Association between Gut Microbiome and Parkinson's Disease Revealed by Sparse Learning

A Thesis Submitted to the

College of Graduate and Postdoctoral Studies

in Partial Fulfillment of the Requirements

for the degree of Master of Science

in the Department of Mathematics and Statistics

University of Saskatchewan

Saskatoon

By

Man Chen

Permission to Use

Abstract

Background: Many studies indicate that the human gut microbiota is likely to have connections with Parkinson disease (PD). Based on these indications, this thesis explores the association between PD and human gut microbiota, from a statistical machine learning perspective. With the purpose of identifying the association between PD and gut microbiota, we assess the predictivity of microbial operational taxonomy units (OTUs) that are extracted from participants' gut.

Methods: We use linear support vector machine (SVM) and logistic regression combined with L_1 penalty and elastic-net penalty, to identify informative OTUs for PD. L_1 penalty is able to do shrinkage for features, which effectively implements feature selection by setting the coefficients of non-significant variables to be zero. Conversely, coefficients with larger absolute values indicate that the OTUs are more closely related to PD. Elastic-net penalty is capable of grouping correlated variables. Under these two penalties, SVM and logistic regression can achieve good predictive results as well as feature selection. In order to make full use of dataset and to avoid overfitting, we run models with Leave-one-out cross-validation (LOOCV). There are tuning parameters, λ for each regularization. After running models with LOOCV, we choose the optimal λ for each model, using test error rate as the criterion.

Results and Conclusions: We analyze the performance of each optimal model , by calculating and understanding evaluation metrics of these models. Then, we find that for our dataset, logistic regression with L_1 penalty has the best performance. R^2_{ER}, R^2_{AMLP}, AUC and AUPR of logistic regression with L_1 are 43.9%, 25.7%, 0.8259 and 0.8788. We focus on the selected OTUs based on coefficients generated by models, and to the ranking of OTUs, according to their level of relevance to PD. Then, we find that some OTUs selected by logistic regression with L_1 have been identified in previous studies of micro-organisms, including Lactobacillus, Roseburia, Bluatia, Akkermansia and Bifidobacterium. We also explore predictive performances of logistic regression with elastic-net and regularized SVM, and then focus on OTUs selected by these models. The OTUs selected by these models also overlap with those identified by previous researchers.

Acknowledgements

I would like to first express my gratitude to my supervisor, Prof. Longhai Li, for his continuous support and inspiring guidance, without which I cannot complete my thesis successfully. I'm also grateful to Department of Mathematics and Statistics, for financial and academic support during my master program.

I would like to thank Dr. Lloyd Balbuena, for his comments on my thesis.

Last but not least, I would like to express many thanks to my parents.

Contents

List of Tables

List of Figures

List of Abbreviations

PD	Parkinson Disease
OTU	Operational Taxonomy Unit
SVM	Support Vector Machine
CV	Cross Validation
LOOCV	Leave One Out Cross Validation
AMLP	Average of Minus Log Predictive Probabilities
ROC	Receiver Operating Characteristic
AUC	Area Under the ROC Curve
PRC	Precision Recall Curve
AUPR	Area Under the Precision Recall Curve
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
KKT	Karush Kuhn Tucker

Chapter 1

Introduction

1.1 Background and Motivation

Parkinson disease (PD) was first described by Dr. James Parkinson in 1817. At that time, he called this disease “shaking palsy”. PD is a long-term progressive disorder of central nervous system. Among all neurodegenerative disorders, PD is relatively common. According to the information from Parkinson Disease Foundation [1], there will be more than 1 million American PD patients by 2020, and more than 10 million people worldwide are living with PD. Generally, the incidence of PD increases with age. Most PD patients are aged people, but an estimated 4% of people with PD are diagnosed before age 50 [1].

PD patients suffer from a movement disorder. In the early stage of PD, the most obvious symptoms are tremor, muscle stiffness, rigidity in facial expressions, slow movement and difficulty with balance. Gradually, with the progression of PD, some severe symptoms occur. Patients in an advanced stage tend to suffer from a series of emotional and psychological behaviors, such as depression, anxiety, difficulty with attention and memory [2][3]. Besides, as the condition becomes severer, other problems such as sleep disturbance and dementia also arise [3].

The exact cause of PD is still unclear. However, both genetic factors and environmental factors are believed to be main causes. It has been proposed recently that PD may have viral origins [4]. Scientists also find some biomarkers of PD. The low-level dopamine and the accumulation of an abnormal protein, Lewy bodies, have been found as indicators of PD. Usually, the death of cells in a specific area of the midbrain cause the motor symptoms of the disease and inadequate dopamine in this area [2]. Even though PD is associated with the accumulation of Lewy bodies [4], the exact reason why these cells die is not well understood.

Without knowing the causes of PD, scientists recognize several risk factors for PD. The probability that male get PD is 1.5 times higher than that of female. Compared with Whites, African Americans and Asians are less likely to get PD. PD usually happens between the ages of 50 and 60. Only 5-10 % patients are younger than 40 [1][5]. Heredity is also a potential factor linked to PD, which means people having relatives with PD are more likely to suffer from PD. Head injury can also increase the risk of getting PD [4].

Recently, the relationship between PD and human gut microbiome has drawn researchers' attention. Some gastrointestinal symptoms, such as drooling, dysphagia, constipation and defecatory dysfunction, have often been observed as precursor signs of PD [6]. Gastrointestinal inflammation also occurs in PD patients [7]. These phenomena have led to speculation that there is a link between PD and the gut, and that the human gut microbiota can be a potential source of novel therapeutics. Thus researchers have embarked on more studies regarding the relation between PD and the gut microbiome.

The gut microbiome is the ecosystem of microorganisms, bacteria, viruses, protozoa, fungi, and their collective genetic materials present in the gastrointestinal tract. The gut microbiota consists of all the bacteria, both commensal and pathogenic, residing in the gastrointestinal tract. There are tens of trillions of microorganisms in human gut, including more than 1000 bacterial species [8][9]. The development of bioinformatics and sequencing technologies in recent decades has opened to researchers more opportunities to explore the gut microbiome. Investigations of the human gut microbiota have been extended beyond classical infectious diseases. Studies have shown changes in the gut microbiota during obesity, diabetes, metabolic disorder and other diseases [10–13]. A well-balanced microbiome plays an important role in immune system preventing some specific disease [12][13].

Furthermore, various studies have indicated links between the gut microbiota and neurodegenerative diseases, such as PD. Intestinal microbiome is altered in PD and is related to motor phenotype [14]. Microbial dysbiosis happens during PD and can lead to inflammation-induced misfolding of α-synuclein and development of PD pathology [15]. Compared with healthy controls, PD patients tend to have less bacterial phylum Bacteroidetes and Prevotellaceae, while Enterobacteriaceae are more abundant in fecal samples from PD patients [16]. Changes in the abundance of 9 genera and 15 species of microorganisms happen during the process of PD [17], triggering local inflammation and consequent aggregation of α-synuclein and Lewy bodies, which are typical features of PD.

In summary, a variety of studies have shown the evidence that PD may be related to the gut microbiome. On this basis, this thesis explores further the association between PD and microbiome, from a statistical machine learning perspective. We attempt to find the decisive microbiome patterns that are highly related with PD. When it comes to predictive analysis of individuals' phenotype based on microbial data, finding ideal models is challenging but important. An optimal predictive model can not only provide a diagnostic tool for PD, but also helps to uncover the biological mechanism for PD.

1.2 Methods

Typically, microbial abundance is quantified by Operational Taxonomy Unit (OTU). The OTU dataset is high-dimensional and over-dispersed , with excessive zeros. Effective predictive models that can deal with OTU data properly and do feature selection are necessary. To explore the association between PD phenotype and gut microbiome, we use logistic regression and linear support vector machine (SVM) combined with LASSO regularization and elastic-net regularization. LASSO is able to conduct feature selection after a process of shrinkage. By adding L_1 penalty to loss function, this regularization enables the coefficients of non-significant features to be zero. Elastic-net regularization is a linear combination of L_1 penalty and L_2 penalty. When there is a group of correlated variables, LASSO often keeps one of them and ignores others, while elastic-net can consider all correlated variables in the group. In LASSO and elastic-net regularizations, different tuning parameters λs have different predictive performances. Hence, we need to find the most optimal λ for each model.

It is incorrect to use the same dataset for learning the parameters of a predictive model and testing predictive performance, because by doing this, predictive accuracy may not generalize to yet-unseen data. This phenomena is called overfitting. In order to avoid overfitting, cross validation (CV) is conducted to learn the parameters of our predictive models. Leave one out cross validation (LOOCV) is a special case of CV, which can make full use of all samples in dataset. In this thesis, we apply LOOCV to regularized classification models, for the purpose of finding the optimal λ. Test error rate is the main criterion for choice of λ. According to test error rates, we choose the best λ of each model. Then, in each optimal model, variables with non-zero coefficients are chosen as significant features. The average of minus log predictive probabilities (AMLP), receiver operating characteristic (ROC) curves and Precision Recall curves (PRC) are alternative metrics for evaluation of predictive models.

