DATABASE
The Journal of Biological Databases and Curation

Database, 2020, 1–8
doi: 10.1093/database/baaa050
Database tool

Database tool

mAML: an automated machine learning pipeline with a microbiome repository for human disease classification

Fenglong Yang and Quan Zou*

Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China,
No. 4, Section 2, North Jianshe Road, Chengdu 610054, China

***Corresponding author:** Email: zouquan@nclab.net

Citation details: Yang,F. and Zou,Q. mAML: an automated machine learning pipeline with a microbiome repository for human disease classification. *Database* (2020) Vol. 2020: article ID baaa050; doi:10.1093/database/baaa050

Received 11 February 2020; Revised 27 May 2020; Accepted 3 June 2020

Abstract

Due to the concerted efforts to utilize the microbial features to improve disease prediction capabilities, automated machine learning (AutoML) systems aiming to get rid of the tediousness in manually performing ML tasks are in great demand. Here we developed mAML, an ML model-building pipeline, which can automatically and rapidly generate optimized and interpretable models for personalized microbiome-based classification tasks in a reproducible way. The pipeline is deployed on a web-based platform, while the server is user-friendly and flexible and has been designed to be scalable according to the specific requirements. This pipeline exhibits high performance for 13 benchmark datasets including both binary and multi-class classification tasks. In addition, to facilitate the application of mAML and expand the human disease-related microbiome learning repository, we developed GMrepo ML repository (GMrepo Microbiome Learning repository) from the GMrepo database. The repository involves 120 microbiome-based classification tasks for 85 human-disease phenotypes referring to 12 429 metagenomic samples and 38 643 amplicon samples. The mAML pipeline and the GMrepo ML repository are expected to be important resources for researches in microbiology and algorithm developments.

Database URL: http://lab.malab.cn/soft/mAML

Introduction

Machine learning (ML) models have enabled key advances in many application fields and are crucial for data-driven medical research and translation, such as microbiome-based disease diagnosis or prognosis (1–3). As domain classification tasks are often context-dependent, no single data preprocessing method and ML strategy can handle all prediction issues. Due to the tedious nature of customizing ML tasks by domain scientists, several well-known automated machine learning (AutoML) systems, including Auto-WEKA (4), Auto-sklearn (5) and Auto-Net (6) have emerged to address the famous CASH (automatically and simultaneously choosing an ML algorithm and setting its hyper-parameters to optimize empirical performance) problem (7). While autoML systems are mature enough to enhance the prediction accuracy, there is not yet any work to build the autoML systems to meet the specific requirements of microbiome-based classification tasks (8–10). Focusing on the scenario of microbiome-associated phenotype prediction and considering the benefit of data preprocessing methods regarding ML estimators (1), an autoML pipeline named mAML was developed here to automatically generate an optimized ML model that exhibits sufficient performance for a personalized microbiome-based classification task.

The mAML pipeline possesses several advantages. Specifically, (i) the mAML pipeline can efficiently and automatically build an optimized, interpretable and robust model for a microbiome-based classification task. (ii) The mAML pipeline is deployed on a web-based platform (the mAML web server) that is user-friendly and flexible and has been designed to be scalable according to user requirements. (iii) The pipeline can be applied to both binary and multi-class classification tasks. (iv) The pipeline is data-driven, and it can be easily extended to the multi-omics data or other data types if only the domain-specific dataset is provided.

Furthermore, we developed a microbiome learning repository from the GMrepo database (11). GMrepo (data repository for Gut Microbiota) is a database of curated and consistently annotated human gut metagenomic data, which contains 58 903 human gut samples/runs, including 17 618 metagenomes and 41 285 amplicons from 253 projects concerning 92 phenotypes. GMrepo consistently processed and annotated the collected samples and manually curated all possible related meta-data of each sample/run. It organized the samples according to their associated phenotypes and offered the taxonomic (genus and species level) abundance information for all samples of high quality. Due to the necessity of aggregating samples across studies and appropriately handling candidate confounders in curating classification tasks, it is reasonable to develop a machine learning repository dedicated to human-disease-associated microbiome-based classification tasks from GMrepo. Hence, we present the GMrepo ML repository (GMrepo Microbiome Learning repository), a public repository of 120 microbiome-based classification tasks developed from the GMrepo database, which involves 38 643 amplicon samples referring to 71 disease phenotypes and 12 429 metagenomic samples covering 49 disease phenotypes. The files in the GMrepo ML repository can be downloaded and directly submitted to the mAML server or they can be imported into the phyloseq pipeline (12) for rapid, reproducible and interactive exploration of microbiome data.

The source code and benchmark datasets for the mAML pipeline are available at https://github.com/yangfenglong/mAML1.0. The docker image of the pipeline can be pulled from https://hub.docker.com/r/yangfenglong/dash_webserver. The GMrepo ML repository is freely available at http://39.100.246.211:8050/Dataset. The source code for the construction of the repository is available at https://github.com/yangfenglong/mAML1.0/blob/master/datasets/GMrepo_datasets/GMrepo.ipynb.

