OptimalFlow: Omni-ensemble and Scalable Automated Machine Learning

2020 GVector Augmented Intelligence Conference Paper

Lei (Tony) Dong[*]
Data Scientist at Genpact
Lei.Dong@Genpact.com

Khader Syed
AVP at Genpact
Khader.Syed@Genpact.com

Abstract

OptimalFlow is an omni-ensemble and scalable automated machine learning Python toolkit, which is using pipeline cluster traversal experiment(PCTE) and selection-based feature preprocessor with ensemble encoding(SPEE), to help data scientists building optimal models in an easy way, and automate supervised learning workflow with simple codes. OptimalFlow is designed with high-level APIs targeting to avoid manual repetitive train-along-evaluate experiments in general pipeline building. It rebuilt the automated machine learning framework by switching the focus from single pipeline components automation to a higher workflow level by creating an automated pipeline ensemble and evaluation mechanism. By modularizing functional modules in reusable APIs, OptimalFlow makes each supervised learning step to be custom tunable along with high scalability.

Keywords Automated Machine Learning · Pipeline Optimization · Feature Preprocessor · Ensemble Encoder

1 Introduction

Growing demand for hands-free machine learning systems has given rise to the new area of automated machine learning(AutoML [1]). The essence of AutoML is to automate repetitive tasks such as pipeline creation and hyperparameter tuning so that data scientists can spend more time on business problems on hand in practical scenarios. AutoML also allows everyone instead of a small group of people to use machine learning technology. Data scientists can accelerate ML development by using AutoML to implement really efficient machine learning.

AutoML platforms provide value to businesses that already have in-house data science teams and allow them to focus on more complex processes such as model construction without spending time on time-consuming processes such as feature engineering and hyperparameter optimization[2]. Whether AutoML will be a success depends on its usage and progress in the machine learning field. Obviously, AutoML is an important part of machine learning in the future.

To keep a live business-dependent machine learning workflow productive and competitive, and effective ensemble, a pipeline ensemble mechanism would be one way to automate the process. This paper proposes a new solution called ***OptimalFlow***[3] to address this challenge. OptimalFlow is an Omni-ensemble Automated Machine Learning Python toolkit, which is based on the Pipeline Cluster Traversal Experiment approach, to help data scientists build optimal models in an easy way, and automate Supervised Learning workflow with simple codes.

2 Pipeline Cluster Traversal Experiments(PCTE)

The common machine learning workflow is automated by a "single pipeline" strategy, which is first introduced and well-supported by the Scikit-learn[4] library. When it comes to practical use, data scientists need to implement repetitive experiments in each component within one pipeline, adjust algorithms and parameters, to get the optimal baseline model. We call this operation mechanism "Single Pipeline Repetitive Experiments", see Figure 1. No matter classic

[*] Data Scientist at Genpact since 2019, with BE in Robotics and MBA in Business Analytics; Research Domain includes Healthcare and Pharmaceutical Industries.