1.3 Summary of Results

After applying regularized logistic regression and SVM, we obtain the error rates of these models with different λ. A λ associated with minimum error rate is chosen as the optimal one. For optimal logistic models with L_1 penalty and elastic-net penalty, error rates are 0.223 and 0.238, respectively; AUC are 0.8259 and 0.8202, respectively; AUPR are 0.8788 and 0.8794; AMLP are 0.515 and 0.506, respectively. For optimal SVM with L_1 penalty and optimal SVM with elastic-net penalty, error rates are 0.278 and 0.275; AUC are 0.7773 and 0.7762; AUPR are 0.8322 and 0.8314; AMLP are 0.550 and 0.562. The R^2_{ER}, R^2_{AMLP} of logistic regression with L_1 are 43.9%, 25.7%, respectively, and for logistic models with elastic-net, they are 40.1%, 27.0%, respectively. For SVM with L_1, they are 30.1%, 20.7%, and for SVM with elastic-net, they are 30.8%, 18.9% respectively. The overall performance of logistic regression is better than that of SVM, and logistic regression with L_1 is the best one.

In different folds of LOOCV, coefficients of OTUs are different. Thus, for each OTU, we find the median of absolute value of its all coefficients. After ranking the median, the OTUs associated with a median with absolute value exceeding zero are considered as significant features related to PD. The boxplot is a proper tool to visualize the relative abundance of OTUs between PD patients and control cases. Finally, we discuss similarities and differences between our results and other published studies.

1.4 Outline

Chapter 2 is an explicit description of predictive models for OTU dataset related to PD, including logistic regression, SVM, LASSO regularization, elastic-net regularization, and cross validation. We can see how regularizations help models to do feature selection. We also introduce cross validation and LOOCV. Evaluation metrics are also described explicitly in Chapter 2, including error rate, AMLP, ROC, AUC, PRC, and AUPR. These metrics provide the comprehensive evaluation of our models. In Chapter 3, we describe our real dataset. The raw dataset is 16r RNA sequencing data. The tool to extract OTU from sequencing data is also introduced in Chapter 3. In Chapter 4, we report our results. The performances of our models are discussed in this chapter. Moreover, we also present the decisive OTUs selected by our model. Discussion on the comparison of our results with other researchers' results is needed, to assess how credible our results are. The conclusion is in Chapter 5, in which we also discuss the limitations of our work and future work.

Chapter 2

Methodology

2.1 Models

2.1.1 Data Structure

Table 2.1: Overview of Data for Models

	OTU_1	OTU_2	.....	OTU_q	Age	Sex	Parkinson	Total reads
$Sample_1$	$otu_1^{(1)}$	$otu_1^{(2)}$	.....	$otu_1^{(q)}$	age_1	sex_1	Y_1	$T_1 = \sum_{j=1}^{q} otu_1^{(j)}$
......	......	......	.....	......	......	......	......	......
......	......	......	.....	......	......	......	......	......
$Sample_n$	$otu_n^{(1)}$	$otu_n^{(2)}$	.....	$otu_n^{(q)}$	age_n	sex_n	Y_n	$T_n = \sum_{j=1}^{q} otu_n^{(j)}$

In this thesis, we explore the association between PD and the human gut microbiome, by establishing models for OTU variables, the person level variables and the phenotype. The person level variables are age and sex. Phenotype indicates whether a subject is the PD case or not.

OTU, the operational taxonomic unit, is an operational definition for classifying groups of closely related organisms. Originally, this notation was proposed by Robert R. Sokal and Peter H. A. Sneath [18]. At first, "Operational Taxonomic Unit" was a pragmatic definition to group organisms according to similarity. Nowadays, OTU is also widely seen in different backgrounds. The meaning of OTU has been extended into a different context. Presently, OTU also refers to clusters of microorganisms, grouped by DNA sequence similarity to a specific taxonomic marker gene. Assigning OTU is the process to cluster similar sequences based on a defined similarity threshold. Sequences that are similar at or above the threshold level are grouped into a taxonomic unit in the sequence collection. After assigning OTU, we obtain an OTU dataset. OTU variables can be denoted by $(otu_i^{(1)}, otu_i^{(2)}, .., otu_i^{(q)})$. $otu_i^{(j)}$ means that, for the i_{th} subject, the number of sequences that are grouped into the j_{th} OTU. The OTU dataset consists of non-negative integers, including lots of zeros.

2.1.2 Logistic Regression

Logistic regression, a kind of generalized linear model, is used to predict the probabilities of certain classes or events, such as pass/fail, healthy/sick, negative/positive. Logistic regression is widely applied to analyze data, especially in medical areas [19–23].

Logistic regression can be extended to multiple classes. Our study involves a classification of Parkinson/non-Parkinson, so we discuss the binary case of logistic regression. Suppose for each person, we have a series of features including OTU features, age and sex, denoted by $X_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(p)})^T$. $\boldsymbol{\beta} = (\beta_1, \beta_2, ..., \beta_p)^T$ is the vector of coefficients associated with features. The response variable Y indicates a Parkinson case or not, with 1 denoting PD, 0 denoting a healthy case. Then, the prediction for the probability of getting PD is:

$$P(y_i = 1) = \sigma(X_i) = \frac{e^{\beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + ...\beta_p x_i^{(p)}}}{1 + e^{\beta_0 + \beta_1 x_i^{(1)} + \beta_2 x_i^{(2)} + ...\beta_p x_i^{(p)}}}. \tag{2.1}$$

The logistic function $\sigma(X)$ is also called sigmoid function. Its negative log-likelihood function is:

$$l(\beta) = -\sum_{i=1}^{n} \log(P_\beta(Y_i|X_i)) = -\sum_{i=1}^{n}(y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_i^{(j)}) - \log(1 + e^{\beta_0 + \sum_{j=1}^{p} \beta_j x_i^{(j)}})). \tag{2.2}$$

Its loss function is:

$$J(\beta) = -\frac{1}{n}\sum_{i=1}^{n}(y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_i^{(j)}) - \log(1 + e^{\beta_0 + \sum_{j=1}^{p} \beta_j x_i^{(j)}})). \tag{2.3}$$

We obtain the coefficients of a logistic regression model by minimizing its loss function.

2.1.3 Linear Support Vector Machine

Support vector machine is a supervised machine (SVM) learning algorithm that finds a hyperplane to separate data points into two groups. A hyperplane is a subspace of dimension $p-1$ in a p-dimensional space. Particularly, in a 2D space, a hyperplane is a line, and in a 3D space, it is a plane. For SVM, the number of dimensions, p, is the number of features. Given dataset (X_i, Y_i) for $i = 1...n$, with $X_i = (x_i^{(1)}, x_i^{(2)}, ..., x_i^{(p)})^T \in R^p$ and $Y_i \in \{-1, 1\}$, SVM is a classification denoted by

$$\begin{cases} \hat{y}_i = +1, & \text{if } f(X_i) \geq 0 \\ \hat{y}_i = -1, & \text{if } f(X_i) < 0 \end{cases} \quad i = 1, 2, 3...n, \tag{2.4}$$

where $f(X) = \boldsymbol{w}^T X + b$ is a hyperplane. Here, $y = +1$ denotes PD and $y = -1$ denotes healthy control. Figure 2.1 is a graphical illustration of SVM [24].

2.1.3.1 Hard Margin for Linearly Seperable Data

A good SVM model can not only correctly separate points into two sides but also separate them as stable as possible, which means points in two sides are divided by a clear distance. The distance between hyperplane and the closest point should be as large as possible. This idea is called SVM with maximum margin. Maximization of the margin distance ensures the stability and reinforcement, so that data can be linearly separable with more confidence. Based on the idea of separating data points as accurately as possible with a maximization of the margin, SVM is formulated as:

$$\begin{aligned} &\underset{\boldsymbol{w},b}{\text{maximizing}} \ \frac{2}{\|\boldsymbol{w}\|}, \\ &\text{subject to } y_i(\boldsymbol{w}^T X_i + b) > 1, i = 1, 2, ..., n. \end{aligned} \tag{2.5}$$

The optimization problem in 2.5 is equivalent to

$$\begin{aligned} &\underset{\boldsymbol{w},b}{\text{minimizing}} \ \frac{1}{2}\|\boldsymbol{w}\|^2, \\ &\text{subject to } y_i(\boldsymbol{w}^T X_i + b) > 1, i = 1, 2, ..., n. \end{aligned} \tag{2.6}$$

