features?
[✓ Datasets]
Feature Meta Data

Upload the file
train.arff
Upload complete

Model(1): **randomForest**

Validation Accuracy: 93.3884297520661 %
Configuration: - *ntree* : '119 - *mtry* : 22

Model(2): **bagging**

Validation Accuracy: 89.2561983471074 %
Configuration: - *minsplit* : '16 - *cp* : -5.98882559991816 - *xval* : 5 - *maxdepth* : 30 - *nbagg* : 25'

Figure 3: Screenshot: Sample experiment output from `SmartML`

tuning or only algorithm selection. In the later case, it is possible to upload only the dataset meta-features file instead of the whole dataset. The user will be also able to configure different options such as whether any kind of feature preprocessing is needed or not, whether model interpretability is needed or not and specify the time budget for hyper-parameter tuning. Then, we will take the audience through the different phases of the framework until returning the final results (Figure 1).

Table 4 shows the performance comparison between `SmartML` and `Auto-Weka`[7] using 10 datasets where a *time budget* of 10 minutes has been allocated for each dataset in each framework. In our experiments, we have bootstrapped the knowledge base of `SmartML` using 50 datasets from various sources including `OpenMl`[8], `UCI repository`[9] and `Kaggle`[10]. The results show that, using this relatively very small knowledge base, the accuracy results of `SmartML` outperform the results of `Auto-Weka` for all the datasets. As a part of our demonstration, we will provide the audience with the live chance to compare the performance of `SmartML` with `Auto-Weka` and other related frameworks using various datasets.

[7]https://www.cs.ubc.ca/labs/beta/Projects/autoweka/
[8]https://www.openml.org/
[9]http://archive.ics.uci.edu/ml/index.php
[10]Kaggle:https://www.kaggle.com/

ACKNOWLEDGMENT

This work is funded by the European Regional Development Funds via the Mobilitas Plus programme (grant MOBTT75)

REFERENCES

[1] Rahul C Deo. 2015. Machine learning in medicine. *Circulation* 132, 20 (2015), 1920–1930.
[2] Thomas G Dietterich. 2000. Ensemble methods in machine learning. In *International workshop on multiple classifier systems.*
[3] Matthias Feurer et al. 2015. Efficient and Robust Automated Machine Learning. In *NIPS.*
[4] Matthias Feurer et al. 2015. Initializing Bayesian Hyperparameter Optimization via Meta-learning. In *AAAI.*
[5] Daniel Golovin et al. 2017. Google Vizier: A Service for Black-Box Optimization. In *KDD.*
[6] Riccardo Guidotti et al. 2018. A survey of methods for explaining black box models. *ACM CSUR* 51, 5 (2018).
[7] Frank Hutter et al. 2011. Sequential Model-based Optimization for General Algorithm Configuration. In *LION.*
[8] Lars Kotthoff et al. 2017. Auto-WEKA 2.0: Automatic Model Selection and Hyperparameter Optimization in WEKA. *J. Mach. Learn. Res.* 18, 1 (2017).
[9] Sendhil Mullainathan and Jann Spiess. 2017. Machine learning: an applied econometric approach. *Journal of Economic Perspectives* 31, 2 (2017).
[10] Randal S. Olson and Jason H. Moore. 2016. TPOT: A Tree-based Pipeline Optimization Tool for Automating Machine Learning. In *Proceedings of the Workshop on Automatic Machine Learning.*
[11] Matthias Reif et al. 2012. Meta-learning for Evolutionary Parameter Optimization of Classifiers. *Mach. Learn.* 87, 3 (2012).