Method

The mAML pipeline is developed completely in Python, and the workflow is represented in Figure 1. The user can upload the BIOM file or Table file to start the pipeline. As shown in the demo dataset, the microbiome data inputs should include the feature count/abundance/presence information for the samples, the metadata for the samples and/or the metadata for the features. The features can represent OTUs as in 16S rRNA gene sequencing, genes as in metagenomics and transcriptomics, metabolites as in metabolomics, etc. The sample metadata should at least contain the labeling information of distinct groups, also called classes or phenotypes. The selection of different types of phenotypes and the collapse of features into higher levels are supported. First, the features that exhibit a prevalence lower than the threshold of 20% by default in all classes were filtered, as low prevalence features are usually not

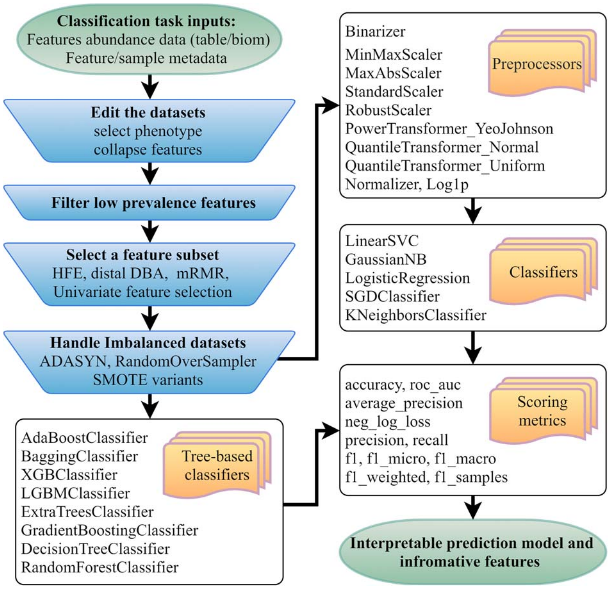


Figure 1. Flowchart of the mAML pipeline. At least two files indicated at the beginning of the pipeline should be provided. Operation steps before training are indicated in the blue inverse-trapezoids.

promising for use in the analysis of gut microbiota that possesses thousands of features across samples. Second, four feature subset selection (FSS) methods (the distal DBA method (13), HFE (14), univariate feature selection (15) and mRMR (16)) are adopted to handle high-dimensional and sparse feature spaces as in microbiome datasets. Then, the class imbalance problem is compensated by using RandomOverSampler (the random sampling with replacement) (17), SMOTE variants (Synthetic Minority Over-sampling Technique) (18) or ADASYN (Adaptive Synthetic Sampling Approach for Imbalanced Learning) (19), as imbalanced datasets containing overrepresented or underrepresented data can induce bias in prediction. Finally, the pipeline will automatically determine the optimized hyper-parameters for all classifiers, including the best preprocessors for non-tree-based classifiers, using parallel grid searches. The use of appropriate data preprocessing methods such as feature scaling is important when used in combination with normality-assumed algorithms such as metric-based, gradient-based and distance-based estimators, and it does not influence tree-based models (1). Hence, the optimized hyper-parameters for those non-tree-based models were explored while considering data preprocessing methods as one of the important hyper-parameters. Nested cross-validation (20) was used to avoid overfitting for each model in regard to training data, and 11 candidate scoring metrics, with accuracy as the default, were applied for model evaluation. In total, 10 data preprocessing methods and 13 classifiers, primarily derived from python machine learning package scikit-learn (1), were involved. Considering the interpretation requirements for the subsequent microbial studies, only those white-box preprocessors and classifiers were incorporated.

Utility of mAML

Overview

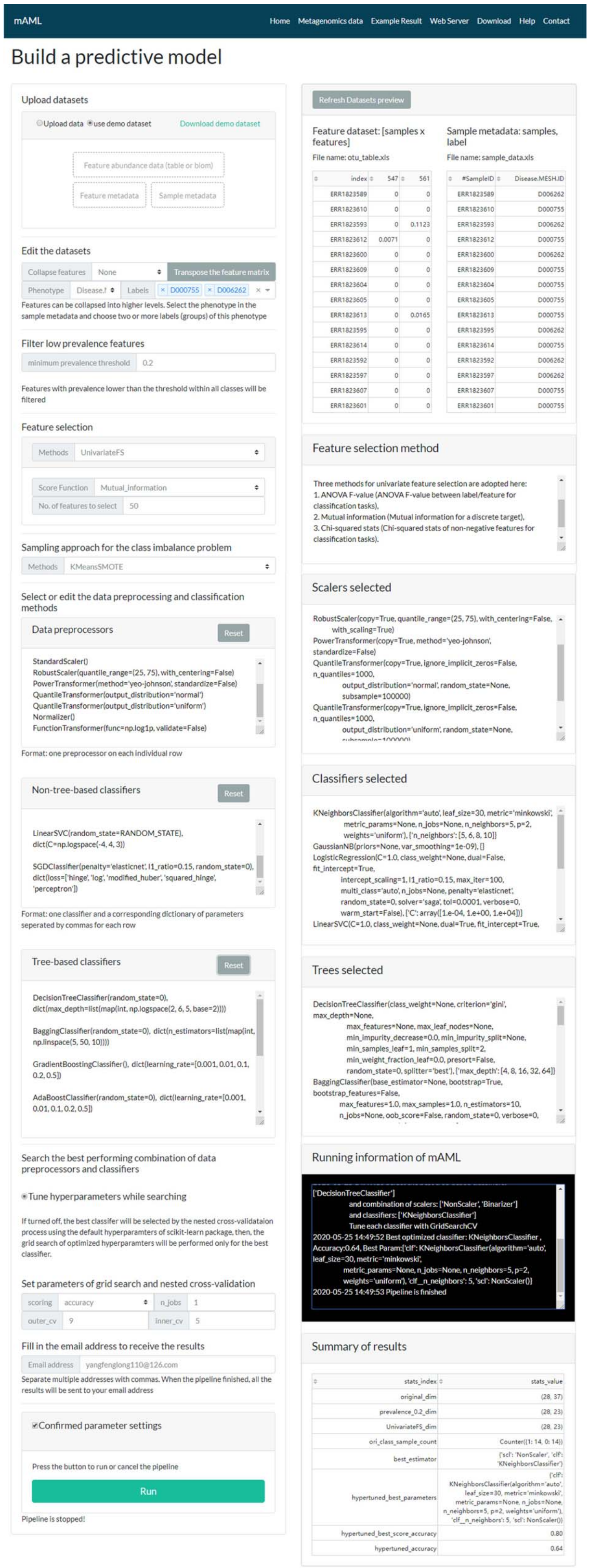
To facilitate the use of the mAML pipeline, we developed the mAML web server, which is a web-based machine learning system that can generate models in a user-friendly, flexible and scalable way. The main points regarding the implementation of the server are described as follows, and the details can be accessed on the server 'Help' page.