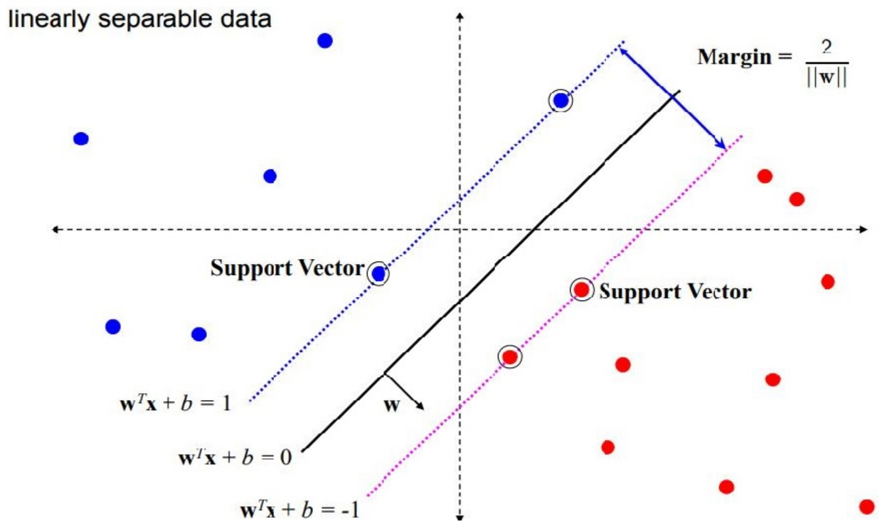

linearly separable data
Margin = $\frac{2}{||\mathbf{w}||}$
Support Vector
Support Vector
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b = -1$

Figure 2.1: Overview of SVM

2.1.3.2 Soft Margin for Non-Separable Data

When we deal with a real dataset, data points are not always separable. In general, data are not always linearly separable. Thus, the soft margin is proposed for fault tolerance. The intuitive idea of soft margin is to tolerate a certain number of points being misclassified, and to control the trade-off between maximization of the margin and minimization of misclassifications. So, the slack variable ξ is introduced to implement soft margin. The more flexible SVM with soft margin is formulated as:

$$\begin{aligned}&\underset{\boldsymbol{w},b}{\text{minimizing}}\ \frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{n}\xi_i\\&\text{subject to } y_i(\boldsymbol{w}^T X_i + b) \geq 1-\xi_i,\\&\xi_i \geq 0, i=1,2...,n.\end{aligned} \tag{2.7}$$

Each variable ξ_i, $i = 1, 2, ...n$, is called a slack variable for each data point. All slack variables are constrained to be no less than 0, $\xi_i \geq 0$. Data points with $\xi_i = 0$ are correctly classified, and they are on the correct side of the margin or just on the margin. Data points with $0 < \xi_i \leq 1$ are located within the margin but still on the correct side of hyperplane. Points with $\xi_i > 1$ are misclassified into the wrong side of hyperplane. Figure 2.2 illustrates values of slack variables of corresponding data points [25]. $C > 0$ is a regularization term. C

can balance the trade-off between minimizing training errors and controlling model complexity. It is a way to control overfitting. A smaller C results in a wider margin, while a larger C results in a narrow margin. When $C = \infty$, the soft margin becomes the hard margin.

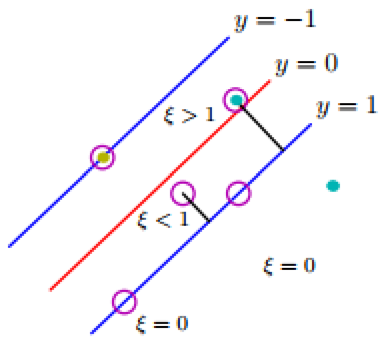


Figure 2.2: Illustration of Slack Variables

2.1.3.3 Hinge Loss Function

From a SVM model like:

$$\begin{cases} \hat{y}_i = +1, & \text{if } \boldsymbol{w}^T X_i + b \geq 0 \\ \hat{y}_i = -1, & \text{if } \boldsymbol{w}^T X_i + b < 0 \end{cases} \quad i = 1, 2, 3...n, \tag{2.8}$$

$y_i(\boldsymbol{w}^T X_i + b) \geq 0$ means the point y_i is classified correctly, so the "loss" of this case is 0. Otherwise, $y_i(\boldsymbol{w}^T X_i + b) < 1$ indicates the classification is imperfect and so the "loss" is defined as $1 - y_i(\boldsymbol{w}^T X_i + b)$. The "loss" of point i is $max(0, 1 - y_i(\boldsymbol{w}^T X_i + b))$. Therefore, the loss function of SVM is denoted by:

$$L_{hinge\ loss} = \sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{w}^T X_i + b)) = \sum_{i=1}^{n} [1 - y_i(\boldsymbol{w}^T X_i + b)]_+. \tag{2.9}$$

The definition of $[x]_+$ is as the following:

$$[x]_+ = \begin{cases} x & x > 0 \\ 0 & x \leq 0. \end{cases}$$

This loss function is termed as Hinge Loss. Alternatively, Function 2.7 can be written equivalent as

$$\text{minimizing}[L_{hinge\ loss} + \lambda \|\boldsymbol{w}\|^2]. \tag{2.10}$$

2.1.4 LASSO Regularization

The least absolute shrinkage and selection operator (LASSO) is a famous algorithm for regularization and feature selection. By adding the L_1 penalty into the loss function, LASSO can set a constraint on the sum of the absolute values of coefficients. The sum needs to be less than an upper bound. Thus, LASSO applies a shrinkage process that penalizes the coefficients of features, setting some of them to zero [26]. LASSO regularization is also named as L_1 regularization. For Logistic Regression, LASSO regularization can be conducted by penalizing the loss function with L_1-norm:

$$J_{LASSO} = -\frac{1}{n}\sum_{i=1}^{n}(y_i(\beta_0 + \sum_{j=1}^{p}\beta_j X_i^{(j)}) - \log(1 + e^{\beta_0 + \sum_{j=1}^{p}\beta_j X_i^{(j)}})) + \lambda\|\boldsymbol{\beta}\|_1 \tag{2.11}$$

The LASSO estimator of Logistic Regression is defined as:

$$\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}}[-\frac{1}{n}\sum_{i=1}^{n}(y_i(\beta_0 + \sum_{j=1}^{p}\beta_j X_i^{(j)}) - \log(1 + e^{\beta_0 + \sum_{j=1}^{p}\beta_j X_i^{(j)}})) + \lambda\|\boldsymbol{\beta}\|_1]. \tag{2.12}$$

$\|\boldsymbol{\beta}\|_1$ is L_1 norm of $\boldsymbol{\beta}$, equal to $\sum_{j=1}^{p}|\beta_j|$. β_0 is the intercept item, which is not included in the process of penalizing.

Similarly, LASSO estimator of linear SVM is defined as:

$$\underset{b,\boldsymbol{w}}{\operatorname{argmin}}[\sum_{i=1}^{n}[1 - y_i(\boldsymbol{w}^T X_i + b)]_+ + \lambda\|\boldsymbol{w}\|_1]. \tag{2.13}$$

L_1 penalty can shrink the coefficients associated with less important features into exactly zeros. In our study, OTUs with non-zero coefficients are treated as significant OTUs. λ is the tuning parameter, controlling the strength of the penalty. When λ is sufficiently large, then strength of penalty becomes more intensive, resulting in more coefficients being forced to be zero. Intuitively, after imposing a LASSO penalty, the higher the absolute value of a coefficient is, the more significant the associated variable is.

2.1.5 Elastic-net Regularization

The elastic-net penalty is a combination of a L_1-norm penalty with a L2-norm penalty:

$$\lambda_1\|\boldsymbol{\beta}\|_1 + \frac{\lambda_2}{2}\|\boldsymbol{\beta}\|_2^2, \tag{2.14}$$

where $\|\boldsymbol{\beta}\|_2^2$ is $\sum_{j=1}^{p} \beta_j^2$, called L2 norm. λ_1 and λ_2 are the tuning parameters. In a package in R [27], Function 2.14 is denoted by

$$\lambda[\alpha\|\boldsymbol{\beta}\|_1 + (\frac{1-\alpha}{2})\|\boldsymbol{\beta}\|_2^2]. \tag{2.15}$$

Here $\alpha = \frac{\lambda_1}{\lambda_2+\lambda_1}$ and $\lambda = \lambda_2 + \lambda_1$. Then α is the mixing parameter and λ is the tuning parameter. When $\alpha = 1$, the regularization becomes L_1 regularization, and when $\alpha = 0$, it is ridge regularization only with L_2 penalty. In R, we can choose an appropriate value of α according to our dataset, and then choose optimal tuning parameter.

In elastic-net regularization, the L_1 norm penalty performs feature selection, whereas the L_2 norm penalty allows correlated features to be selected together. With the L_2 penalty, highly related features tend to get similar coefficients [28]. This is called the grouping effect. In our case, PD is characterized by patterns of gut microbial data. Microbial data has high-dimensional OTU features. It is possible that some OTUs are highly correlated. With the grouping effect of elastic-net regularization, those correlated OTU features will be selected together, and non-zero coefficients will become more interpretable, by inferring what is common among selected features.