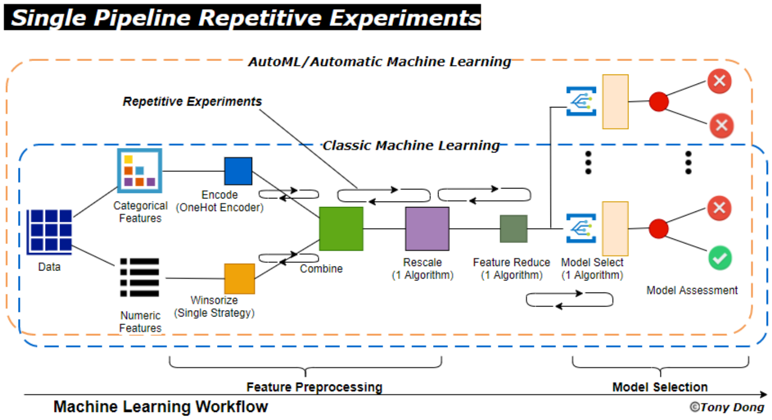


Figure 1: Single Pipeline Repetitive Experiments

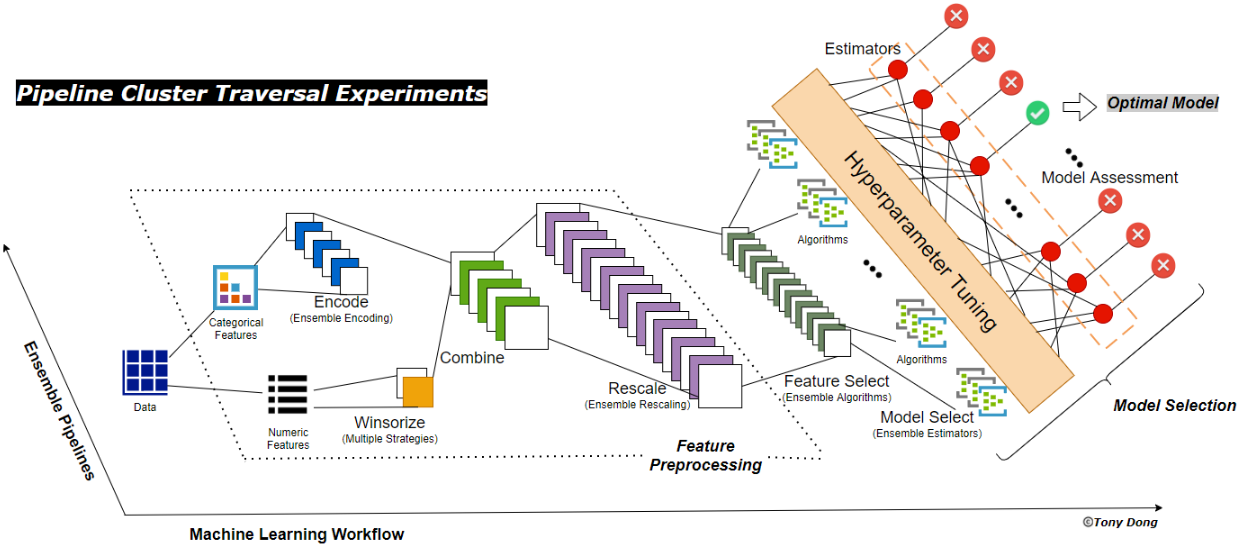

Pipeline Cluster Traversal Experiments
Estimators
Optimal Model
Model Assessment
Hyperparameter Tuning
Algorithms
Algorithms
Ensemble Pipelines
Data
Categorical Features
Encode (Ensemble Encoding)
Numeric Features
Winsorize (Multiple Strategies)
Combine
Rescale (Ensemble Rescaling)
Feature Select (Ensemble Algorithms)
Model Select (Ensemble Estimators)
Feature Preprocessing
Model Selection
Machine Learning Workflow
©Tony Dong

Figure 2: Pipeline Cluster Traversal Experiments

machine learning or current popular AutoML libraries, it's hard to avoid this single pipeline focused experiment, which is the biggest pain point in the supervised modeling process.

Pipeline Cluster Traversal Experiments(PCTE) is a greedy optimization method to simulate standard data scientist workflow in supervised learning processes. It uses Deep Belief Networks(DBN)[5] model structure, stacked regularized feature preprocessing, feature selection, as well as model selection and hyperparameter tuning. See Figure 2.

PCTE creates pipeline clusters by building a cross-matching pipeline ensemble, covering major components of machine learning workflow, and apply Depth-first-search(DFS) traversal[6] experiment to search the optimal baseline model. Besides, modularizing all key pipeline components in reusable packages, it allows all components to be custom updated along with high scalability.

Comparing other AutoMLs or classic machine learning workflow's repetitive experiments using a single pipeline, PCTE is more powerful, since it expends the workflow from 1 dimension to 2 dimensions by adding all qualified pipelines to the ensemble (Pipeline Cluster) and automated experiments. With a larger searching scope, to find the optimal model without manual intervention, and also more flexible with elasticity to cope with unseen data due to its ensemble designs in each component, the PCTE provides data scientists an alternative more convenient and "Omni-automated" machine learning approach.

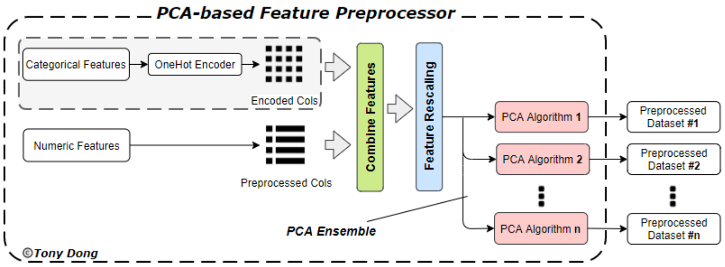


Figure 3: Typical PCA-based feature preprocessor.

3 Selection-based Feature Preprocessor with Ensemble Encoding(SPEE)

The performance of an automated machine learning workflow depends on how we process and feed different types of variables to the model, due to most machine learning models only accept numerical variables. Thus, categorical features encoding becomes a necessary step for any automated machine learning approaches. It not only elevates the model quality but also helps in better feature engineering.

There are two major feature reduction strategies: principal component analysis(PCA[7]) and feature selection.

PCA is widely used in current AutoML frameworks, due to it often used for reducing the dimensionality of a large data set so that it becomes more practical to apply machine learning where the original data are inherently high dimensional. It relies on linear relationships between feature elements and it's often unclear what the relationships, as it also "hides" feature elements that contribute little to the variance in the data, it can sometimes eradicate a small but significant differentiator that would affect the performance of a machine learning model[8].

The withdraw of PCA is more apparent when AutoML system coping with categorical features. Most AutoML frameworks are using the OneHot algorithm, which will easily generate high dimension dummies features when a categorical feature has high cardinality. That will cause information loss and hard to tune without manual diagnosis and interruption. Typical PCA-based feature preprocessor uses only one encoder to cope with categorical features and has at least one PCA algorithm to implement feature reduction. This preprocessor system is widely applied in AutoML frameworks, i.e. Auto-ML and H2O autoML[9], and Auto-Sklearn [10] has a PCA ensemble component in it, which allows multiple PCA algorithms to generate input datasets for different pipelines. Figure 3 above represents a typical PCA-based feature prerocessor, applied in most AutoML frameworks.