Web server implementation

Here, we will introduce the key points to navigate the server.

Submit a task

A typical classification task can be submitted to the mAML web server by the following steps (Figure 2). First, users



Build a predictive model

Upload datasets

○ Upload data ● use demo dataset

Edit the datasets

Collapse features: None

Phenotype: Disease.! Labels: D000755, D006262

Features can be collapsed into higher levels. Select the phenotype in the sample metadata and choose two or more labels (groups) of this phenotype

Filter low prevalence features

minimum prevalence threshold: 0.2

Features with prevalence lower than the threshold within all classes will be filtered

Feature selection

Methods: UnivariateFS

Score Function: Mutual_information

No. of features to select: 50

Sampling approach for the class imbalance problem

Methods: KMeansSMOTE

Select or edit the data preprocessing and classification methods

Data preprocessors

```
StandardScaler()
RobustScaler(quantile_range=(25, 75), with_centering=False)
PowerTransformer(method='yeo-johnson', standardize=False)
QuantileTransformer(output_distribution='normal')
QuantileTransformer(output_distribution='uniform')
Normalizer()
FunctionTransformer(func=np.log1p, validate=False)
```

Format: one preprocessor on each individual row

Non-tree-based classifiers

```
LinearSVC(random_state=RANDOM_STATE),
dict(C=np.logspace(-4, 4, 3))

SGDClassifier(penalty='elasticnet', l1_ratio=0.15, random_state=0),
dict(loss=['hinge', 'log', 'modified_huber', 'squared_hinge',
'perceptron'])
```

Format: one classifier and a corresponding dictionary of parameters seperated by commas for each row

Tree-based classifiers

```
DecisionTreeClassifier(random_state=0),
dict(max_depth=list(map(int, np.logspace(2, 6, 5, base=2))))

BaggingClassifier(random_state=0), dict(n_estimators=list(map(int,
np.linspace(5, 50, 10))))

GradientBoostingClassifier(), dict(learning_rate=[0.001, 0.01, 0.1,
0.2, 0.5])

AdaBoostClassifier(random_state=0), dict(learning_rate=[0.001,
0.01, 0.1, 0.2, 0.5])
```

Search the best performing combination of data preprocessors and classifiers

● Tune hyperparameters while searching

If turned off, the best classifer will be selected by the nested cross-validataion process using the default hyperparamters of scikit-learn package, then, the grid search of optimized hyperparamters will be performed only for the best classifier.

Set parameters of grid search and nested cross-validation

scoring: accuracy; n_jobs: 1

outer_cv: 9; inner_cv: 5

Fill in the email address to receive the results

Email address: yangfenglong110@126.com

Separate multiple addresses with commas. When the pipeline finished, all the results will be sent to your email address

☑ Confirmed parameter settings

Press the button to run or cancel the pipeline

Pipeline is stopped!

Feature dataset: [samples x features]

File name: otu_table.xls

index	547	561
ERR1823589	0	0
ERR1823610	0	0
ERR1823593	0	0.1123
ERR1823612	0.0071	0
ERR1823600	0	0
ERR1823609	0	0
ERR1823604	0	0
ERR1823605	0	0
ERR1823613	0	0.0165
ERR1823595	0	0
ERR1823614	0	0
ERR1823592	0	0
ERR1823597	0	0
ERR1823607	0	0
ERR1823601	0	0

Sample metadata: samples, label

File name: sample_data.xls

#SampleID	Disease.MESH.ID
ERR1823589	D006262
ERR1823610	D000755
ERR1823593	D006262
ERR1823612	D000755
ERR1823600	D006262
ERR1823609	D000755
ERR1823604	D000755
ERR1823605	D000755
ERR1823613	D000755
ERR1823595	D006262
ERR1823614	D000755
ERR1823592	D006262
ERR1823597	D006262
ERR1823607	D000755
ERR1823601	D000755

Feature selection method

Three methods for univariate feature selection are adopted here:
1. ANOVA F-value (ANOVA F-value between label/feature for classification tasks),
2. Mutual information (Mutual information for a discrete target),
3. Chi-squared stats (Chi-squared stats of non-negative features for classification tasks).

Scalers selected

```
RobustScaler(copy=True, quantile_range=(25, 75), with_centering=False,
      with_scaling=True)
PowerTransformer(copy=True, method='yeo-johnson',
standardize=False)
QuantileTransformer(copy=True, ignore_implicit_zeros=False,
n_quantiles=1000,
          output_distribution='normal', random_state=None,
          subsample=100000)
QuantileTransformer(copy=True, ignore_implicit_zeros=False,
n_quantiles=1000,
          output_distribution='uniform', random_state=None,
```

Classifiers selected

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
          weights='uniform'), {'n_neighbors': [5, 6, 8, 10]}
GaussianNB(priors=None, var_smoothing=1e-09), {}
LogisticRegression(C=1.0, class_weight=None, dual=False,
fit_intercept=True,
          intercept_scaling=1, l1_ratio=0.15, max_iter=100,
          multi_class='auto', n_jobs=None, penalty='elasticnet',
          random_state=0, solver='saga', tol=0.0001, verbose=0,
          warm_start=False), {'C': array([1.e-04, 1.e+00, 1.e+04])}
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
```

Trees selected

