**Supplementary Material**

1. Supplementary Method
   1. Simulation datasets

We conducted experiments based on the semi-synthetic data. For fair evaluation and comparison, we followed the same simulation procedures and settings of MITRE *(Bogart et al., 2019)* to generate simulation datasets. The simulation datasets are simulated from *(Bokulich et al., 2016)* to predict delivery type using a parametric bootstrapping-style procedure. For the raw dataset, we excluded subjects with fewer than 13 time points, who had no samples before 10 days, and who were studied for less than 600 days. This yielded 20 subjects for bootstrapping. We then truncated the data to an interval between 10 and 600 days, since this contained the densest sampling across subjects.

Then the same as MITRE *(Bogart et al., 2019)*, we set single-clade perturbation and two-clade perturbation on the phylogenetic tree to simulate and generate the disease subjects. Simulated cases were generated by randomly selecting and perturbing bacterial clades over a randomly selected limited time window (about 20% of the duration of the study, here 120 days), an equal number of control subjects were simulated. For the one-clade perturbation scenarios, the clade remained unperturbed for the simulated cases; for the two-clade perturbation scenarios, one clade was perturbed in the simulated control subjects, and both were perturbed in the simulated cases. The specific parameters and principles of perturbation settings can be seen from the raw MITRE paper *(Bogart et al., 2019).*

The perturbation is introduced to simulate subjects with a “disease.” After that, MITRE introduces a model of microbiome dynamics to simulate the time points that not present in the original dataset. MITRE models the underlying microbiome data as arising from latent time-dependent stochastic processes (Gaussian random walks):

Logo

Description automatically generated with low confidence(1)

Here, is the latent trajectory for OTU *o* in subject *s* at time *t*, and *τ2* is the process variance parameter, which is empirically estimated from the real data as approximately the 75-percentile variance. MITRE assumes a Bayesian model and infers the posterior latent trajectories using a one-step ahead MCMC algorithm, except in this case, trajectories are assumed to be independent of one another. The observed data , consisting of sequencing counts, is assumed to arise through a two-stage error model:

Text, letter

Description automatically generated(2)

Here, DMD denotes the Dirichlet-Multinomial distribution with concentration parameter *α* and number of simulated sequencing reads per sample *N*; we use parameters estimated from data (*α* = 286; N = 50,000) as previously described in MITRE *(Bogart et al., 2019)*. The model thus provides temporal coherence through the Gaussian random walk latent trajectory, while modelling compositionality and over-inflation through the two-stage error model. Posterior samples from the model capture temporal trends seen in the real data (e.g., periods of time in which a particular OTU are increasing), but with randomness introduced so that subjects sampled with replacement look sufficiently different.

For fairness, we directly leveraged the simulation method of MITRE described above to simulate the dataset since MITRE is the only published study related to using simulation datasets for disease prediction in the longitudinal microbiome. After the simulation, the new datasets are composed of 32 subjects, and each subject includes 308 OTU features with 18 observations. The detailed description for each classification task is listed in Table 1.

* 1. Detailed depictions for real datasets

The real 16S rRNA datasets are retrieved from the published human microbiome studies based on 16S rRNA amplicon sequencing. The first dataset is from *David et al.(2014)*. They studied the influences of dietary intake type for the human gut microbiome and explored the individual differences between animal-based diet and plant-based subjects. For this dataset, we define one binary classification task to predict the human diet type (animal-based diet or plant-based diet). The second dataset is from *Bokulich et al.(2016)*. They profiled the early-life microbiome development of infants to illustrate the complexity of its sensitivity to perturbation. For this dataset, we define two tasks to predict infant diet type (formula-dominant diet or breastmilk-based diet) and delivery mode (cesarean delivery or normal delivery) respectively. The third dataset is from *Vatanen et al.(2016)*.They followed the gut microbiome development from birth until age three in hundreds of infants to uncover the potential mechanism linking it to immune diseases. For this dataset, we define three tasks made up of nationality (Russian nationality or else), allergens (be allergy to eggs or not), and serum IgE (Immunoglobulin E) levels (elevate or not).

For metagenomic datasets, we downloaded the latest version of curatedMetagenomicData R package *(Pasolli E et al. 2017)* from (https://github.com/waldronlab/curatedMetagenomicData) to retrieve the longitudinal shotgun metagenomic datasets. We checked the existing mixed datasets in curatedMetagenomicData one by one and picked out 5 longitudinal datasets for use, since they all have the available binary description of study case (host status), enough observations for each subject (more than 3 time points), and enough total number of subjects (at least 20).

