Supplementary Material of Meta-data Augmentation based Search Strategy through Generative Adversarial Network for AutoML Model Selection

1 Meta-features in MDASS

We collected a set of meta-features for each OpenML dataset, which are listed in 1. They are divided into five groups: simple, statistical, information-theoretic, complexity and landmarkers.

2 The hyperparameter optimization space in MDASS

We used 13 algorithms from scikit-learn. The hyperparameter optimization space of each algorithm is listed in Table 2.

3 Base learner setting for SAM

SAM uses average accuracy/RMSE on held-out training set using 10-fold cross-validation to determine which base learner is set from aforementioned learners. Figure 1 visualizes the process of base learner setting for accuracy-oriented meta-model. It presents the performance of base learners for each algorithm prediction in one of 10 hold-out-validation. Since accuracy-oriented meta-model is viewed as multi-output classification problem, we maximize the average accuracy of each base learner, and select the one with highest average accuracy for each algorithm selection. From Figure 1, we observe that the commonly used learner, KNN, is not the optimal for each algorithm selection consistently, e.g. KNN learner has the highest average accuracy on the prediction of passive aggressive algorithm while the penultimate lowest average accuracy on the prediction of linear SVM algorithm, so are the other learners. Therefore, it is greatly necessary to build appropriate learner for adapting different algorithm prediction to reach global optimal.

Similarly, we test the above different categories of learners for runtime-oriented meta-model. Since it is viewed as multi-output regression problem, we minimize the RMSE of each base learner, and select the one with lowest RMSE for each algorithm runtime prediction. Table 3 also shows the necessity of the SAM, since there exists no such thing as a universal learner outperforming other learners on all algorithm runtime prediction consistently. For example, the minimum RMSE of runtime prediction of Gaussian naive Bayes algorithm is 0.342 using Ridge learner, while decision tree algorithm is 0.874 using RF learner and SGD algorithm is 1.296 using SVR learner.

Table 1. The implemented meta-features in MDASS.

Category	No.	Meta-features
Simple Meta-features	1	number of instances
	2	log number of instances
	3	number of features
	4	log number of features
	5	dataset ratio
	6	log dataset ratio
	7	inverse dataset ratio
	8	inverse log dataset ratio
	9	number of classes
	10	number of outliers
	11	percent of outliers
	12	number of numerical features
	13	number of categorical features
	14	ratio numerical to categorical
	15	ratio categorical to numerical
Statistical Meta-features	16	kurtosis max
	17	kurtosis min
	18	kurtosis mean
	19	kurtosis std
	20	skewness max
	21	skewness min
	22	skewness mean
	23	skewness std
	24	class probability min
	25	class probability max
	26	class probability mean
	27	class probability std
	28	symbols min
	29	symbols max
	30	symbols mean
	31	symbols std
	32	symbols sum
	33	PCA 95%
	34	PCA kurtosis first pcn
	35	PCA skewness first pc
	36	correlation min
	37	correlation max
	38	correlation mean
	39	correlation std
	40	correlation of variation
Information Meta-features	41	class entropy
	42	noise signal ratio
Complexity Meta-feature	43	overlap volume
Landmarker Meta-features	44	landmarkerLDA
	45	landmarkerNaiveBayes
	46	landmarkerDecisionTree
	47	landmarkerRondomNode
	48	landmarkerDecisionNode
	49	landmarker1NN

Table 2. The algorithms and hyperparameter space

Algorithm Type	**Hyperparameter names (values)**
Kernel SVM	C: [0.03125, 100] kernel: 'rbf', 'poly', 'sigmoid' gamma: [3.0517578125e-5, 8] degree: range(2, 6) coef0: [-1, 1] tol: [1e-5, 1e-1] shrinking: True, False
Linear SVM	C: [0.03125, 32768] penalty: 'l1', 'l2' loss: 'hinge', 'squared_hinge' multi_class: 'ovr' tol: [1e-5, 1e-1]
Gaussian naive Bayes	
Bernoulli naive Bayes	alpha: [1e-2, 100] fit_prior: True, False
KNN	n_neighbors: range(1, 100) wights: 'uniform', 'distance' p: range(1, 3)
Decision Tree	max_depth: range(1, 20) max_features: [0.0, 1.0] criterion: "gini", "entropy" min_samples_leaf: range(1, 20) min_samples_split: range(2, 20)
Random Forest	n_estimators: 100 criterion: "gini", "entropy" max_features: [0.0, 1.0] min_samples_leaf: range(1, 20) min_samples_split: range(2, 20) bootstrap: True, False
Adaboost	loss: 'hinge','log','modified_huber','squared_hinge', 'perceptron' alpha_sgd: [1e-7, 0.1] l1_ratio: [1e-9, 1] penalty: 'l2','l1','elasticnet' learning_rate: 'constant','optimal','invscaling' eta0: [1e-7, 1e-1] tol: [1e-5, 1e-1] average: [1,0] power_t: [1e-5, 1] epsion: [1e-5, 1e-1]
LDA	shrinkage_factor: [0, 1] shrinkage: 'auto', 'manual' n_components: range(1, max_features - 1) solver: 'svd','lsqr','eigen' tol: [1e-5, 1e-1]
QDA	reg_param: [0.0, 1.0]
Passive Aggressive	C: [1e-5, 10] loss: 'hinge','squared_hinge' tol: [1e-5, 1e-1] average: [1,0]
Extree Tree	n_estimators: 100 criterion: "gini", "entropy" max_depth: range(1, 20) max_features: [0.0, 1.0] min_samples_leaf: range(1, 20) min_samples_split: range(2, 20) bootstrap: True, False

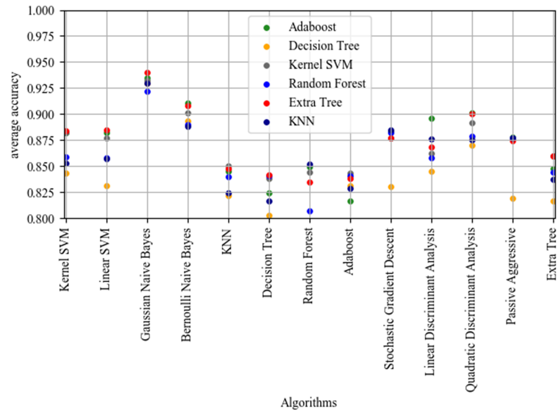

Fig. 1. The average accuracy of base learners for accuracy-oriented meta-model in one of 10 hold-out-validation

Table 3. The RMSE of base learners for runtime-oriented meta-model.

Algorithm Type	base learners		
	Ridge	Kernel SVR	RF
Kernel SVM	0.790	**0.601**	0.646
Linear SVM	2.585	**2.087**	2.194
Gaussian naive Bayes	**0.342**	0.372	0.348
Bernoulli naive Bayes	0.275	0.258	**0.257**
KNN	0.402	0.398	**0.333**
Decision Tree	1.128	1.030	**0.874**
Random Forest	0.637	0.669	**0.526**
Adaboost	4.223	4.196	**3.980**
SGD	1.710	**1.296**	1.305
LDA	1.125	**0.408**	0.502
QDA	1.316	0.606	**0.502**
Passive Aggressive	0.781	0.498	**0.490**
Extra Tree	**0.220**	0.229	0.233