```
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=None,
          max_features=None, max_leaf_nodes=None,
          min_impurity_decrease=0.0, min_impurity_split=None,
          min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, presort=False,
          random_state=0, splitter='best'), {'max_depth': [4, 8, 16, 32, 64]}
BaggingClassifier(base_estimator=None, bootstrap=True,
bootstrap_features=False,
        max_features=1.0, max_samples=1.0, n_estimators=10,
        n_jobs=None, oob_score=False, random_state=0, verbose=0,
```

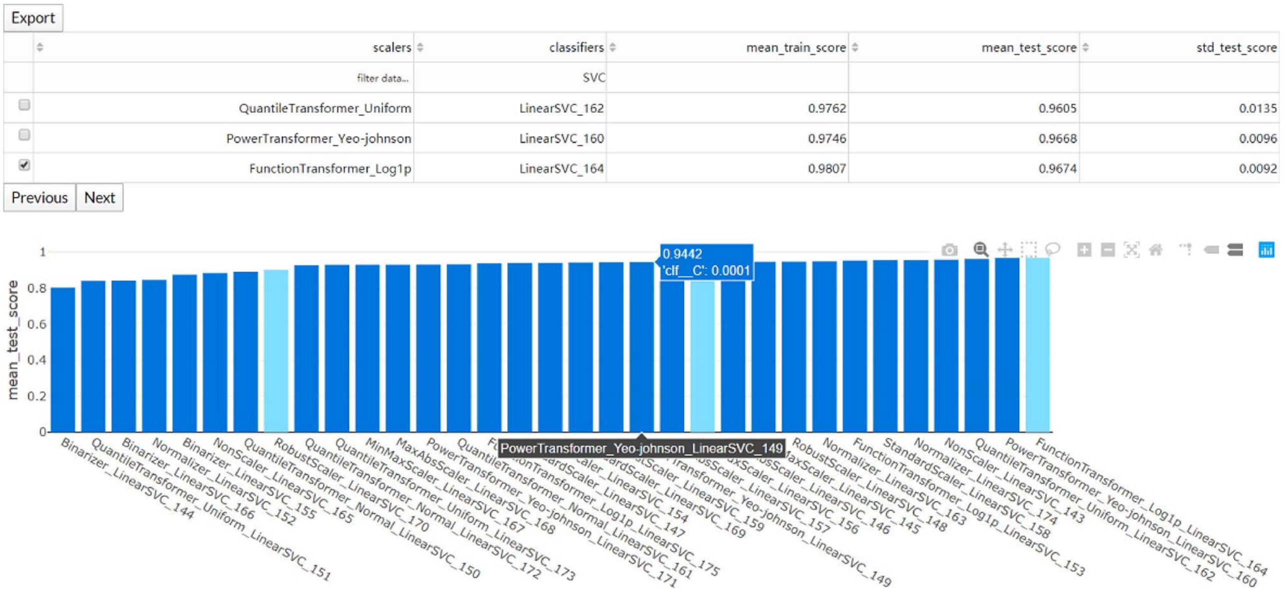
Running information of mAML