For HMP datasets, we entered the HMP website (https://hmpdacc.org/) to download and select HMP datasets *(Integrative H M P, 2014).* There are only 3 longitudinal datasets in iHMP (HMP for integrated longitudinal datasets) and we choose 2 of them (HMPibdmdb, HMPt2d) since they have available predicted variables.

For BrooksB dataset, we define the task to predict cesarean delivery or normal delivery *(Brooks B et al. 2017);* for HallAB dataset, we define the task to predict have inflammatory bowel disease (IBD) or not *(Hall A B et al. 2017);* for HeizBA dataset, we define the task to predict have type 1 diabetes (T1D) or *not (Heintz-Buschart A et al. 2016)*; for RaymondF dataset, we define the task to predict whether the subject used antibiotics (Cephalosporins) or not *(Raymond F et al. 2016);* for VincentC dataset, we define the task to predict whether the subject has clostridium difficile infection (CDI) or not *(Vincent C et al. 2016);* for ShaoY dataset, we define the task to predict cesarean delivery or normal delivery *(Shao Y et al. 2019)*; for HMPibdmdb dataset, we define the task to predict have the inflammatory bowel disease (IBD) or not; for HMPt2d dataset, we define the task to predict have type 2 diabetes (T2D) or not.

For all the utilized HMP and shotgun metagenomic datasets, we keep the same data preparation strategies, classifiers, and parameters settings as 16S rRNA amplicon sequencing data to conduct the comparison experiments.

* 1. Feature engineering

For feature engineering, in the machine learning field, feature aggregation is a commonly used feature engineering technology *(Bahnsen A C et al. 2016).* Existing studies have proved that it is possible to enrich the feature representation by feature aggregation to improve the final prediction *(Nargesian F et al. 2017).* However, one big drawback of the feature aggregation method is that sometimes the created new feature is unexplainable and makes no sense, although they can help improve the final prediction performance. Therefore, in this study, instead of randomly creating features of artifacts, we generate the new features by a phylogenetic tree. Extracting information from a phylogenetic tree for feature engineering has been adopted by many existing studies and has been proved to be *useful (Reiman D et al. 2020, Oudah M et al. 2018, Fioravanti D et al. 2018)*. In this kind of context, we proposed our phylogenetic aggregation to achieve better performance. The new features integrate the phylogenetic hierarchy relationship which can further enrich the microbiome feature representation. The relative abundance estimate of ancestral nodes on a phylogenetic tree is obtained by summing the relative abundances of its children for each node in the phylogenetic tree and then added to the end of the temporal abundance table.

Specifically, the input features consist of 2 sources: the first one is based on the raw OTU tables obtained by analysis pipeline and the second one is obtained by summing the relative abundances of its children for each node in the phylogenetic tree, which is similar to the data incorporation technology in single-point microbiome host trait prediction. For the sake of clarity, we make a simple example here.

Assuming that 0.1, 0.2, 0.5, 0.2 represent the relative abundance of OTU1, OTU2, OTU3 and OTU4 respectively, their placements on the phylogenetic tree are the children nodes as in Figure S1. For input features, except the abundance features derived from the analysis pipeline (0.1, 0.2, 0.5, 0.2), we also add extra taxa abundance features by summing the relative abundances of its children for each node in the phylogenetic tree to add more features (0.3, 0.7, 1). The phylogenetic aggregation adds the ancestral features which can make the feature representation more comprehensive than the opposite case. Since the OTU abundances for each observation changed, the phylogenetic aggregation also changed; therefore, it can be also seen as the time series data of phylogenetic information.

**图片包含 示意图

描述已自动生成**

**Figure S1. The example of the phylogenetic aggregation**.

After getting the extra features from the phylogenetic tree, we directly merged them with the raw abundance profiles to form a multivariate time series table, and the phylogenetic features can be seen as the new generated features by introducing domain knowledge.

* 1. Data preprocessing (imputation)

**Diagram

Description automatically generated**

**Figure S2. The overview of the data collection steps**.

For data collection before data preprocessing, there are several steps to obtain the time series data for each subject in the datasets. At first, we retrieved our real datasets from three published human microbiome studies related to the analysis of host status. The raw files obtained by the sequence feature extraction pipeline need to be preprocessed to obtain our required data format, therefore we implemented some data collection steps before data preprocessing. Specifically:

1. For each subject in the dataset, we searched the Subject ID and gathered all the observations (sampled from different time points) belonged to this ID.
2. Then we defined the label for each subject (host status) according to the research topic in the original papers.
3. We aggregated the observations to obtain a matrix according to the initial orders of their corresponding sampling time points for each subject.