To design a robust automated feature preprocessor, we chose feature selection as the alternative strategy, and an ensemble encoding mechanism is introduced to cope with categorical features in the input data. This Selection-based Feature Preprocessor with Ensemble Encoding(SPEE) system improves AutoML's adaptivity for multiple categorical features by ensemble encoding method. Instead of PCA algorithms, it keeps features relationship information and variance by ensemble feature selection algorithms. See Figure 4.

For the feature selection ensemble component, SPEE is using 3 major algorithms:

- Select K Best Features(SelectKBest) with ANOVA F-value between label/feature or Chi-squared stats of non-negative features approach, etc.
- Recursive Feature Elimination(RFE) with Logistic Regression, SVM, or Decision Tree estimator
- Recursive Feature Elimination and Cross-validated Selection(RFECV) with logistic regression, SVM, or decision tree estimator

The Figure 5 below show that SPEE performed better to cope with multiple categorical features in perceiving the trend and pattern, which will also be beneficial for further AutoML workflow. Here's the performance compare plot based on the categorical features added *breast cancer* dataset.

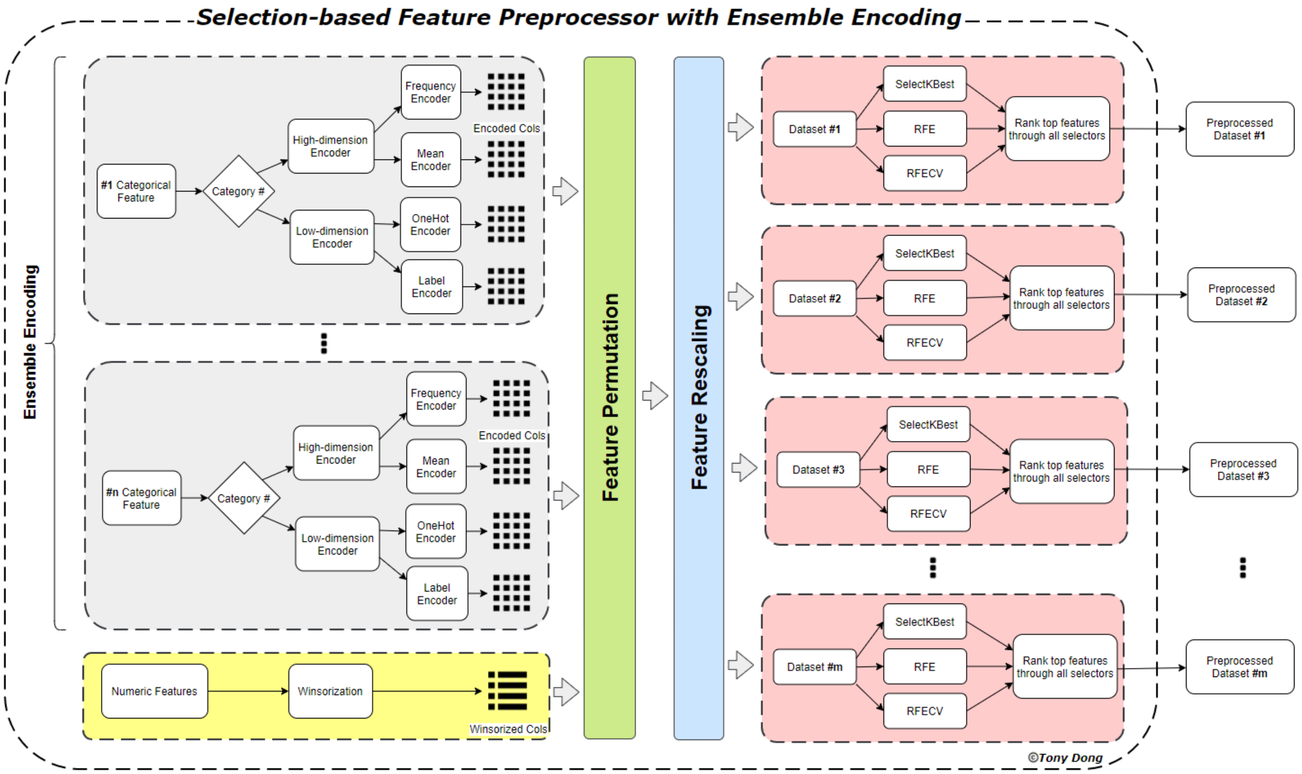


Figure 4: Selection-based feature preprocessor with ensemble encoding.