So, elastic-net estimators of logistic regression and linear SVM are:

$$\underset{\beta_0,\boldsymbol{\beta}}{\operatorname{argmin}}[-\frac{1}{n}\sum_{i=1}^{n}(y_i(\beta_0 + \sum_{j=1}^{p}\beta_j X_i^{(j)}) - \log(1+e^{\beta_0+\sum_{j=1}^{p}\beta_j X_i^{(j)}})) + \lambda(\alpha\|\boldsymbol{\beta}\|_1 + (\frac{1-\alpha}{2})\|\boldsymbol{\beta}\|_2^2)] \tag{2.16}$$

and

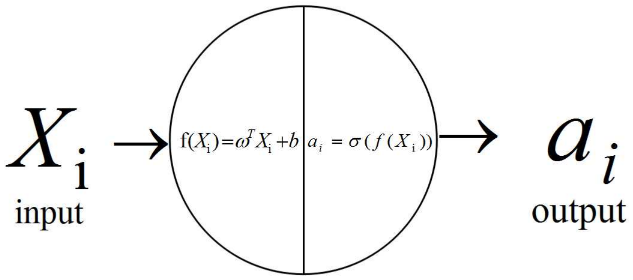
$$\underset{b,\boldsymbol{w}}{\operatorname{argmin}}[\sum_{i=1}^{n}[1 - y_i(\boldsymbol{w}^T X_i + b)]_+ + \lambda(\alpha\|\boldsymbol{w}\|_1 + (\frac{1-\alpha}{2})\|\boldsymbol{w}\|_2^2)]. \tag{2.17}$$

From the function 2.10, we can see that the original maximum-margin SVM model has a built-in L2 regularization. Algorithms to solve out Functions 2.12, 2.13, 2.16 and 2.17 were developed by Trevor Hastie and Rob Tibshirani [26][29].

2.1.6 Probabilistic Prediction with SVM's Results

After training SVM models, the outputs consist of relative distance of each observation, $\boldsymbol{w}^T X_i + b$, and the corresponding predicted label. The original SVM model does not generate the probability of each observation getting PD. However, we need probabilistic results to calculate some comprehensive evaluation metrics, which are intensively used for the further comparison of models. Hence, transforming outputs of SVM into probabilistic values is necessary. We use logistic regression, with the relative distance as the input feature, and the phenotype as the response variable. From Figure 2.3, we can see that this process is similar to a neural network with one hidden layer, the layer has only one neuron unit, and the activation function is the sigmoid function. Here, the parameter vector $\boldsymbol{w}$ and intercept parameter b are from the outputs of SVM.

Figure 2.3: Overview of The Transformation Process



2.1.7 Transformation of OTU Data

In microbiome analysis, people always assume that the phenotype can influence the relative composition or relative abundance of OTUs, rather than original counts of OTUs. Therefore, when we fit models to OTU dataset, a reasonable transformation for OTU data is necessary, changing count number into information of relative abundance. Here, we choose one variance-stability transformation of the proportion:

$$\widetilde{X}_i^{(j)} = \log(X_i^{(j)} + 1) - \log(\sum_{j=1}^{p} X_i^{(j)} + p). \tag{2.18}$$

2.2 Cross Validation

Cross-validation is a statistical method to evaluate the performance of learning algorithms by dividing the dataset into two segments. One segment is for learning or training the model, while the other one serves as "unknown data", for testing the model. With cross-validation, we can make full use of the existing dataset to assess the model's ability to generalize independent datasets that were not seen during the process of training models. There are two popular forms of cross validation, k-fold cross-validation and Leave-p-out cross-validation. K-fold cross-validation means randomly partitioning all samples into k equal-sized and non-overlapping parts. One of k parts is regarded as the validation data for testing the model, and the remaining $k-1$ parts are used as the training set. K-fold cross-validation process repeats k times, with each of the k subsets being used once as the validation set.

Leave-p-out cross-validation means using p observations as the validation set and all remaining observations as the training set. This process repeats on all ways and iterates for each p points of dataset. Leave-p-out cross-validation trains and validates a model C_p^n times. Here, n denotes the number of observations. When $p = 1$, we obtain Leave-one-out cross-validation (LOOCV), which is a particular case of Leave-p-out cross-validation. Besides, LOOCV is also a particular case of K-fold cross-validation with $K = n$. LOOCV trains and validates a model $C_1^n = n$ times, iterating for each data point. Figure 2.4 shows the process of LOOCV [30]. The computation time of LOOCV is $C_1^n = n$. It is less than C_p^n, when $p > 1$.

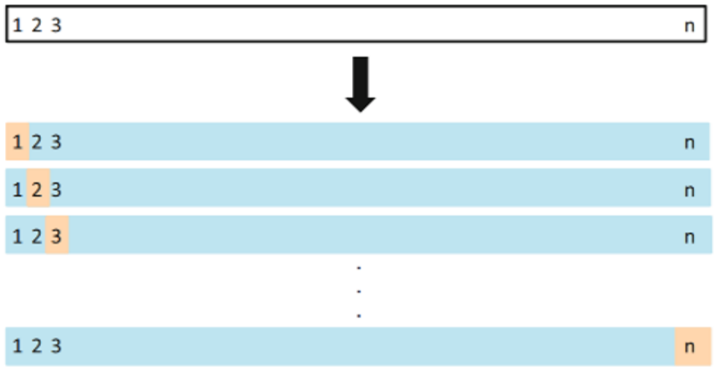


Figure 2.4: Overview of Leave-One Out Cross Validation

2.3 Predictive Metrics

2.3.1 Error Rate and AMLP

In order to evaluate predictive performance, we need to compare our predicted results with the true cases, and to rely on several evaluation criteria. One of the criteria is error rate. Error rate means the proportion of wrong predictions, which is defined as:

$$ER = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i), \tag{2.19}$$

where y_i is the observed case of sample i, and $\hat{y}_i$ is predicted case of sample i.

Another metric, the average of minus log predictive probabilities (AMLP) is:

$$AMLP = -\frac{1}{n}\sum_{i=1}^{n} \log(\hat{P}_i(y_i|x_i)). \tag{2.20}$$

AMLP is more sensitive than error rate because it measures not only the correctness of a point estimate $\hat{y}_i$ but also the level of correctness expressed by $\hat{P}_i(y_i|x_i)$.

Both error rate and AMLP should be compared with their baseline values. The baseline is the result of a very intuitive and straightforward solution. Generally, baseline is only a random guess based on the frequency of observed y_i without using any models and predictors. For all observations, the frequency of $y_i = 1$ is denoted by $f_1 = \frac{1}{n}\sum_{i=1}^{n} I(y_i = 1)$, and the frequency of $y_i = 0$ is denoted by $f_0 = \frac{1}{n}\sum_{i=1}^{n} I(y_i = 0)$. Then the baseline error rate is $ER_{(0)} = min\{f_0, f_1\}$ and the baseline AMLP is $AMLP_{(0)} = -[f_0 log(f_0) + f_1 log(f_1)]$. For instance, if there is a dataset with 95% patients and 5% controls, then the baseline error rate is 5% and the baseline AMLP is -[0.05log(0.05)+0.95log(0.95)]. If the test error rate of a predictive model is 6%, it looks like very good with only 0.06 error rate. However, even if we use the random guess without any predictors, we can obtain the baseline error rate, 5%. Therefore, the model is not better than the baseline model. In brief, to assess the level of predictivity of a model, we should compare error rate and AMLP with baseline values. Thus, we define the relative error rate as:

$$R^2_{ER} = \frac{ER_{(0)} - ER}{ER_{(0)}} \tag{2.21}$$

and the relative AMLP as:

$$R^2_{AMLP} = \frac{AMLP_{(0)} - AMLP}{AMLP_{(0)}} \tag{2.22}$$

2.3.2 AUC

The predicted labels of all samples are either positive (P) or negative (N). For a sample whose predicted label is P, if its actual label is also P, then it is called a true positive (TP); if the actual label is N, then it is a false positive (FP). Similarly, a true negative (TN) means both the predicted label and the actual label are N, and false negative (FN) means the predicted label is N, whereas its actual label is P. We can summary these notations by building a confusion matrix:

Table 2.2: Confusion Matrix

		Predicted Labels	
		Class 1	Class0
Actual Labels	Class 1	TP	FN
	Class 0	FP	TN

The receiver operating characteristic curve (ROC) represents the trade-off between sensitivity and specificity. They are defined as:

$$Sensitivity = \frac{the\ number\ of\ TP}{the\ number\ of\ TP + the\ number\ of\ FN} \tag{2.23}$$

$$Specificity = \frac{the\ number\ of\ TN}{the\ number\ of\ TN + the\ number\ of\ FP} \tag{2.24}$$

One minus Specificity is called false positive rate. ROC space is defined by One minus Specificity and Sensitivity as x axis and y axis, respectively, which depicts relative trade-offs between true positives and false positives. The predictive model provides predicted probabilities of being positive for all samples. Suppose c is the threshold, and $\hat{P}_i$ is the predicted probability of a certain sample being positive. If $\hat{P}_i \geq c$, then the predicted label of this sample is P; if $\hat{P}_i < c$ then its predicted label is N. By iteratively using each predicted probability as the threshold, we calculate corresponding sensitivity and specificity. Corresponding to a particular threshold, each point in ROC space is located by sensitivity as y-coordinate and 1-specificity as x-coordinate.