The overview steps can be seen in Figure S2. Here, we take David dataset as an example. Assuming the experiment time window for this study is from day -5 to day 10, it means the sampling time from the beginning of the experiment to the end of the experiment (day -5 is defined by the original paper). Therefore, the perfect matrix size for each subject should be 16\*n, where 16 is the number of the observations (time points) in the whole experiment period (16 days in total) and n is the number of the OTU features. Observations mean the OTU abundance profiles (a one-dimensional feature vector) for this subject sampled at different time points (e.g., day -4, day -2, day -1, day 9 ……). Quantitatively, if the dataset is complete, for each subject, we will have 16 observations from day -5 to day 10. However, actually, the number of observations and the specific sampling time point may not be consistent for each subject. For example:

Subject 1: [#., -4., #., -2., -1., 0., 1., 2., 3., 4., 5., 6., 7., 8., 9., #.]

Subject 2: [#., -4., -3., -2., -1., 0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.]

Subject 3: [#., -4., #., -2., #., 0., 1., 2., 3., #., 5., #., 7., #., 9., 10.]

Where # means the missing observation at this time point. After obtaining the observations for each subject, we aggregated the observations to be a matrix. Take Subject 1 for example, the matrix can be seen in Figure S3.

Table

Description automatically generated with medium confidence

**Figure S3. The time series matrix of Subject 1 in David dataset**.

For Subject 1, there are missing observations on day -5, day -3, and day 10. Besides, we also listed the distributions of all the observations sampled at all the time points for each dataset which can be seen in Figure S4.



**Figure S4. The distribution of all the observations in all datasets.**

As we can see in Figure S4, in David dataset, there are 233 observations for 20 subjects; in Bokulich dataset, there are 471 observations for 35 subjects; in Karelia dataset, there are 976 observations for 113 subjects.

After obtaining the incomplete time series matrix for each subject, we conduct a series of data preprocessing steps to filter some raw OTU features and the phylogenetic aggregation for feature engineering to add some new OTU features which have been described in 2.3. Next, since the subjects in the datasets can have different numbers of observations, and the time interval for each observation is not consistent, we divided the experiment into certain amounts of equal pieces and taking any certain consecutive intervals as a valid time window. The whole process can be seen in Figure S5.

Graphical user interface, application

Description automatically generated

**Figure S5. The process of adaptive imputation approach.**

**Table S1. The feature value of OTU 1 and the corresponding sampling time.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Day | -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| OTU | # | 45 | # | 67 | 0 | 78 | 0 | 10 | 34 | 87 | 0 | 0 | 65 | 74 | 0 | # |

Here we take OTU 1 feature in the first subject for example (As seen in Table S1).

The first row means the sampling time while the second row means the abundance features of OTU1 on a specific day. There are 16 time points while some are missing values and some may be outliers (e.g., 0). For this example, we have 2 parameters with window length of and overlap of 1, we set the small window pieces to be 3 and the overlap to be 1 to traverse the time series sequence, and we can get small pieces as:

[(-5.0, -3), (-3, -1), (-1, 1), (1, 3), (3, 5), (5, 7), (7, 9), (9, 11)]

Parameters are chosen to ensure each divided period contains at least one observation for each subject and maximizes the temporal resolution. Then we get 8 new feature values for small time window pieces:

1st: [#, 45, #]. Missing 2 values, therefore directly 45 for the first window piece.

2nd: [#, 67, 0]. (67+0)/2

3rd: [0, 78, 0]. (0+78+0)/3

….

Missing values do not count for averaging, but 0 value counts. After that, we obtained 8 observations at 8 window pieces for each subject. The before-after comparison can be seen in Figure S5. Simply linear interpolation is not good since the David dataset is the most even and the unit of the time point is 1 day, but for the other two datasets is 50 days and even 100 days, therefore there are many redundant observations. Besides, the utilized method can make the time series data smoother. For David dataset, the window length is set to be 3 and the overlap is set to be 1. For Bokulich dataset, the window length is set to be 60 and the overlap is set to be 30. For Karelia dataset, the window length is set to be 400 and the overlap is set to be 300. The operation was conducted for each subject in the dataset and finally, we obtained an OTU matrix measured over time points. For the task of David dataset, each subject has 308 OTU features with 8 observations; for both tasks of Bokulich dataset, each subject has 117 features with 11 observations; for three tasks of Karelia dataset, each subject has 417 features with 11 observations.