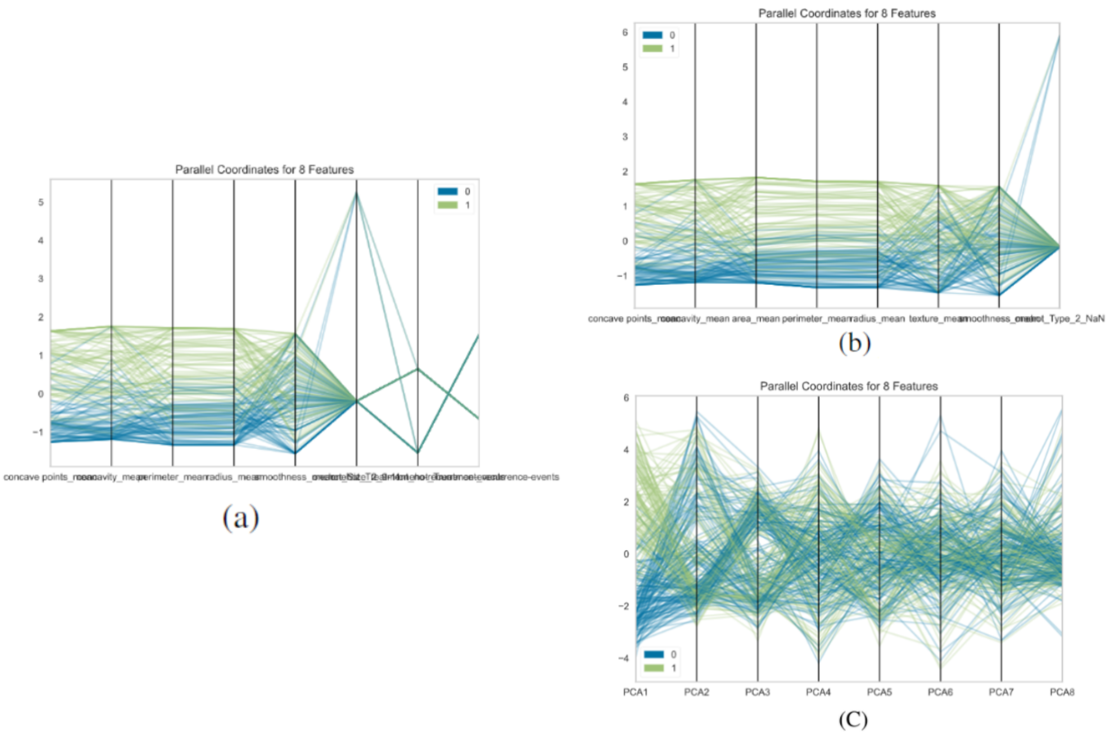


Figure 5: Preprocessor performance compare, based on categorical features added *breast cancer* dataset.. (a) Non-ensemble preprocessing (OneHot encoding + SelectKBest feature selection); (b) SPEE preprocessor; (c) PCA-based preprocessor.

4 An omni-ensemble and scalable automated machine learning system

4.1 OptimalFlow

OptimalFlow wraps the Scikit-learn supervised learning framework to automatically create an ensemble of machine learning pipelines based on algorithms permutation in each framework component. It includes feature engineering methods in its preprocessing module such as missing value imputation, categorical feature encoding, numeric feature standardization, and outlier winsorization.

OptimalFlow uses *Pipeline Cluster Traversal Experiments*(PCTE) as the optimizer and Selection-based feature preprocessor with ensemble encoding(SPEE) to build an omni-ensemble workflow for the optimal baseline model searching, including feature preprocessing/selection optimization, hyperparameters tuning, model selection, and assessment(See Table1-4). OptimalFlow wraps the algorithms updating operations in an ensemble way and modularizes each component(See Figure 6). This makes it easy for data scientists to implement iterated experiments across all ensemble components in the pipeline.

Another advantage of OptimalFlow is scalability. It uses ensemble feature and model selection technique for data to be iteratively trained across all selected models. By design the existing machine learning code pipeline is re-trained and OptimalFlow's package is updated by extending the new selector/estimator's algorithm or capacity thus making it scalable and extendable.

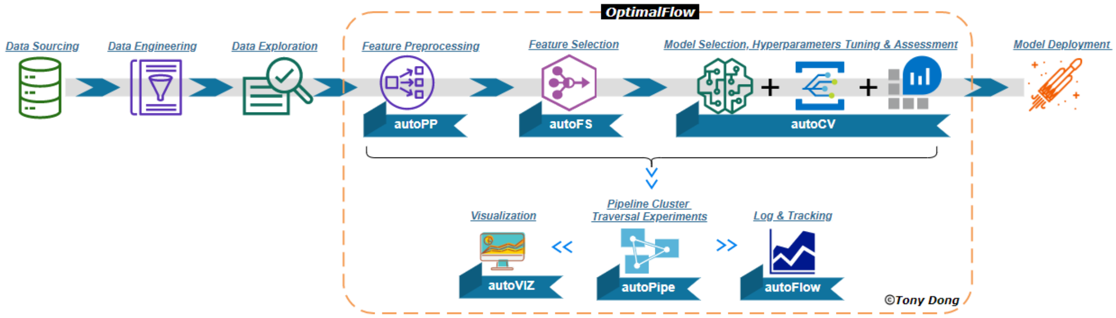


Figure 6: Standardized workflow for AutoML(Box with rounded edges denotes OptimalFlow core modules).