```
['DecisionTreeClassifier']
          and combination of scalers: ['NonScaler', 'Binarizer']
          and classifiers: ['KNeighborsClassifier']
          Tune each classifier with GridSearchCV
2020-05-25 14:49:52 Best optimized classifier: KNeighborsClassifier ,
Accuracy:0.64, Best Param:{'clf': KNeighborsClassifier(algorithm='auto',
leaf_size=30, metric='minkowski',
        metric_params=None, n_jobs=None, n_neighbors=5, p=2,
        weights='uniform'), 'clf__n_neighbors': 5, 'scl': NonScaler()}
2020-05-25 14:49:53 Pipeline is finished
```

Summary of results

stats_index	stats_value
original_dim	(28, 37)
prevalence_0.2_dim	(28, 23)
UnivariateFS_dim	(28, 23)
ori_class_sample_count	Counter({1: 14, 0: 14})
best_estimator	{'scl': 'NonScaler', 'clf': 'KNeighborsClassifier'}
hypertuned_best_parameters	{'clf': KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=None, n_neighbors=5, p=2, weights='uniform'), 'clf__n_neighbors': 5, 'scl': NonScaler()}
hypertuned_best_score_accuracy	0.80
hypertuned_accuracy	0.64

Figure 2. The task submission page of the mAML server. The left column displays the settings for each step, and the right column shows the real-time feedback of the parameter settings.



Export

	scalers	classifiers	mean_train_score	mean_test_score	std_test_score
	filter data...	SVC			
☐	QuantileTransformer_Uniform	LinearSVC_162	0.9762	0.9605	0.0135
☐	PowerTransformer_Yeo-johnson	LinearSVC_160	0.9746	0.9668	0.0096
☑	FunctionTransformer_Log1p	LinearSVC_164	0.9807	0.9674	0.0092

Previous Next

mean_test_score
1
0.8
0.6
0.4
0.2
0
0.9442
'clf__C': 0.0001
PowerTransformer_Yeo-johnson_LinearSVC_149
Binarizer_LinearSVC_144
QuantileTransformer_Uniform_LinearSVC_151
Binarizer_LinearSVC_166
Normalizer_LinearSVC_152
Binarizer_LinearSVC_155
NonScaler_LinearSVC_165
QuantileTransformer_Normal_LinearSVC_150
RobustScaler_LinearSVC_170
QuantileTransformer_Normal_LinearSVC_172
QuantileTransformer_Uniform_LinearSVC_173
MinMaxScaler_LinearSVC_167
MaxAbsScaler_LinearSVC_168
PowerTransformer_Yeo-johnson_LinearSVC_171
QuantileTransformer_Normal_LinearSVC_161
FunctionTransformer_Log1p_LinearSVC_153
QuantileTransformer_Uniform_LinearSVC_162
PowerTransformer_Yeo-johnson_LinearSVC_160
FunctionTransformer_Log1p_LinearSVC_164

Figure 3. The interaction diagram for the performance of all the candidate models. The users can screen the candidate models by the scalers, classifiers, mean training score, mean test score and standard test score.

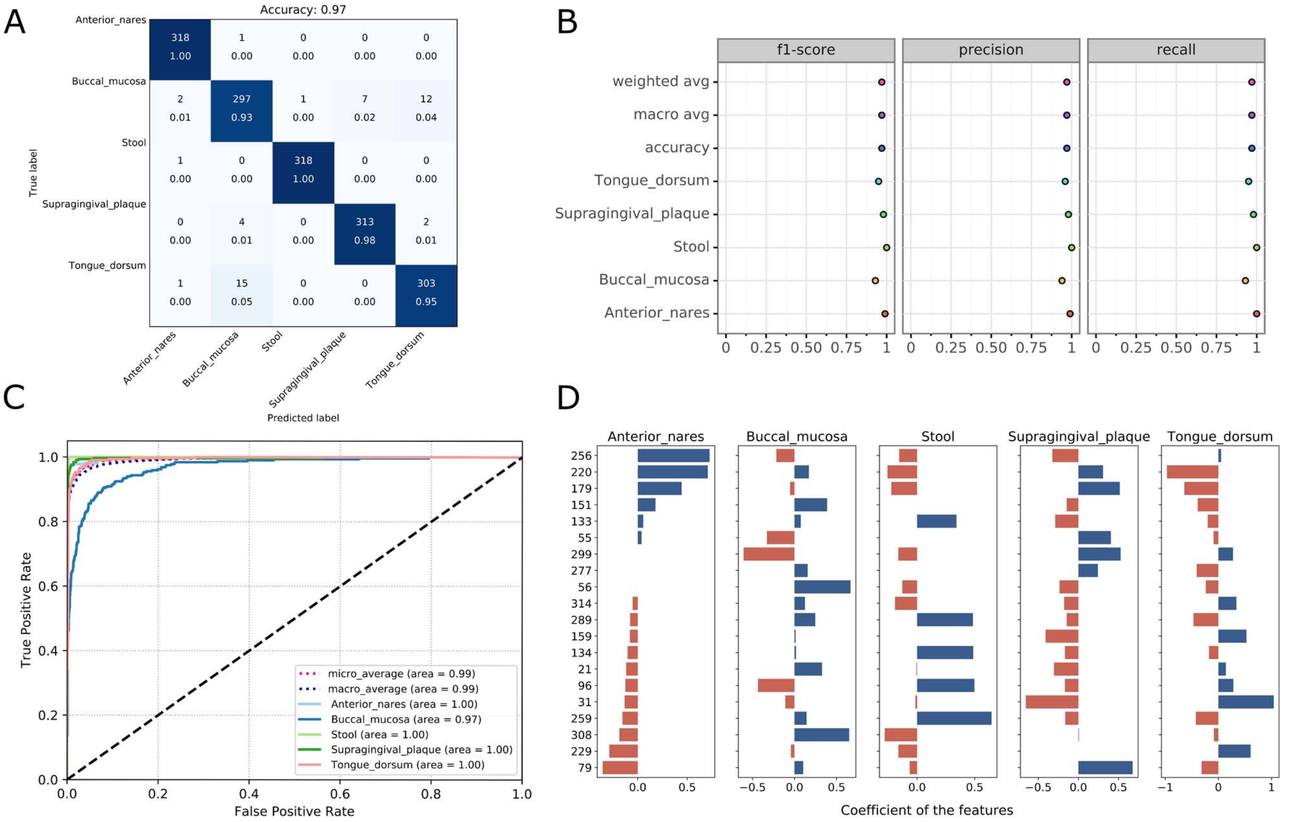
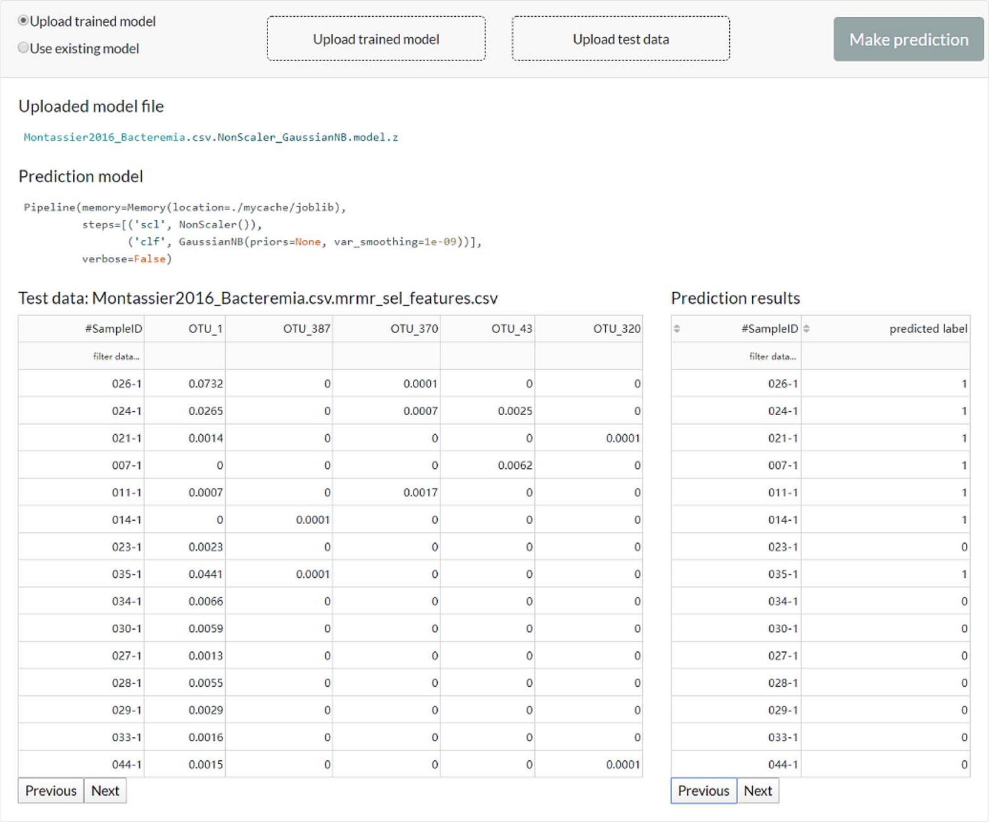


Figure 4. Visualizations for the optimized model: the heatmap for confusing matrix (**A**), the classification report (**B**), ROC curve (**C**) and the histogram for top features (**D**, default: 20). Note that, in the case of tree-based models, the feature importance will be provided instead of the feature's coefficient in the histogram.

can choose the example datasets or upload datasets of their own to start the pipeline. Second, the input features will be filtered at the specified threshold (by default, taxonomic features with a percentage lower than 20% in all classes were disregarded in this work). Third, the most relevant features will be selected using Fizzy-mRMR (top 50 features by default), which is general for various kinds of features. The FSS option can be deselected if there is no need to



● Upload trained model
○ Use existing model

Uploaded model file