* 1. Comparison of different imputation methods

To measure the influence of data imputation on the final prediction performance, we conduct the extra experiment based on different data imputation methods and data missing scenarios. We keep the same GRU classification model for all the comparison methods. In Table S2, Baseline, Non-parametric smooth (NPS), Gaussian Processes (GP), and Average imputation (AI) are all the results based on longitudinal microbiome data. Baseline means the prediction results by raw longitudinal data without using any data imputation. For the situation that different subjects may have different numbers of observations, we simply padded the zero vectors to achieve consistent dimensions. NPS *(Chu C K et al. 1995)* is a data imputation method that utilized local linear smoother (LLS) to simulate missing data. GP *(Quiñonero-Candela et al. 2003)* provides the quantification of posterior uncertainty for data imputation which is more flexible than traditional methods. Besides, we also conducted the experiment based on single-point microbiome data. Mean-time-based (MTb), First-time-based (FTb), and End-time-based (ETb) are all based on the single-point non-longitudinal microbiome data, i.e., there is only one single observation for each subject. MT represents the results based on the average abundances of all the observations in a subject. FT and ET represent the prediction results based on the observations of the first-time point and the last time point. AI is our proposed average imputation method. The comparison results can be seen in Table S2.

**Table S2. The comparison results (Average AUC) based on different time series imputation methods**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Baseline | NPS | GP | MTb | FTb | ETb | AI |
| **David** | 0.5100 | 0.8500 | 0.900 | 0.5450 | 0.7320 | 0.8600 | 1.0000 |
| **Bdiet** | 0.5880 | 0.6330 | 0.7090 | 0.6033 | 0.6730 | 0.6940 | 0.7950 |
| **Bdeliv** | 0.6257 | 0.6153 | 0.6237 | 0.6133 | 0.5628 | 0.5280 | 0.7177 |
| **Knat** | 0.6211 | 0.8456 | 0.9345 | 0.8792 | 0.8126 | 0.8750 | 0.9504 |
| **Kegg** | 0.4855 | 0.5480 | 0.5990 | 0.446 | 0.5120 | 0.5872 | 0.6060 |
| **Kige** | 0.5341 | 0.5630 | 0.6120 | 0.5450 | 0.4623 | 0.5627 | 0.7168 |

As we can see from the results that our method is the best than the methods of interest, we believe the reason may be that the missing data is generated by averaging the existing neighboring time points, compared with other methods, the obtained results are more smooth and reliable. On the other hand, the number of observations is small, incurring difficulties in imputation. When it comes to the non-longitudinal microbiome data (MTb, FTb, and ETb), the proposed RNN degenerates to the shallow neural network. Comparing with the results based on longitudinal microbiome data, the prediction performance decreased by 5%-10%, which indicates that the longitudinal microbiome sampling is valuable that can better help in modeling microbiome dynamics over time, and future microbiome studies should focus more on having more longitudinal samples than instead focusing on sampling once a very large number of people. Comparing the results with Baseline and AI, it can be observed that the prediction results dropped a lot without imputation. We attribute that to the extremely uneven sampling of the raw dataset, which posed difficulty in modeling microbiome dynamics. We conclude that in most cases, especially when there are a lot of missing values in the raw dataset, imputing data is a necessary step for human host status inference.

* 1. Comparison of the feature extraction algorithms that are aware of the temporal aspect of the data

FPCA (Functional principal component analysis, *(Burns et al. 2013)*) is proposed to reduce the dimension of multivariate time series data and in this study, we conducted FPCA in a subject-by-subject manner, and an aggregation approach is utilized to generate the overall loading matrix of OTUs. Specifically, for each subject, we implemented FPCA to reduce the dimensions of the OTU features while keeping the same number of observations, then the reduced OTU features for all the observations in each subject are aggregated as a matrix according to their initial temporal order. The parameter of FPCA is set to achieve the best prediction performance.

Moreover, we implement and compare the proposed feature extraction approach with other feature extraction methods on the flatten temporal data combing with MLP as the final classifier. The input of each subject in the dataset is a matrix. Each row in the matrix denotes an observation (time point: day 1, day 2, day 3, …, day n) while each column in the matrix denotes the OTU features. Therefore, the first column means the abundance changes of OTU1 from day 1 to day n and the input for each subject is of multivariate time series data type. For other feature extraction methods, we directly flatten the matrix to be a one-dimensional vector, therefore it incorporates the temporal information explicitly and the comparison is fairly conceived. Since the flatten input is not a time series matrix where MLP will be more suitable than GRU for comparison. Here is the comparison result.