Comparing other popular AutoML frameworks or APIs, OptimalFlow is designed to avoid manual repetitive train-along-evaluate experiments in general supervised pipeline building. It rebuilt the automated machine learning framework by jumping out of a single pipeline's scope, while treats the whole pipeline as an entity, and automates the production of all possible pipelines for assessment, until finding one of them leads to the optimal model. Thus, when we say a pipeline represents an automated workflow, OptimalFlow is designed to assemble all these workflows, and find the optimal one. Using OptimalFlow, data scientists, including experienced users or beginners, can build optimal models easily without tedious experiments and pay more attention to convert their industry domain knowledge to the deployment phase with practical implementation.

Table 1: Classification model selection algorithms and search dimensions

Name	$\#\lambda$	Cat	Cont
Logistic Regression	2	1	1
C-Support Vector	2	2	0
Multi-layer Perceptron	5	4	1
AdaBoost	3	2	1
Random Forest	3	2	1
Gradient Boost	4	3	1
XGBoost	4	3	1
Linear Support Vector Machine	1	1	0
Hist Gradient Boost	2	2	0
Stochastic Gradient Descent	3	3	0
Ridge Cross-validation	1	1	0

Table 2: Regression model selection algorithms and search dimensions

Name	$\#\lambda$	Cat	Cont
Linear Regression	2	1	1
K-nearest Neighbors	3	3	0
Random Forest	2	2	0
AdaBoost	4	3	1
Gradient Boost	3	3	0
Decision Tree	4	3	1
Multi-layer Perceptron	5	4	1
XGBoost	4	3	1
Hist Gradient Boost	2	2	0
Huber	1	1	0
Ridge Cross-validation	1	1	0
Lasso Cross-validation	1	1	0
Stochastic Gradient Descent	3	3	0

In summary, OptimalFlow shares a few useful properties for data scientists:

- Easy coding: High-level APIs to implement PCTE, and each machine learning component is highly automated and modularized.
- Easy transformation: Focus on process automation for local implementation, which is easy to transfer current operation and meet compliance restrictions, i.e. pharmaceutical compliance policy.
- Easy maintaining: Wrap machine learning components to well-modularized code packages without miscellaneous parameters inside.
- Light and swift: Easy to transplant among projects. Faster, along with convenience, to deploy comparing other cloud-based solutions.
- Omni-ensemble: Easy for data scientists to implement iterated experiments across all ensemble components in the workflow.
- High Scalability: Each module could add new algorithms easily due to its ensemble and reusable coding design. PCTE and SPEE make it easier to adapt unseen datasets with the right pipeline.
- Customization: Support custom settings to add/remove algorithms or modify hyperparameters for elastic requirements.

Table 3: Feature preprocessing algorithms

Name	Core approach
Imputation	Mean strategy Frequency strategy
Encoding	OneHotEncoder LabelEncoder Frequency calculation Mean calculation
Winsorization	1% sample removal 5% sample removal
Rescaling	StandardScaler MinMaxScaler MaxAbsScaler RobustScaler

Table 4: Feature selection algorithms

Name	Core approach
SelectKBest	ANOVA F-value Chi-squared stats
RFE	Logistic Regression SVM Decision Tree
RFECV	Logistic Regression SVM Decision Tree

4.2 Performance improvement

OptimalFlow uses PCTE to build automated supervised learning workflow, which gets better performance comparing the typical machine learning pipeline approach. Here's the compare plot(Figure 7) for a classification problem based on the categorical features added *breast cancer* dataset. A learning curve shows a measure of predictive performance on a given domain as a function of some measure of varying amounts of learning effort [11]. It's clear to show in the learning curve, that model built by OptimalFlow has a better generalization behavior of machine learning, comparing other typical models based on the standard manual pipeline building approach. And OptimalFlow's model also shows the best accurate scores with good efficiency in learning performance over time in terms of experience.