```
Montassier2016_Bacteremia.csv.NonScaler_GaussianNB.model.z
```

Prediction model

```
Pipeline(memory=Memory(location=./mycache/joblib),
         steps=[('scl', NonScaler()),
                ('clf', GaussianNB(priors=None, var_smoothing=1e-09))],
         verbose=False)
```

Test data: Montassier2016_Bacteremia.csv.mrmr_sel_features.csv

#SampleID	OTU_1	OTU_387	OTU_370	OTU_43	OTU_320
filter data...					
026-1	0.0732	0	0.0001	0	0
024-1	0.0265	0	0.0007	0.0025	0
021-1	0.0014	0	0	0	0.0001
007-1	0	0	0	0.0062	0
011-1	0.0007	0	0.0017	0	0
014-1	0	0.0001	0	0	0
023-1	0.0023	0	0	0	0
035-1	0.0441	0.0001	0	0	0
034-1	0.0066	0	0	0	0
030-1	0.0059	0	0	0	0
027-1	0.0013	0	0	0	0
028-1	0.0055	0	0	0	0
029-1	0.0029	0	0	0	0
033-1	0.0016	0	0	0	0
044-1	0.0015	0	0	0	0.0001

Prediction results

#SampleID	predicted label
filter data...	
026-1	1
024-1	1
021-1	1
007-1	1
011-1	1
014-1	1
023-1	0
035-1	1
034-1	0
030-1	0
027-1	0
028-1	0
029-1	0
033-1	0
044-1	0

Figure 5. The web server interface for the users to reuse the existing model or upload a previously trained model to make predictions.

downsize the number of features. Fourth, the unbalanced datasets will be rebalanced using SMOTE by default, and this option can be turned off if it is unnecessary to perform imbalanced learning. Finally, the default parameters and hyperparameters of the preprocessors and classifiers could be reconfigured, and the adding and pruning of any preprocessor or classifier is supported. The grid search settings for the hyperparameters can be altered in the dict function of each classifier. By default, the pipeline will search the optimized combination of preprocessors and non-tree-based classifiers and simultaneously optimize the hyperparameters for all classifiers. The parameters of nested cross-validation, the metrics for model evaluation and the number of parallel processes are also tunable.

When the email address is filled in and all the above settings are confirmed, the user can start the pipeline and it will run in the background. The status of the running task will update automatically in the 'Running information' window of the 'Web Server' page.

Preview the result files and download

Once the run is completed, the compressed result will be automatically sent to the predefined e-mail address or can be downloaded from the 'Web Server' page of the server. The user can preview the interaction diagram (Figure 3) for the performance of all the candidate models on the server. For each task, the pipeline automatically outputs the visualization results for the optimal model, including the heatmap of the confusing matrix (Figure 4A), the classification report (Figure 4B), the ROC curve (Figure 4C) and the histogram of top features (Figure 4D, default: 20), which can be investigated in further study.

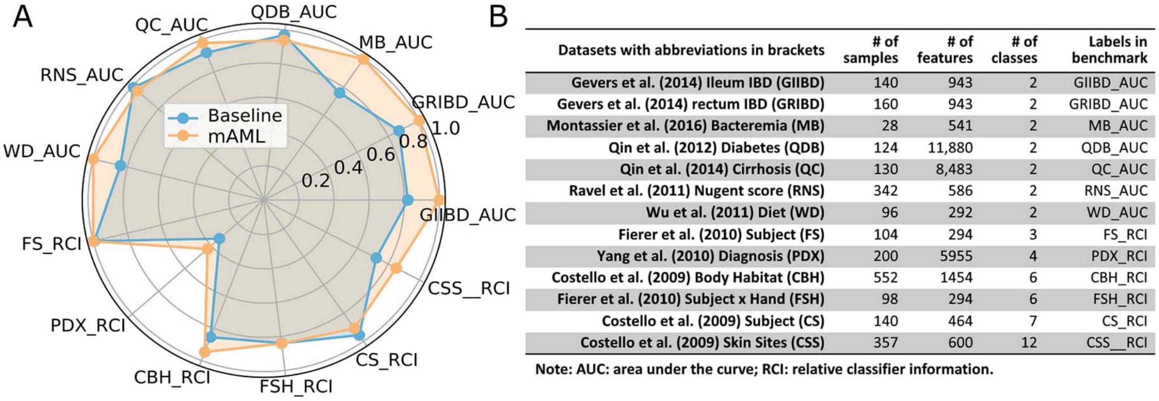
The result is reproducible within a container started based on the docker image or via the webserver. An example result is represented on the 'Example Result' page of the server.

Make new predictions

The user can feed new data to the existing model or upload a previously trained model to get new predictions (Figure 5).

Performance of mAML

The performance of the pipeline was investigated by performing analyses on 13 human microbiome datasets that are publicly available and appropriate for benchmarking, which involve 7 binary classification tasks (21) and 6 multi-class classification tasks (22). These datasets, including 11



Datasets with abbreviations in brackets	# of samples	# of features	# of classes	Labels in benchmark
Gevers et al. (2014) Ileum IBD (GIIBD)	140	943	2	GIIBD_AUC
Gevers et al. (2014) rectum IBD (GRIBD)	160	943	2	GRIBD_AUC
Montassier et al. (2016) Bacteremia (MB)	28	541	2	MB_AUC
Qin et al. (2012) Diabetes (QDB)	124	11,880	2	QDB_AUC
Qin et al. (2014) Cirrhosis (QC)	130	8,483	2	QC_AUC
Ravel et al. (2011) Nugent score (RNS)	342	586	2	RNS_AUC
Wu et al. (2011) Diet (WD)	96	292	2	WD_AUC
Fierer et al. (2010) Subject (FS)	104	294	3	FS_RCI
Yang et al. (2010) Diagnosis (PDX)	200	5955	4	PDX_RCI
Costello et al. (2009) Body Habitat (CBH)	552	1454	6	CBH_RCI
Fierer et al. (2010) Subject x Hand (FSH)	98	294	6	FSH_RCI
Costello et al. (2009) Subject (CS)	140	464	7	CS_RCI
Costello et al. (2009) Skin Sites (CSS)	357	600	12	CSS__RCI

Note: AUC: area under the curve; RCI: relative classifier information.

Figure 6. The performance comparison of the mAML proposed models against the baseline (**A**). The labels are connected abbreviations with an underline between the name of the database and the metric used in the original study (**B**).