**Table S3. The Comparison of the feature extraction algorithms that are aware of the temporal aspect of the data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **FPCA** | **AE1** | **AE2** | **VT1** | **VT2** | **PCA1** | **PCA2** |
| **David** | 0.35 | 0.45 | 0.485 | 0.55 | 0.625 | 0.8 | 0.8 |
| **Bdiet** | 0.5033 | 0.19 | 0.192 | 0.71 | 0.845 | 0.4433 | 0.788 |
| **Bdeliv** | 0.5133 | 0.305 | 0.3407 | 0.77 | 0.7673 | 0.5133 | 0.725 |
| **Knat** | 0.8799 | 0.4824 | 0.6183 | 0.9256 | 0.9528 | 0.5702 | 0.9655 |
| **Kegg** | 0.646 | 0.4941 | 0.4946 | 0.3494 | 0.4135 | 0.514 | 0.595 |
| **Kige** | 0.445 | 0.5004 | 0.5203 | 0.5758 | 0.6636 | 0.5461 | 0.6863 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **REF1** | **REF2** | **LR1** | **LR2** | **RF1** | **RF2** |
| **David** | 0.65 | 0.995 | 1 | 1 | 0.9 | 0.995 |
| **Bdiet** | 0.56 | 0.7643 | 0.7867 | 0.795 | 0.74 | 0.8627 |
| **Bdeliv** | 0.52 | 0.712 | 0.4567 | 0.7177 | 0.5533 | 0.6437 |
| **Knat** | 0.8728 | 0.9403 | 0.9104 | 0.9504 | 0.8862 | 0.9536 |
| **Kegg** | 0.4762 | 0.5481 | 0.5596 | 0.606 | 0.4651 | 0.6046 |
| **Kige** | 0.5807 | 0.6406 | 0.6266 | 0.7168 | 0.6482 | 0.6684 |

In Table S3, FPCA means the results obtained by functional PCA while the suffix versions 1 and 2 denote the version of the feature extraction methods which use the flatten temporal data (1) or independent pseudo subject (2). As we can see from the results that our method is slightly better than the FPCA and raw flatten methods, we conclude that our proposed methods can choose the regular time series data for each OTU and the dynamitic pattern may be more important than the random features selected by flatten vectors for classification.

In this study, we conducted a series of comparison experiments based on two directions: feature selection and dimension reduction, to compare the effects of different feature extraction algorithms. Feature selection is to select the best features from the raw input. There are three commonly used methods for feature selection: Filter, Wrapper, and Embedded. The filter-based methods are to score each feature according to divergence or relevance and set the threshold or the number of thresholds to be selected to select features. The wrapper-based methods uses the objective function (prediction effect score) to select or exclude several features each time. For embedded-based methods, supervised algorithms such as logistic regression are trained to obtain the weight coefficients of each feature, and features are selected from large to small according to the coefficients. Herein, we utilized the Variance Threshold (VT) and Recursive Feature Elimination (RFE) with respect to the ﬁlter and wrapper-based methods for feature selection. VT can remove features with a low variance while RFE utilizes Logistic Regression as the basic classiﬁer since it trains faster and robust to small data noises. Besides, L1-based feature selection and Tree-based feature selection are chosen for embedded methods. The basic classifier for L1-based model is Support Vector Machine Classifier (SVC) while Random Forest (RF) is utilized as the classifier for Tree-based methods.

Additionally, we also utilize Principal Components Analysis (PCA) and Auto Encoder (AE) for dimension reduction as the benchmark comparison. PCA conducts a linear transformation to transform the original data into a set of linearly independent representations to extract the main features of the input variables. PCA can efficiently remove the noise as well as decrease the computational cost by reducing the feature dimension. AE is utilized to reconstruct the feature presentation and reduce the feature dimensions. AE is an unsupervised deep neural network that can transform the input variables into low dimensional representation. AE is composed of input layer, hidden layer, and output layer. Generally, the numbers of the nodes of the input and output layers should be the same, and both of them should be greater than the number of hidden nodes. The Rectified Linear Unit (ReLU) is used as the activation function in this study since it can increase the nonlinear relationship between the layers of the neural network and make the function converge faster. In particular, we added a sparsity constraint (L1 Regularization) based on AE to construct a sparse autoencoder (SAE) to get the compressed representation of the input without sparse. The sparsity restrictions can make AE elucidate the potential data manifolds even the number of hidden neurons is large. (Adam) is used as the gradient update algorithm, and the output of the latent layer will be directly sent to the classifier. The architecture of AE in our study can be seen in Supplementary Figure S6.