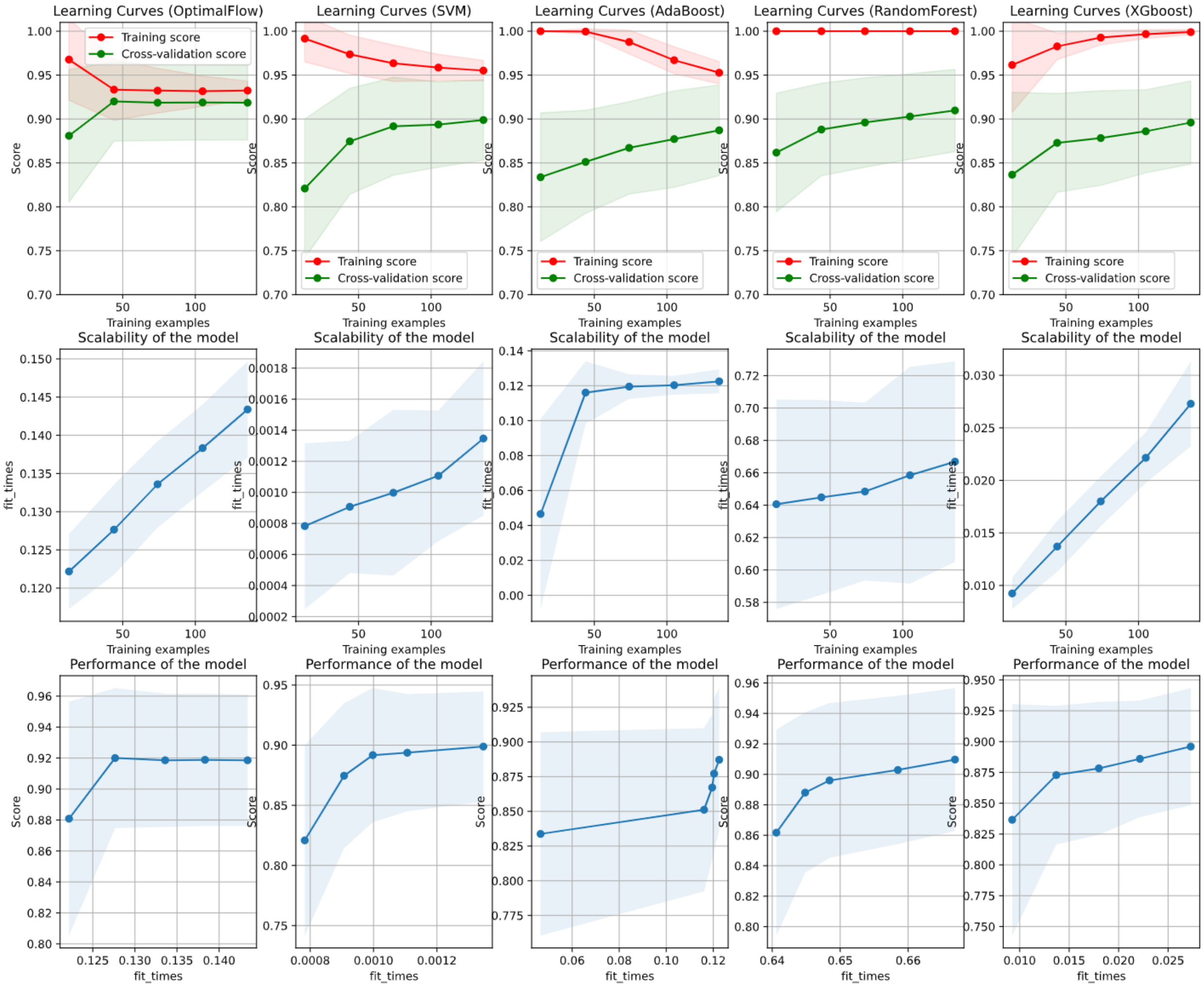


Figure 7: Learning curve, scalability and performance of the models.

4.3 Scalability of OptimalFlow

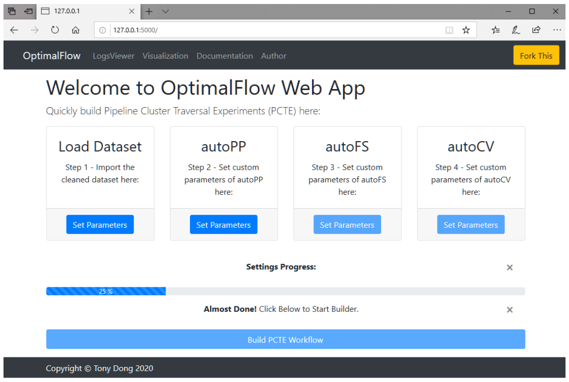
OptimalFlow builds a computing ensemble in each workflow component, including feature preprocessing, feature selection, and model selection, which makes it performance more adaptively to unseen data inputs. Plus, as previous comments about PCTE, comparing single pipeline experiment framework, PCTE builts pipelines for optimal selection, which apparently improves the searching coverage, and flexibility to various data scenarios.

The OptimalFlow's models inherit algorithms from Scikit-learn and XGBoost[12] estimators for classification and regression problems. And the extendable coding structure, based on highly modularized programming, supports adding models from external Python estimator libraries, which distinct OptimalFlow's scalability out of most of AutoML toolkits.

OpitmalFlow supports two methods of hyperparameter optimization; cross-validated cartesian grid search and random grid search[4]. Users can select one of them, based on the complexity of the model, to exhaustive or randomly search over a parameter grid with cross-validation. One simple, but a recent step toward formalizing hyper-parameter optimization is the use of random search [13]. [14] showed that random search was much more efficient than a grid search for optimizing the parameters of one-layer neural network classifiers[15].

In practical use, OptimalFlow's modules can be connected as a sequential workflow, but each module also can be accessible to use independently[3]. And the end and intermediate outcomes can be saved and reloaded into the OptimalFlow framework to be used in other custom processes.

Besides, OptimalFlow provides interfaces to makes it easy to continued development, and users could build applications on them. In the latest version(0.1.10), it added a Web App demo based on the flask framework, as an OptimalFlow's GUI. Users could build AutoML workflow all by clicks, without any coding. See Figure 8-9.