AUC is the area under the ROC curve. The baseline AUC is 0.5, which can be interpreted as a random guess. A prediction having a AUC closer to 1, is considered superior. The theoretical definition of AUC is the probability that a randomly selected actual positive has a higher test result than a randomly selected actual negative [31]. AUC expresses the degree of separability, indicating how much a model is able to distinguish two classes. Higher AUC means a stronger ability to predict labels correctly.

2.3.3 AUPR

Precision recall curve (PRC) represents the trade-off between Precision and Recall. They are defined as:

$$Precision = \frac{the\ number\ of\ TP}{the\ number\ of\ TP + the\ number\ of\ FP} \tag{2.25}$$

$$Recall = \frac{the\ number\ of\ TP}{the\ number\ of\ TP + the\ number\ of\ FN} \tag{2.26}$$

Recall is another name for sensitivity. PRC space is defined by Recall and Precision as x axis and y axis, respectively. Similar with ROC, each point in PRC is positioned by recall as x coordinate and precision as y coordinate. Precision and recall for each point are obtained by using corresponding predicted probability as the threshold. The baseline of PRC is a horizontal line at y=f_1. Here, f_1 is the frequency of positive observations in our dataset. This line divides the precision-recall space into two parts. Curves in the area above the line denote relatively good predictive performance.

AUPR is the area under the precision-recall curve. The best possible value is 1.0. If a predictive model has a perfect AUPR, it means this model can find all positive samples without incorrectly classifying any negative samples to be positive. If we classify all points as P, we will get a perfect recall but a bad precision, and we will achieve a perfect precision but a bad recall, by classifying all points as N. Therefore, we should consider recall and precision together to evaluate a model.

Chapter 3
Dataset

The raw dataset consists of 16s rRNA sequencing data from a case-control study of 327 participants. The full name of 16s rRNA is 16S ribosomal RNA, which is the component of the small subunit of the ribosome in prokaryotic cells. The gene coding for 16s rRNA is referred to 16S rRNA gene. 16S rRNA gene has 9 variable regions (V1–V9). Usually, we do amplicon analysis and sequencing on v3-v4 and v6. Sequences of 16S rRNA gene are widely used in bacterial phylogeny [32] and classification of bacteria. Reasons for its extensive usage are that it is present in most bacteria and microbes [33], that the function of the 16S rRNA gene has proper changes over time [33], and that the 16S rRNA gene is large enough for general information [34] [35]. Sequences and metadata of the dataset we used are available in the European Bioinformatics Institute European Nucleotide Archive, with accession number ERP016332.

Quantitative Insights Into Microbial Ecology-version 1 (QIIME1) is a tool for pre-processing 16s RNA raw sequencing dataset downloaded from EBI website. QIIME1 is a bioinformatics pipeline based on the Linux operating system, specially designed for conducting microbiome analysis for 16s rRNA sequencing data. QIIME1 can perform a series of analyses on raw sequencing data, including demultiplexing, quality filtering, OTU picking, taxonomic assignment, phylogenetic reconstruction, diversity analysis and visualizations. These steps can be conducted by direct commands and scripts [36]. In this thesis, operational taxonomic units (OTUs) are picked based on the closed-reference option, by using the SortMeRNA method [37] against the Green genes 16S rRNA gene sequence database released on August 2013 [38].

The SortMeRNA algorithm can rapidly go through millions of reads, and sort out reads that can match to the reference database. This algorithm searches for many short similar areas between each read and the rRNA database. Each input read is scanned by a sliding window. A window only captures the subpattern of the read. The window keeps sliding until it goes through the whole read. For the subpattern in each sliding window, if the beginning fraction and the ending fraction can be found in the reference database with high frequency, then the remaining part of the subpattern will be further compared with the database. After comparison, if the subpattern is present in the rRNA database, with at most 1 error, then this sliding window is accepted. For a given read and a given window on the read, one error (substitution, insertion or deletion) between the window and the rRNA database is allowed. If the number of accepted windows exceeds a certain cut-off, then the read is kept and allocated into the respective OTU. Here we choose 94% similarity as the similarity threshold, so the output OTUs are in genus level. SortMeRNA algorithm involves a versatile data structure, Burst trie [39], to store RNA database. This data structure enables effective searching and comparing. The original idea of SortMeRNA is explicitly introduced in [37], and then the idea is applied to OTU picking.

From the EBI website, we downloaded 376 samples, of which 28 were blank samples with scientific name “metagenome”, and 348 were with scientific name “human gut metagenome”. In this study we only used the 348 samples labelled as human gut. During the process of picking OTU, 21 samples were excluded from these 348 samples, and then 327 samples were kept. In these 21 exclusions, 3 samples were deleted automatically by QIIME in pick-OTU step. The outputs of “pick OTU” step did not keep these 3 samples. 14 samples were deleted because of their insufficient total reads. A sample whose total reads were less than 5000 was treated as non-informative enough. 5 samples were deleted, because 2 of them had “bad metadata” and 3 of them were “duplicate samples”. In addition, the group of 14 samples with total reads less than 5000 had a overlap with the group of 2 samples with bad metadata. Then, we excluded 21 samples (3+14+5-1=21) from 348 ones. Finally 327 samples were used in the analysis.

Outputs of QIIME1 contain 382 different OTUs (genera) for 327 samples.

Chapter 4

Results

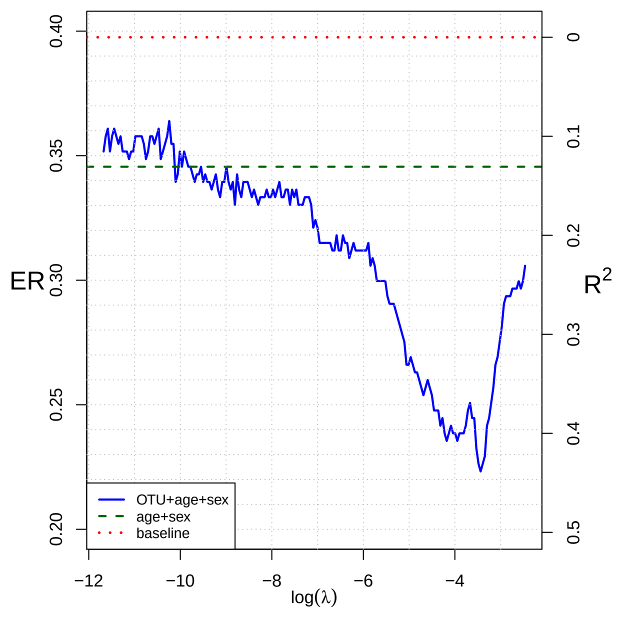
4.1 Evaluation of Predictive Models

4.1.1 Choice of the Optimal Values of Tuning Parameters

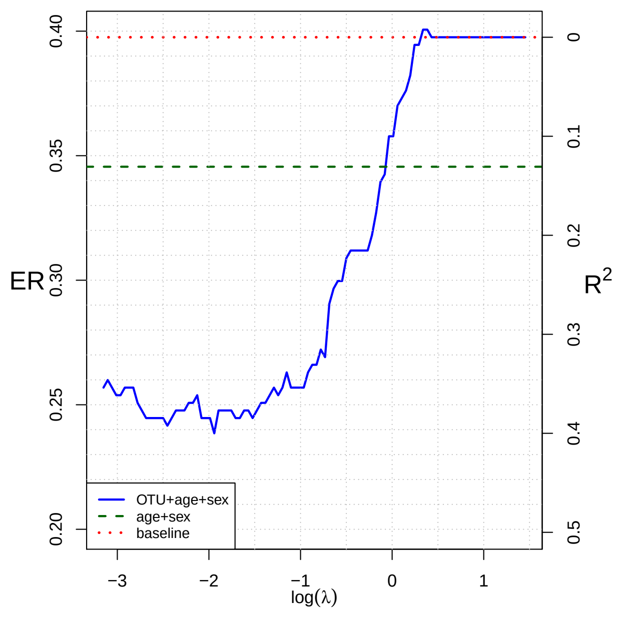
This chapter shows the results of fitting regularized SVM and logistic regression to the dataset mentioned in Chapter 3. Our models include SVM with L_1 penalty, SVM with elastic-net penalty, logistic regression with L_1 penalty, and logistic regression with elastic-net penalty. For these four models, LOOCV is applied to find the optimal values of tuning parameters.