amplicon datasets and 2 metagenomic datasets, vary across microbiome-related phenotypes and cover 2471 samples, which can be retrieved from the 'Metagenomics data' page of the server or GitHub repository.

The models proposed by mAML outperform most of the models in the original studies (Table S1, Figure 6), confirming the robustness and reliability of this method. Since only the 'white box' preprocessors and classifiers are involved in the prediction, the optimized model selected for each task is interpretable and the top features indicated by the model merit further study. The detailed results for the candidate models and optimized model of each task are available at the GitHub repository.

GMrepo ML repository

Furthermore, we developed a GMrepo ML repository (GMrepo Microbiome Learning repository) from the GMrepo database to facilitate the utilization of mAML and expand the microbiome learning repository related to human disease. The framework of the GMrepo ML repository construction is presented in Figure 7, and the details are described as follows.

We downloaded the metadata for all samples that passed the QC procedure (QCStatus = 1) from the GMrepo website and retrieved the taxonomic abundance information (including the genus and species level) for all the metagenomic and amplicon samples respectively using the RESTful APIs of GMrepo. Taxa with the scientific name labeled as unknown or Others were deleted from the taxonomic abundance table since they are not meaningful features. Taxonomic lineage and tree were retrieved according to the NCBI taxonomy id of each taxon by using the ETE3 python module. Then, four files (taxonomic abundance table, metadata table, taxonomic lineage table and taxonomic tree file) were obtained for all the metagenomic and amplicon samples respectively, as represented in Figure 7. Each file was then divided into subfiles according to different phenotypes, and only those phenotypes with healthy control samples are preserved. Totally, the repository involves 12 429 metagenomic samples covering 49 disease phenotypes and 38 643 amplicon samples referring to 71 disease phenotypes. For each phenotype, the taxonomic abundance table and metadata table can be directly submitted to the mAML server to build an optimized model for disease prediction. By using the phyloseq R package (12), all the subfiles files from each phenotype can be taken as components to build the phyloseq-class object (phyloseq.RData), which can be imported into the Shiny-phyloseq web application (23) for subsequent interactive exploration of microbiome data. Additionally, the GMrepo ML repository is also provided as

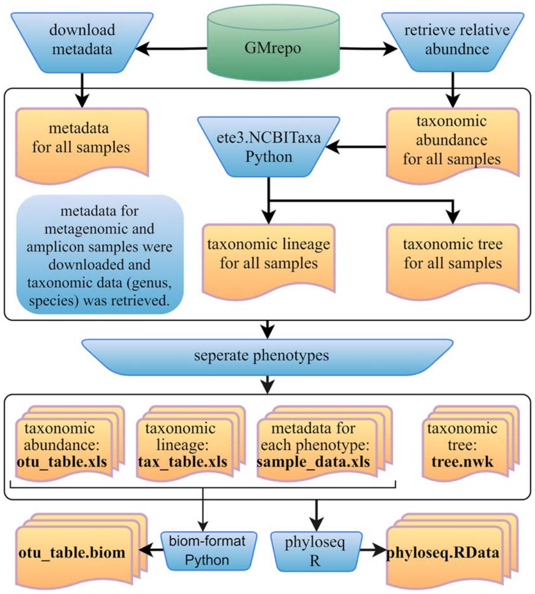

download metadata
GMrepo
retrieve relative abundnce
metadata for all samples
ete3.NCBITaxa Python
taxonomic abundance for all samples
metadata for metagenomic and amplicon samples were downloaded and taxonomic data (genus, species) was retrieved.
taxonomic lineage for all samples
taxonomic tree for all samples
seperate phenotypes
taxonomic abundance: otu_table.xls
taxonomic lineage: tax_table.xls
metadata for each phenotype: sample_data.xls
taxonomic tree: tree.nwk
otu_table.biom
biom-format Python
phyloseq R
phyloseq.RData

Figure 7. Framework of the GMrepo ML repository construction. Operation steps are indicated in the blue inverse-trapezoids. Files with names in bold are all contained in the repository, and they can be retrieved from the 'Metagenomics data' page of the server or from the GitHub.

BIOM format, which is a general-use format to represent biological data and is currently supported by almost all state-of-the-art software in the field of microbiome.

The users can apply the mAML pipeline to their interested datasets in the GMrepo ML repository. The datasets can also be merged with their own samples to perform meta-analysis. Moreover, multiple feature types such as metabolites and metatranscripts are encouraged to be integrated with the taxonomic or functional features from metagenomics to build a multi-omics-feature based model to the target disease.

Conclusion

Considering the tedious work and context-dependent nature of manually performing the microbiome-based classification tasks, we developed an autoML pipeline, namely mAML, which can rapidly and automatically generate an optimized and interpretable model with sufficient performance for binary or multi-class classification tasks in a reproducible way. The pipeline is deployed on a web-based platform, and the mAML server is user-friendly and flexible and has been designed to be scalable according to user requirements.