* 1. Implementation details (Parameters)

**For Section 3.2 and 3.3**

For all the utilized DNN/GRU models in this study, the number of hidden units is selected among {512, 256, 128, 64, 32}, and the number of layers is selected among {1, 2, 3, 4, 5}. We choose the final number of hidden units and layers by using an internal 5-fold cross-validation on the training data. The combination of 2 layers with 512 and 256 hidden units respectively gets the best average performance on all the datasets. Therefore, we choose this combination and conduct the ablation experiments as described in Section 3.2. We used ReLU as the activation function at each hidden layer. The learning rate is set to be 0.0001. The optimizer is stochastic gradient descent (SGD) and the momentum is set to be 0.9 to speed up the convergence. Batch size is set to be 1 for all the datasets. The other parameters can be found in Table 2.

For feature extraction methods utilized in this study, no matter for supervised methods (RFE, LR, RF) or unsupervised methods (AE, VT, PCA), the experiments were strictly conducted in each cross-validation iteration and we only trained the feature extraction models based on the training data. Even for supervised methods (which use subject labels), we make sure that the model is trained only with the labels of the training data to avoid data leakage and overestimated bias.

**AE:** Adam is used as the gradient update algorithm. Mean squared error (MSE) is used as the loss function. The number of hidden units is selected to be {n, n/2, n/4}, and the number of layers is selected among {1, 2, 3}. we added a sparsity constraint (L1 Regularization) based on AE to construct a sparse autoencoder (SAE) to get the compressed representation of the input without sparse. The L1 regularization penalty is set to be 0.0001. Other parameters are set to be the same as the above-mentioned DNN-based models.

**VT:** Threshold of VT as 0.99 to remove features that more than 99% are either one or zero in all the observations.

**PCA:** For PCA, we use a threshold (0.99) to keep 99% information while reducing the feature dimensions.

**RFE:** RFE utilizes logistic regression as the basic classifier. The inverse regularization parameter *C* is selected among {1e-4, 1e4} with grid search and 5-fold cross-validation.

**L1:** L1-based feature selection, the inverse regularization parameter *C* is selected to among {1e-4, 1e4} with grid search and 5-fold cross-validation.

**RF:** Tree-based feature selection using random forest. The number of estimators is selected among {10, 20, 50, 100, 500} with grid search and 5-fold cross-validation.

**MITRE:** Keep all the parameters to be the same as the raw paper.

**RNN:** Keep all the parameters to be the same as the raw paper.

**SVM:** The inverse regularization parameter C is selected among {1e-4, 1e4} with grid search and 5-fold cross-validation.

**KNN:** KNN, the k value was chosen between 2 and the sample size for the minority class.

**LR:** the inverse regularization parameter C is selected to among {1e-4, 1e4} with grid search and 5-fold cross-validation.

**RF:** The number of estimators is selected among {10, 20, 50, 100, 500} with grid search and 5-fold cross-validation.

**MLP:** The same as aforementioned.

**GRU:** The same as aforementioned.

* 1. Results based on shotgun metagenomic sequencing and HMP data

**Table S4. Comparison results (Average AUC (Standard deviation)) with recently published predictors and baseline classiﬁers on shotgun metagenomic datasets. The introduction of each algorithm can be found in the main text.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Datasets** | **MITRE** | **RNN** | **SVM** | **KNN** | **LR** | **RF** | **MLP** | **GRU** | **FE\_GRU** |
| **BrooksB** | 0.8235 (0.0753) | 0.6372  (0.5174) | 0.7452  (0.0160) | 0.6346  (0.0532) | 0.8173  (0.0124) | 0.8526  (0.0121) | 0.8632  (0.0558) | 0.9326  (0.0055) | **0.9438**  **(0.0021)** |
| **HallAB** | 0.7932  (0.1256) | 0.5467  (0.0162) | 0.7213  (0.0172) | 0.7235  (0.0163) | 0.7539  (0.0235) | 0.8123  (0.0059) | 0.8123  (0.0032) | **0.8354**  **(0.0023)** | 0.8269  (0.0017) |
| **HeitzBA** | 0.9324  (0.1202) | 0.7324  (0.0389) | 0.6548  (0.0201) | 0.8539  (0.0355) | 0.8246  (0.0138) | 0.9221  (0.0032) | 0.9455  (0.0048) | 0.9542  (0.0122) | **0.9785**  **(0.0085)** |
| **RaymondF** | 0.9145  (0.0734) | 0.6349  (0.1202) | 0.6835  (0.0502) | 0.7523  (0.0182) | 0.9432  (0.0180) | 0.8973  (0.0028) | 0.9152  (0.0021) | 0.9328  (0.0015) | **0.9540**  **(0.0009)** |
| **VincentC** | 0.7326  (0.0186) | 0.6213  (0.0867) | 0.6741  (0.0238) | 0.7543  (0.0234) | 0.7521  (0.0152) | 0.6845  (0.0065) | 0.7892  (0.0075) | 0.7654  (0.0072) | **0.7986**  **(0.0026)** |
| **ShaoY** | 0.8575  (0.0135) | 0.6511  (0.0018) | 0.7324  (0.0127) | 0.6578  (0.0324) | 0.6437  (0.0123) | 0.5634  (0.0024) | 0.8619  (0.0017) | 0.8732  (0.1472) | **0.8941**  **(0.0021)** |
| **HMPibdmdb** | 0.7355  (0.0018) | 0.6842  (0.0216) | 0.6312  (0.0156) | 0.6432  (0.0012) | 0.7452  (0.0466) | 0.7829  (0.0329) | 0.7465  (0.0013) | 0.7659  (0.0782) | **0.7856**  **(0.0013)** |
| **HMPt2d** | 0.7524  (0.0023) | 0.5213  (0.0021) | 0.5654  (0.0265) | 0.5327  (0.0324) | 0.5464  (0.0567) | 0.7690  (0.0172) | 0.7728  (0.0125) | 0.7732  (0.0012) | **0.7912**  **(0.0049)** |