We fit models using R packages glmnet [27] and sparseSVM [40]. Before fitting models, the OTU dataset is transformed by formula $\widetilde{X}_i^{(j)} = \log(X_i^{(j)} + 1) - \log(\sum_{j=1}^{p} X_i^{(j)} + p)$. After transformation, we get the input dataset. Functions in these two packages can do standardization by default. The total number of observations is $n = 327$. With each λ, we select 326 observations as a training set, to construct models. The remaining observations are regarded as the test set. After 327 iterations, we obtain the predicted label of each observation. Comparing with true labels of 327 observations, evaluation metrics can be calculated, including error rate, AMLP AUC, AUPR. As discussed previously, we need predicted probabilities to generate AMLP, ROC curve and PR curve. However, the predicted results of the SVM model are only distances to hyperplane and predicted labels, without probabilities. Thus, to get probabilistic results from SVM, we perform a logistic transformation, using distance as the predictor and true label as response variable. The λ associated with smallest test error rate are is selected as the optimal λ of corresponding model. Consequently, we further explore models with these optimal λs.

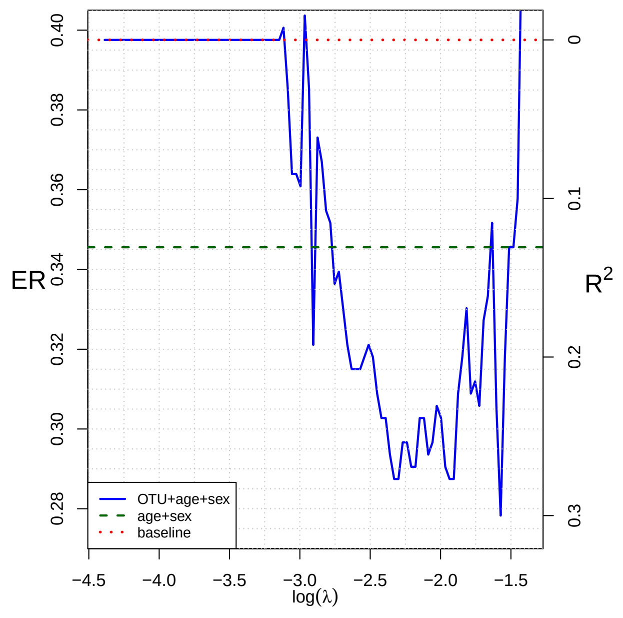
We display a series of figures for visualizing results. For conciseness, we use LR to denote



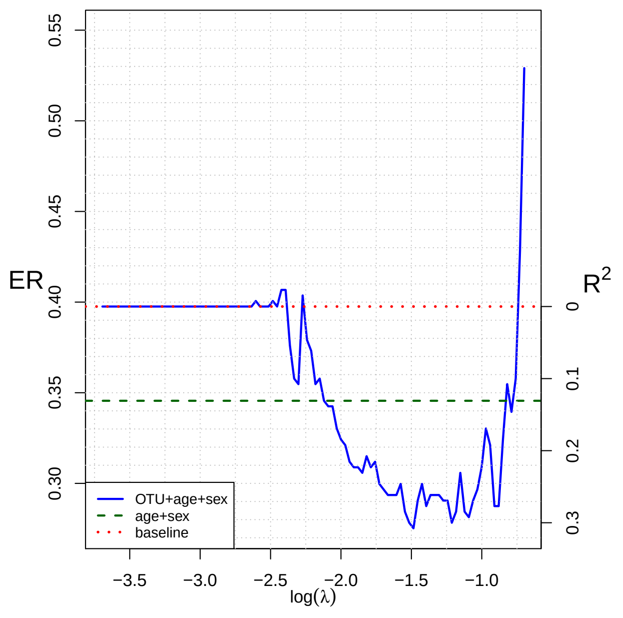
(a) Error Rates of of LR with L_1



(b) Error Rates of of LR with EN

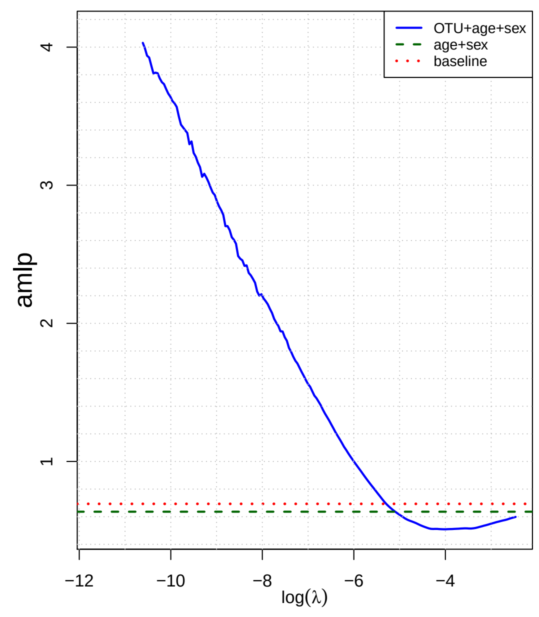


(c) Error Rates of of SVM with L_1

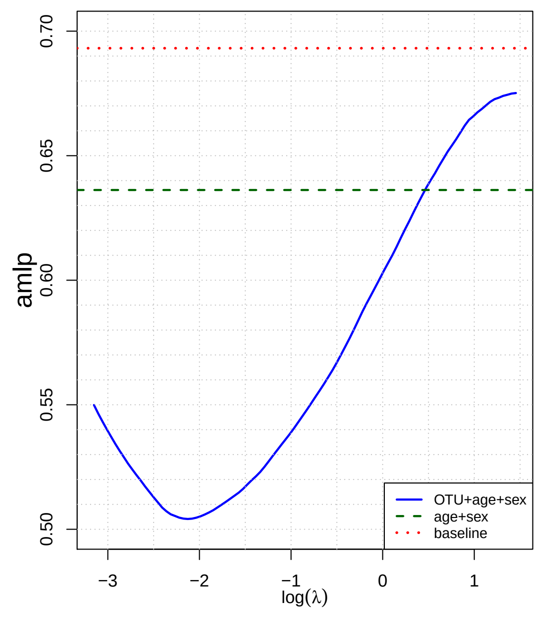


(d) Error Rates of of SVM with EN

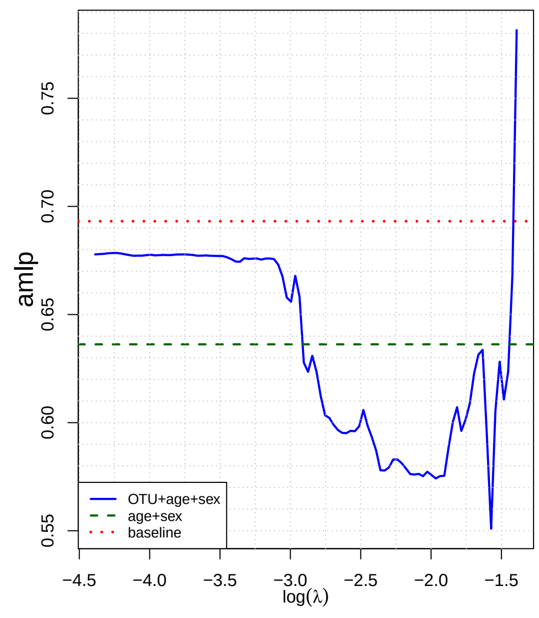
Figure 4.1: Error Rates of Regularized Logistic Regression and Regularized SVM, as A Function of Different λs.