Welcome to OptimalFlow Web App

Quickly build Pipeline Cluster Traversal Experiments (PCTE) here:

Load Dataset

Step 1 - Import the cleaned dataset here:

autoPP

Step 2 - Set custom parameters of autoPP here:

autoFS

Step 3 - Set custom parameters of autoFS here:

autoCV

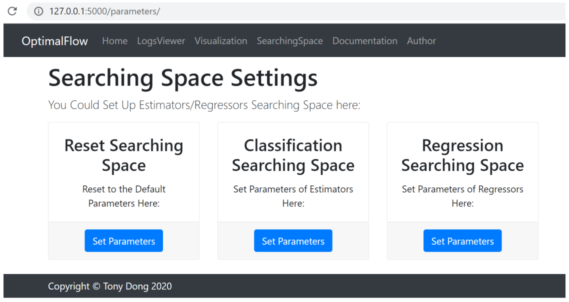
Step 4 - Set custom parameters of autoCV here:

Settings Progress: ×

Almost Done! Click Below to Start Builder. ×

Copyright © Tony Dong 2020

Figure 8: No-code Web Application built on OptimalFlow(Home page).



Searching Space Settings

You Could Set Up Estimators/Regressors Searching Space here:

Reset Searching Space

Reset to the Default Parameters Here:

Classification Searching Space

Set Parameters of Estimators Here:

Regression Searching Space

Set Parameters of Regressors Here:

Copyright © Tony Dong 2020

Figure 9: No-code Web Application built on OptimalFlow(Customization parameters page).

4.4 Comparing OptimalFlow to Auto-Sklearn and H2O autoML

To meet the demand for accessible, scalable, and automated machine learning(AutoML) platforms, multiple AutoML frameworks have been developed to dive into data for business value as quickly and with as little effort from human intervention as possible. AutoML platform is designed to standardize the machine learning workflow and automate components with tasks associated with machine learning pipelines.

We perform a benchmarking of AutoML by comparing OptimalFlow to Auto-Sklearn and H2O autoML, which are well-known AutoML frameworks claiming to produce the most valuable outputs with the least amount of effort. See Table 5.

Table 5: Comparing OptimalFlow to Auto-Sklearn and H2O AutoML

Metrics	Items	OptimalFlow	Auto-Sklearn	H2O AutoML
Operability	Written Language	Python	Python/C++	Java
	Operating Systems	Windows/Mac/Linux	Linux	Windows/Mac/Linux
	Parameters Settings	Simple	Miscellaneous	Miscellaneous
	User Interface	✓	-	✓
	Components Modularized	✓	-	✓
Functionality	Pipeline Ensemble	PCTE	CASH[2]	-
	Automated Preprocessing	✓	✓	-
	Categorical Engineering	Ensemble Encodering	OneHot	OneHot
	Features Reduction	Feature Selection	PCA	PCA
	Deep Learning	MLP[16]	-	MLP
	Support XGBoost	✓	-	✓
	Support Big Data	-	-	✓
Scalability	Custom Scalability	✓	✓	-
	Scikit-Learn Adaptivity	✓	✓	-
	Pandas Adaptivity	✓	✓	Need conversion
Efficiency	Computing System	Sequential	Sequential/Parallel	Distributed
	Resource Usage	Low	Low	High
	Hyperparameter Optimization	Grid/Random Search[4]	Bayesian Search	Random Search[14]

This comparison is based on OptimalFlow(0.1.11), Auto-Sklearn(0.10), and H2O AutoML(3.32.1).

To build a robust AutoML system, both OptimalFlow and auto-sklearn frameworks chose scikit-learn, one of the best known and most widely used machine learning libraries. It offers a wide range of well established and efficiently-implemented ML algorithms and is easy to use for both experts and beginners. Similar to OptimalFlow's PCTE workflow, auto-sklearn also uses ensemble selection. A greedy method that adds individual models iteratively to the ensemble if and only if they increase the validation performance. Its core approach is called "Combined Algorithm Selection and Hyperparameter optimization“ (CASH)[2].

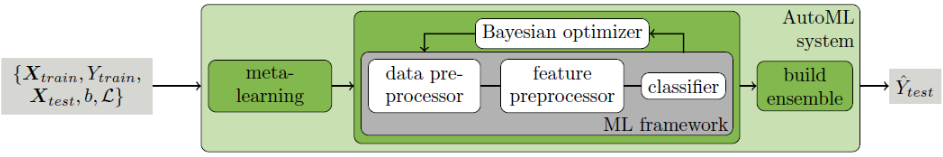


Figure 10: Auto-Sklearn's AutoML approach - CASH workflow.