**Table S5. Comparison results (Average AUC (Standard deviation)) with different feature extraction algorithms on shotgun metagenomic datasets. The introduction of each algorithm can be found in the main text.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Datasets** | Baseline | AE | VT | PCA | RFE | L1 | RF |
| **BrooksB** | 0.9326  (0.0055) | 0.5568  (0.0253) | 0.9027  (0.0155) | 0.9375  (0.0161) | 0.9438  (0.0253) | **0.9438**  **(0.0021)** | 0.9401  (0.0016) |
| **HallAB** | 0.8354  (0.0023) | 0.6743  (0.0177) | 0.8126  (0.0212) | 0.8210  (0.0157) | 0.8155  (0.0027) | 0.8269  (0.0017) | 0.8354  (0.0328) |
| **HeitzBA** | 0.9542  (0.0122) | 0.5839  (0.0248) | 0.9572  (0.0207) | 0.9628  (0.0177) | 0.9752  (0.0258) | **0.9785**  **(0.0085)** | 0.9678  (0.0124) |
| **RaymondF** | 0.9328  (0.0015) | 0.5210  (0.0237) | 0.9470  (0.0097) | 0.9532  (0.0085) | 0.9440  (0.0019) | **0.9540**  **(0.0009)** | 0.9113  (0.0095) |
| **VincentC** | 0.7654  (0.0072) | 0.4832  (0.0116) | 0.7236  (0.0168) | 0.7942  (0.0230) | 0.7071  (0.0126) | **0.7986**  **(0.0026)** | 0.7712  (0.0176) |
| **ShaoY** | 0.8732  (0.1472) | 0.6273  (0.0125) | 0.8521  (0.0027) | 0.8732  (0.0031) | **0.8952**  **(0.0052)** | 0.8941  (0.0021) | 0.8842  (0.0035) |
| **HMPibdmdb** | 0.7659  (0.0782) | 0.5392  (0.0217) | 0.7742  (0.0051) | 0.7739  (0.0125) | 0.7710  (0.0328) | **0.7856**  **(0.0013)** | 0.7855  (0.0032) |
| **HMPt2d** | 0.7732  (0.0012) | 0.5046  (0.0528) | 0.7823  (0.0024) | 0.7631  (0.0211) | 0.7901  (0.0015) | **0.7912**  **(0.0049)** | 0.7892  (0.0008) |

* 1. Comparison of running time

As far as we know, there is few research in predicting the host status from longitudinal microbiome taxonomic distribution, even for random forest, the performance is based on our data preprocessing and implementation. Therefore, we just compared the published methods MITRE *(Bogart et al. 2019)* and RNN *(Metwally et al. 2019)*. As we can notice that, although that *(Metwally et al. 2019)* is also fast enough, its performance cannot be guaranteed since they utilized zero-padding for inconsistent sampling which introduced extra data noises *(Metwally et al. 2019).* In addition, the pipeline was specially proposed to predict the food allergy. Therefore, its generalization to other tasks remains speculative. Then MITRE was introduced to extract the rules from microbiota temporal data and discover the relationship with its host status *(Bogart et al. 2019).* Although the performance is promising, it can take several days to train the model even for very small datasets. Here is the extra description for running time.