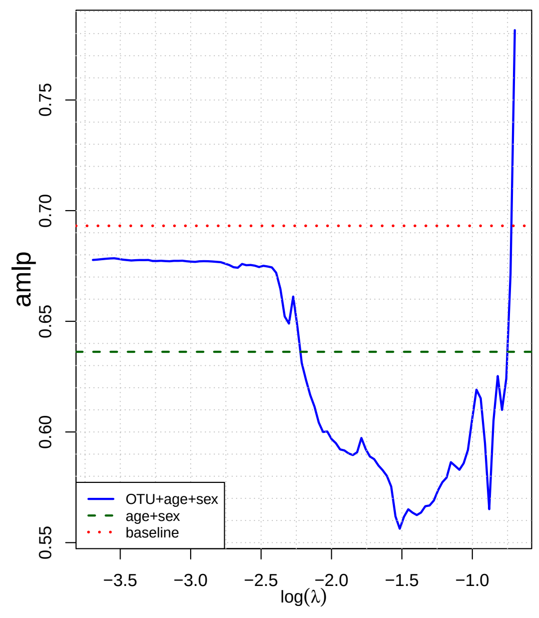
(a) AMLPs of LR with L_1



(b) AMLPs of LR with EN

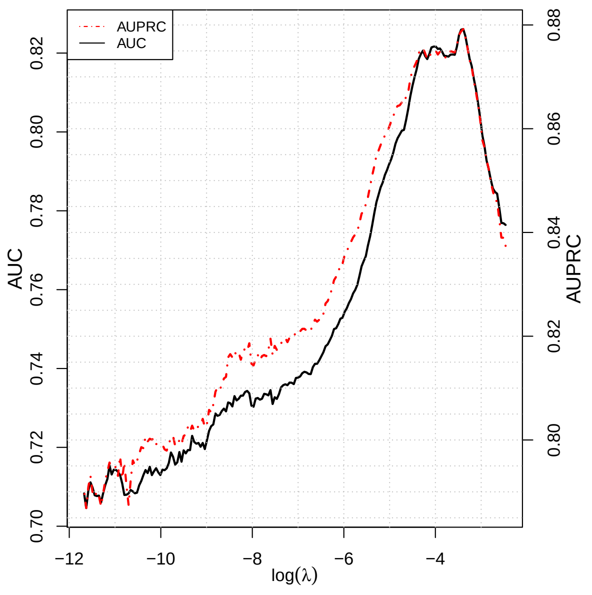


(c) AMLPs of SVM with L_1

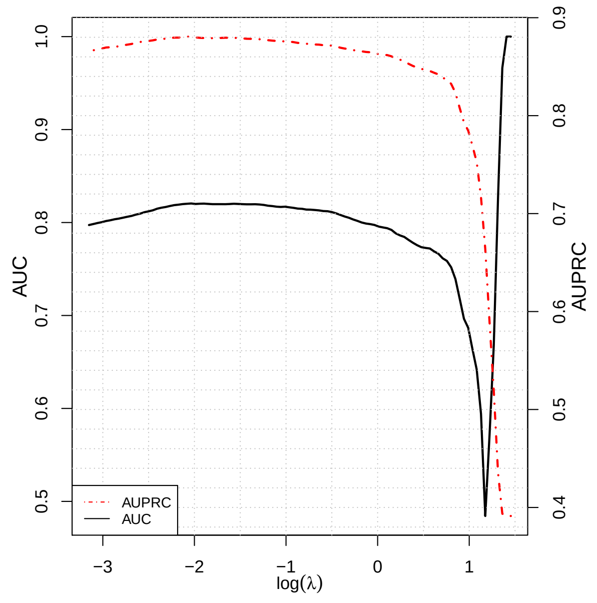


(d) AMLPs of SVM with EN

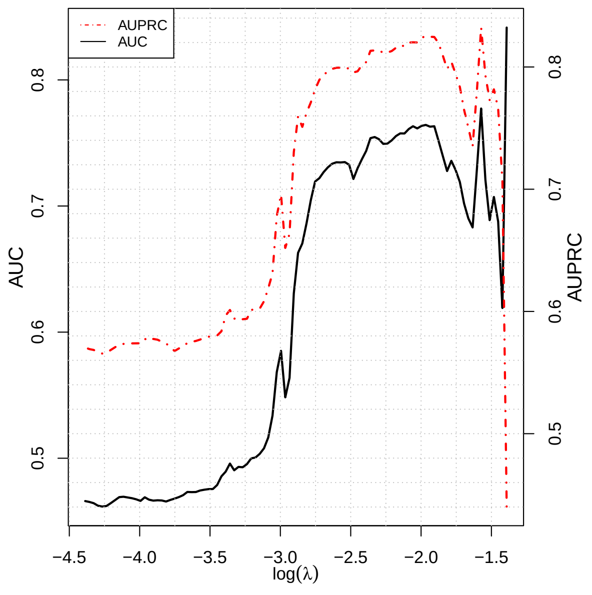
Figure 4.2: AMLPs of Regularized Logistic Regression and Regularized SVM, as A Function of Different λs.



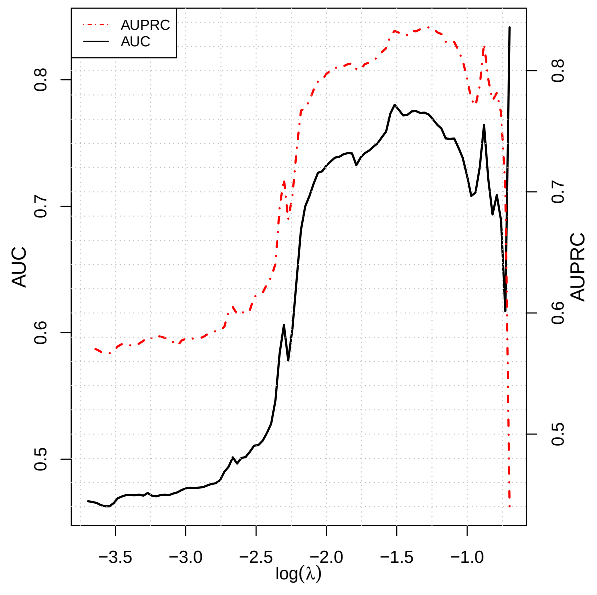
(a) AUC and AUPR of LR with L_1



(b) AUC and AUPR of LR with EN



(c) AUC and AUPR of SVM with L_1



(d) AUC and AUPR of SVM with EN

Figure 4.3: AUC and AUPR of Regularized Logistic Regression and Regularized SVM, as A Function of Different λs.

logistic regression and EN to denote elastic-net penalty, in each sub-figure. Blue lines in Figure 4.1 display test error rates of regularized logistic regression and SVM, with different values of tuning parameter λ. The optimal λ is chosen according to the smallest error rate for each model. The smallest error rate of logistic regression with L_1 penalty, logistic regression with elastic-net penalty, SVM with L_1 penalty and SVM with elastic-net penalty are 0.223, 0.238, 0.278, 0.275, respectively.

Blue lines in Figure 4.2, red and black lines in Figure 4.3 show AMLP, AUC and AUPR of regularized logistic regression and SVM, with different values of tuning parameter λ. Under the optimal λ, the optimal AMLP of logistic regression with L_1 penalty, logistic regression with elastic-net penalty, SVM with L_1 penalty and SVM with elastic-net penalty are 0.515, 0.506, 0.550, 0.562, respectively. The optimal AUC of logistic regression with L_1 penalty, logistic regression with elastic-net penalty, SVM with L_1 penalty and SVM with elastic-net penalty are 0.8259, 0.8202, 0.7773, 0.7762, respectively; the optimal AUPR are 0.8788, 0.8794, 0.8322, 0.8314, respectively.

These figures provide not only the choice of optimal λ, but also the degree of difficulty in predicting the PD. In Figures 4.1 and 4.2, we use red lines to denote the baseline error rate and AMLP. As mentioned in Section 2.3 , R^2 is the percentage of the reduction of error rate or AMLP from the null model based on the frequency of observed y_i without any predictors. We show R^2 on the right vertical axes.

R^2 is a comparison of predictive models versus a random guess without variables, so, R^2 can indicate the predictivity of selected variables for PD, showing how much PD is related to these variables. Under the optimal λ, the R^2 of error rate for logistic regression with L_1 penalty, logistic regresion with elastic-net penalty, SVM with L_1 penalty and SVM with elastic-net penalty are 43.9%, 40.1%, 30.1%, 30.8%, respectively; the R^2 of AMLP are 25.7%, 27.0%, 20.7%, 18.9%, respectively. We can see R^2 of regularized logistics regression models are larger than those of regularized SVM, which means the predictivity of regularized logistics regression models with selected OTUs is stronger than that of regularized SVM. In our study, the effect of R^2 seems to be not obvious enough because we only use one dataset. Particularly, for those studies with different datasets, R^2 is much more useful because different datasets have different baselines of error rate and AMLP. Hence, we cannot directly compare error rates and AMLPs from different dataset, but the comparison of R^2, a relative predictive metric, is reasonable.