Comparatively, OptimalFlow is an improved approach to modularize components with non-miscellaneous setting parameters, which makes each module package flexible for practical usage. By importing the SPEE feature preprocessing method to better handle categorical features with ensemble encoding and feature selection. And PCTE builder secures all possible pipelines to be added to the omni-ensemble pipeline cluster. Rather than using time and resource as the restiction, OptimalFlow's PCTE approach is using sparsity and column number as the constraints, which expands the optimal model's searching range to get better performance. In addition, OptimalFlow's extentable interfaces with visualization supports helps beginners to implement automated machine learning easily. For compatibility, OptimalFlow supports universal operating systems, including Windows, Mac and Linux; while Auto-Sklearn can be only installed in Linux OS [17], which not mapping well with general business computational environments. Besides, Auto-Sklearn has not yet tackled regression and semi-supervised problems[18].

When it comes to H2O autoML, it was alternatively built on Java and applied distributed cumputing system, which makes it more efficient to handel larger size of datsets. And comparing Auto-Sklearn and OptimalFlow, H2O autoML doesn't support pipeline ensemble, thus it still need manual intervention and experiments for feature preprocessing steps. Similar to Auto-Sklearn, H2O autoML is using PCA-based feature preprocessing approach, which has poor performance in coping with categorical features. Backed with H2O AI platform and derivative products, H2O autoML can be integrated and deployment in various machine learning production lines.

Auto-Sklearn's based on Bayesian search [19] for hyperparameter optimization, while it is unclear for practical problems what an appropriate choice is for the covariance function and its associated hyperparameters. Second, as the function evaluation itself may involve a time-consuming optimization procedure, problems may vary significantly in duration and this should be taken into account[20]. On the contrary, H2O autoML and OptimalFlow are using Random Search strategy to optimize hyperparameters, which is proved more efficient in practical operation and well mapping on business requirements.

5 Conclusion and Discussion

Therefore, we believe that OptimalFlow is a promising system for use by both machine learning beginners and experts. The source code of OptimalFlow is available under an open-source license at `https://github.com/tonyleidong/OptimalFlow`. And we benchmarked omni-ensemble OptimalFlow with PCTE and SPEE performance favorably comparing other edged AutoML frameworks. OptimalFlow is demonstrated in practical considerations as a new and innovative solution to address challenges in coping with categorical features and accessible usage for AutoML business users.

In the future work, we plan to apply OptimalFlow's methods to modern deep learning systems that yield performance on big data, include state-of-art estimators/regressors to the algorithm ensemble construction, and import parallel computing method with multiple optimized searching approaches, including Gaussian-process-based Bayesian optimization(BO [21]) [20], for hyperparameter tuning in PCTE workflow for better efficiency performance.

References

[1] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds. Automatic Machine Learning: Methods, Systems, Challenges. *ser. SSCML*, Springer 2019.

[2] Adithya Balaji and Alexander Alle. Benchmarking Automatic Machine Learning Frameworks. *arXiv preprint arXiv: 1808.06492*. 2018.

[3] Lei (Tony) Dong. OptimalFlow GitHub. In *https://github.com/tonyleidong/OptimalFlow*. Accessed: 2020-10-30.

[4] Scikit-learn GitHub. In *https://github.com/scikit-learn/scikit-learn*. Accessed: 2020-10-30.

[5] Geoffrey E. Hinton. Deep belief networks. In *Scholarpedia, 4(5)*, page 5947. 2009.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. In *Introduction to Algorithms, Third Edition (3rd ed.)*. MIT 2009.

[7] Jake Lever, Martin Krzywinski, and Naomi Altman. Principal component analysis. In *Nat Methods 14*, page 641–642. Nature Methods 2010.

[8] M. Pechenizkiy, A. Tsymbal, and S. Puuronen. PCA-based Feature Transformation for Classification: Issues in Medical Diagnostics. *Proceedings. 17th IEEE Symposium on Computer-Based Medical Systems*, page 535-540. IEEE 2004.

[9] H2o.ai automl GitHub. In *https://github.com/h2oai/h2o-3*. Accessed: 2020-10-16.

[10] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-Sklearn 2.0: The Next Generation. In *arXiv: 2007.04074*. 2020.

[11] Claudia Perlich. Learning Curves in Machine Learning. In *SEncyclopedia of Machine Learning*, page 577–580. Springer 2010.

[12] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM 2016.

[13] Nicolas Pinto, David Doukhan, James J. DiCarlo, and David D. Cox. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. *PLoS Comput Biol, 5(11):e1000579*. 2009.

[14] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *The Learning Workshop (Snowbird)*. 2011.

[15] James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems 25*, pages 2546–2554. Curran Associates, Inc., 2011.

[16] Popescu Marius, Valentina Emilia Balas, Perescu-Popescu Liliana, and Nikos E Mastorakis. Multilayer perceptron and neural networks. In *WSEAS Transactions on Circuits and Systems 8*. 2009.

[17] Auto-Sklearn GitHub. In *https://github.com/automl/auto-sklearn*. Accessed: 2020-10-29.

[18] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc. 2015.

[19] E. Brochu, V. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. In *CoRR, abs/1012.2599*. 2010.

[20] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. *arXiv 1206.2944*. 2012.

[21] B. Shahriari, K. Swersky, Z. Wang, R. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. In *Proceedings of the IEEE, vol. 104, no. 1*, pages 148–175. IEEE 2016.