We highlight the reliability and robustness of mAML with its high performance on 13 benchmark datasets. Being data-driven, the mAML pipeline can be easily extended to the multi-omics data of microbes and other data types if only the domain-specific feature data are supplied. Moreover, we constructed the GMrepo ML repository of 120 microbiome-based classification tasks for 85 disease phenotypes, which facilitates the application of mAML and is expected to be an important resource for algorithm developers.

Supplementary Data

Supplementary data are available at *Database* Online.

Funding

The work was supported by the National Key R&D Program of China [2018YFC0910405] and the National Natural Science Foundation of China [No. 61771331, No. 61922020, No. 91935302].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pedregosa,F., Varoquaux,G., Gramfort,A. *et al.* (2011) Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
2. Wirbel,J., Pyl,P.T., Kartal,E. *et al.* (2019) Meta-analysis of fecal metagenomes reveals global microbial signatures that are specific for colorectal cancer. *Nat. Med.*, **25**, 679–689.
3. Qin,J., Li,Y., Cai,Z. *et al.* (2012) A metagenome-wide association study of gut microbiota in type 2 diabetes. *Nature*, **490**, 55–60.
4. Kotthoff,L., Thornton,C., Hoos,H.H. *et al.* (2017) Auto-WEKA 2.0: automatic model selection and hyperparameter optimization in WEKA. *J. Mach. Learn. Res.*, **18**, 1–5.
5. Feurer,M., Klein,A., Eggensperger,K. *et al.* (2015) Efficient and robust automated machine learning. In: Cortes C, Lawrence ND, Lee DD *et al.* (eds). *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., pp. 2962–2970.
6. Mendoza,H., Klein,A., Feurer,M. *et al.* (2019) Towards automatically-tuned deep neural networks. In: Hutter F, Kotthoff L, Vanschoren J (eds). *Automated Machine Learning: Methods, Challenges*. Springer International Publishing, Cham, pp. 135–149.
7. Hutter,F., Kotthoff,L. and Vanschoren,J. (2019) Automatic machine learning: methods, systems, challenges. Automatic machine learning: methods, systems, challenges. In: *Challenges in Machine Learning*. Springer, Germany.
8. Pasolli,E., Truong,D.T., Malik,F. *et al.* (2016) Machine learning meta-analysis of large metagenomic datasets: tools and biological insights. *PLoS Comput. Biol.*, **12**, e1004977.
9. Moitinhosilva,L., Steinert,G., Nielsen,S. *et al.* (2017) Predicting the HMA-LMA status in marine sponges by machine learning. *Front. Microbiol.*, **8**, 752.
10. Topcuoglu,B.D., Lesniak,N.A., Ruffin,M.T. *et al.* (2020) A framework for effective application of machine learning to microbiome-based classification problems. *bioRxiv*, 816090. doi.org/10.1101/816090.
11. Wu,S., Sun,C., Li,Y. *et al.* (2019) GMrepo: a database of curated and consistently annotated human gut metagenomes. *Nucleic Acids Res*, **48**, D545–D553.
12. McMurdie,P.J. and Holmes,S. (2013) Phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data. *PLoS One*, **8**, e61217.
13. Quinn,T.P. and Erb,I. (2020) Interpretable log contrasts for the classification of health biomarkers: a new approach to balance selection. *mSystems*, **8**, e00230–19.
14. Oudah,M. and Henschel,A. (2018) Taxonomy-aware feature engineering for microbiome classification. *BMC Bioinformatics*, **19**, 227.
15. Pedregosa,F., Varoquaux,G., Gramfort,A. *et al.* (2011) Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
16. Brown,G., Pocock,A.C., Zhao,M. *et al.* (2012) Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *J. Mach. Learn. Res.*, **13**, 27–66.
17. Lemaître,G., Nogueira,F. and Aridas,C.K. (2017) Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. *J. Mach. Learn. Res.*, **18**, 1–5.
18. Chawla,N.V., Bowyer,K.W., Hall,L.O. *et al.* (2002) SMOTE: synthetic minority over-sampling technique Nitesh. *J. Artif. Intell. Res.*, **16**, 321–357.
19. He,H., Bai,Y., Garcia,E.A. *et al.* (2008) ADASYN: adaptive synthetic sampling approach for imbalanced learning. 1322–1328. doi.org/10.1109/IJCNN.2008.4633969.

20. Cawley,G.C. and Talbot,N.L.C. (2010) On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.*, **11**, 2079–2107.
21. Vangay,P., Hillmann,B.M. and Knights,D. (2019) Microbiome learning repo (ML repo): a public repository of microbiome regression and classification tasks. *Gigascience*, **8**, 1–12.
22. Statnikov,A., Henaff,M., Narendra,V. *et al.* (2013) A comprehensive evaluation of multicategory classification methods for microbiomic data. *Microbiome*, **1**, 1–12.
23. McMurdie,P.J. and Holmes,S. (2015) Shiny-phyloseq: web application for interactive microbiome analysis with provenance tracking. *Bioinformatics*, **31**, 282–283.