**Table S6. The Comparison of running time**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Study** | **MITRE** | **RNN** | **GRU** | **FE\_GRU** |
| **David** | **45 h** | **7 min** | **5 min** | **6 min** |
| **Bokulich** | **38 h** | **9 min** | **7 min** | **8 min** |
| **Karelia** | **65 h** | **13 min** | **10 min** | **11 min** |

1. Supplementary Figures

**Diagram

Description automatically generated**

**Figure S6. The architecture of AE.**

****

**Figure S7. The comparison results of fine-tuning.**

Graphical user interface, application

Description automatically generated

**Figure S8. The reconstructed losses for all the datasets.**

Reference

David, Lawrence A., et al. "Diet rapidly and reproducibly alters the human gut microbiome." Nature 505.7484 (2014): 559-563.

Bokulich, Nicholas A., et al. "Antibiotics, birth mode, and diet shape microbiome maturation during early life." Science translational medicine 8.343 (2016): 343ra82-343ra82.

Vatanen, Tommi, et al. "Variation in microbiome LPS immunogenicity contributes to autoimmunity in humans." Cell 165.4 (2016): 842-853.

Bogart, Elijah, Richard Creswell, and Georg K. Gerber. "MITRE: inferring features from microbiota time-series data linked to host status." Genome biology 20.1 (2019): 1-15.

Chu C K, Cheng P E. Nonparametric regression estimation with missing data[J]. Journal of Statistical planning and Inference, 1995, 48(1): 85-99.

Quiñonero-Candela J, Roweis S T. Data imputation and robust training with Gaussian processes[J]. 2003.

Burns, Devin M., et al. "Functional principal components analysis of workload capacity functions." Behavior research methods 45.4 (2013): 1048-1057.

Metwally, Ahmed A., et al. "Utilizing longitudinal microbiome taxonomic profiles to predict food allergy via Long Short-Term Memory networks." PLoS computational biology 15.2 (2019): e1006693.

Pasolli E, Schiffer L, Manghi P, et al. Accessible, curated metagenomic data through ExperimentHub[J]. Nature methods, 2017, 14(11): 1023.

Brooks B, Olm M R, Firek B A, et al. Strain-resolved analysis of hospital rooms and infants reveals overlap between the human and room microbiome[J]. Nature communications, 2017, 8(1): 1-7.

Hall A B, Yassour M, Sauk J, et al. A novel Ruminococcus gnavus clade enriched in inflammatory bowel disease patients[J]. Genome medicine, 2017, 9(1): 1-12.

Heintz-Buschart A, May P, Laczny C C, et al. Integrated multi-omics of the human gut microbiome in a case study of familial type 1 diabetes[J]. Nature microbiology, 2016, 2(1): 1-13.

Raymond F, Ouameur A A, Déraspe M, et al. The initial state of the human gut microbiome determines its reshaping by antibiotics[J]. The ISME journal, 2016, 10(3): 707-720.

Vincent C, Miller M A, Edens T J, et al. Bloom and bust: intestinal microbiota dynamics in response to hospital exposures and Clostridium difficile colonization or infection[J]. Microbiome, 2016, 4(1): 1-11.

Shao Y, Forster S C, Tsaliki E, et al. Stunted microbiota and opportunistic pathogen colonization in caesarean-section birth[J]. Nature, 2019, 574(7776): 117-121.

Integrative H M P. The Integrative Human Microbiome Project: dynamic analysis of microbiome-host omics profiles during periods of human health and disease[J]. Cell host & microbe, 2014, 16(3): 276-289.

Bahnsen A C, Aouada D, Stojanovic A, et al. Feature engineering strategies for credit card fraud detection[J]. Expert Systems with Applications, 2016, 51: 134-142.

Nargesian F, Samulowitz H, Khurana U, et al. Learning Feature Engineering for Classification[C]//Ijcai. 2017: 2529-2535.

Zheng A, Casari A. Feature engineering for machine learning: principles and techniques for data scientists[M]. " O'Reilly Media, Inc.", 2018.

Reiman D, Metwally A A, Sun J, et al. PopPhy-CNN: a phylogenetic tree embedded architecture for convolutional neural networks to predict host phenotype from metagenomic data[J]. IEEE journal of biomedical and health informatics, 2020, 24(10): 2993-3001.

Oudah M, Henschel A. Taxonomy-aware feature engineering for microbiome classification[J]. BMC bioinformatics, 2018, 19(1): 1-13.

Fioravanti D, Giarratano Y, Maggio V, et al. Phylogenetic convolutional neural networks in metagenomics[J]. BMC bioinformatics, 2018, 19(2): 1-13.