Relative comparison with baseline is also shown in Figures 4.2 and 4.3. In these figures, the red line denotes the baseline, and the green line represents those predictive models with age and sex only as predictors, without OTUs. They are parallel to the horizontal line and not affected by the value of λ because their corresponding models don not contain any regularizations. In these figures, we can see the lowest point of those blue lines are much lower than the red line, and red line is higher than green line. Thus, with each optimal λ, OTU features are very predictive, thereby having lower error rate and AMLP compared to the baseline model. Prediction only based on age and sex is more reliable than random guess, which indicates that age and sex are also factors related to PD. This is consistent with the basic factors of PD, discussed in Section 1.1

4.1.2 Performance of Models with Respective Optimal Tuning Parameters

Table 4.1: Evaluation of Predictive Models with Respective Optimal λ

Metric / Model	ER (R^2_{ER})	AMLP (R^2_{AMLP})	AUC	AUPR
Baseline (No Predictor)	0.398 (0%)	0.693 (0%)	0.5000	0.6024
LR only based on age+sex	0.346 (13.1%)	0.636 (8.20%)	0.6633	0.7182
LR	0.391 (1.8%)	Inf	0.6355	0.7319
LR+L_1	**0.223 (43.9%)**	0.515 (25.7%)	**0.8259**	0.8788
LR+EN	0.238 (40.1%)	**0.506 (27.0%)**	0.8202	**0.8794**
SVM	0.355 (10.8%)	0.639 (7.8%)	0.6899	0.7666
SVM+L_1	0.278 (30.1%)	0.550 (20.7%)	0.7773	0.8322
SVM+EN	0.275 (30.8%)	0.562 (18.9%)	0.7762	0.8314

In this section, we further investigate optimal models with their own ideal λ. Table 4.1

summarizes specific values of evaluation metrics for each model with its optimal λ. As a straightforward demonstration for predictive performances of these models, Table 4.1 shows that logistic regression and SVM, without L1 and elastic-net penalties, perform not so good when all OTUs are considered as predictors. The error rate of logistic regression based on all OTU variables is 0.391, almost no improvement comparing with the baseline. What's worse, the AMLP is ∞ because for some samples, the predicted probabilities of true labels are 0s. The error rate and AMLP of SVM based on all OTU variables are a little better, but still worse than the results of the model only based on age and sex. Without regularizations, the dataset is with much noise, and it may be unable to converge during the mathematical optimization process of solving out the loss functions. From the table, we can see that logistic regression with L_1 penalty performs best. Although the AMLP of logistic model with L_1 is a little larger than that of logistic regression with elastic-net, its other metrics are still the best of all. If we only focus on regularizations, we can see that the performances of two penalties are very close, hard to be distinguished; if we pay attention to models, it is clear that logistic regression performs better than SVM.

Nevertheless, there is no clear answer for the question whether logistic regression does better than linear SVM. Actually it depends on what dataset we analyze. In our study, we use linear SVM without kernel tricks, which performs similar with logistic regression, but there are still differences between them. The loss function of SVM is hinge loss whereas logistic regression has a cross entropy loss function. The corresponding convex optimization of SVM involves Lagrange Duality and Karush Kuhn Tucker (KKT) conditions, resulting in only points within the margin (support vectors) being decisive for optimal solution. In SVM, only points within the margin affect the model, so alterations of other points cannot change solutions of the model. Therefore, SVM has a stronger generalization ability and is preferred for noisy high-dimensional datasets.

Unlike SVM, logistic regression considers all training data. Thus, logistic regression is more sensitive to outliers and alterations. However, it is perfect for a dataset that does not have so many dimensions and noises. Sometimes SVM may ignore some significant information by only relying on points near the margin, especially when the dataset is not so large. Our dataset has 327 observations and 382 features. Compared with datasets having thousands of dimensions and observations, our dataset is low dimensional. Hence, it's advisable to make use of all samples and features, to make our models stable and reliable. Under this circumstance, using SVM relying on support vectors may have missed some significant information.

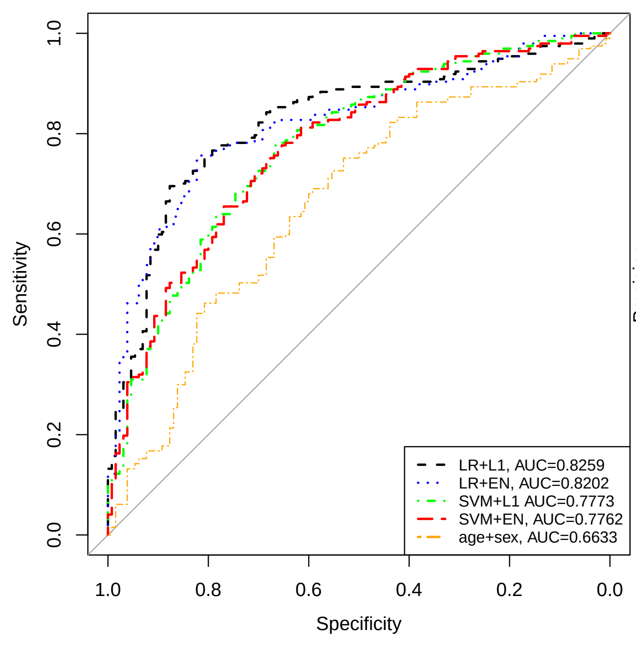
Figure 4.4 displays the ROC and PRC for models. ROC reveals the rate of correctly classified positive observations (true positive rate) versus the rate of incorrectly classified negative observations (false positive rate), under each threshold. The diagonal line represents points where the true positive rate equals the false positive rate. Without using any variables as predictors, such as random guessing, then the true positive rate will always equal the false positive rate. The diagonal line represents this baseline situation.

Intuitively, we expect that true positive rate can be as high as possible whereas false positive rate can be as low as possible. Then, in ROC space, the closer a curve is to the left-top corner, the better the predictive performance is. Curves close to left-top corner and enclose the diagonal line are indicators of favorable predictions, while curves close to diagonal line imply worse predictive performance.

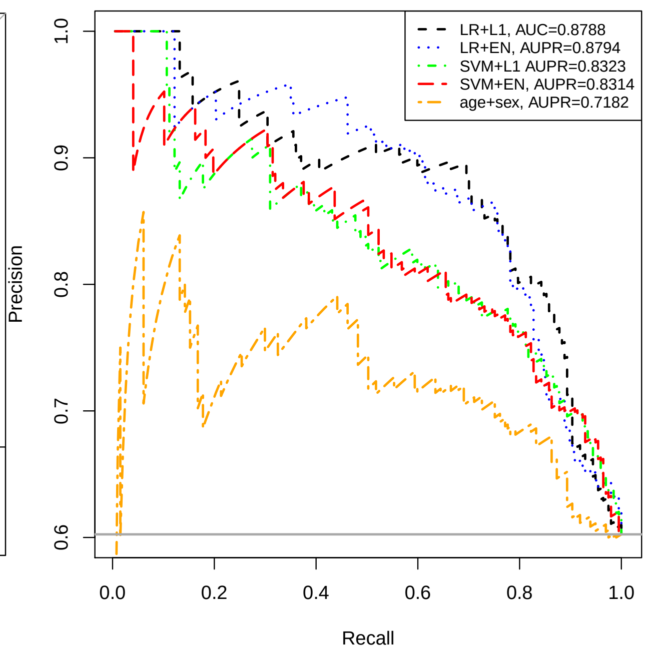
ROC curves are shown in Figure 4.4a. The diagonal line is the baseline model without any predictors, the orange line is ROC curve of the predictive model only with age and sex as predictors, and other lines are ROC curves for regularized models with respective optimal λ. The black line is ROC curve of logistic regression with L_1; the blue line is ROC curve of logistic regression with elastic-net; the green line is ROC curve of SVM with L_1; the red line is ROC curve of SVM with elastic-net. We can see these lines are close to the left-top corner, enclosing diagonal lines, and the orange line is between lines of regularized models and the diagonal line. This relative position suggests that regularized models with OTU, age and sex are more predictive than the model only with age and sex as predictors. However, the model only with age and sex is still better than baseline situation. Two curves of regularized logistic regression are more closer to left-top than those of regularized SVM, which indicates that in our study, the performance of regularized logistic regression is better than that of regularized SVM. However, the curve of logistic regression with L_1 is very close to the curve of logistic regression with elastic-net, and the curve of SVM with L_1 is also very close to the curve of SVM with elastic-net. Then, it is hard to compare their performances explicitly only by looking at curves. Thus, we show AUC values for further comparisons.

Figure 4.4b shows PR curves for models. PR curve depicts precision versus recall under each threshold. The baseline of PR curve depends on the distribution of data. In PRC space, baseline is a horizontal line whose y-coordinate is the frequency of positive observations in a dataset. If we do not use any variables as predictors, and guess randomly, then the precision will be always equal to the ratio of positive samples to all samples. The higher the precision and recall are, the more advantageous a model is. In PR space, the closer a curve is to the right-top corner, the better the predictive performance is, while curves closer to baseline imply relatively worse predictive performances.

In Figure 4.4b, the gray horizontal line at the bottom is the baseline without any predictors. The orange line is PR curve of the predictive model only with age and sex as features. Other lines are PR curves of regularized models, with respective optimal λ. Black, blue, green and red lines are PR curves of logistic regression with L_1, logistic regression with elastic-net, SVM with L_1, and SVM with elastic-net, respectively. The line of the model only based on age, sex and lines of regularized models are all close to the right-top corner, and they are located far above the baseline. The orange line is also above the baseline obviously, but still underneath other lines. It suggests that age, sex are factors of PD, and that models with age, sex and OTU as predictors would be much more predictive. PR curves of regularized logistic regression are much closer to the right-top corner than those of regularized SVM. However, two curves of regularized logistic regression are close to each other, and two curves of regularized SVM are also very close. So, we show values of AUPR for further comparisons.



(a) ROC Curves of LR and SVM



(b) PR Curves of LR and SVM

Figure 4.4: ROC and PR Curves of Regularized Logistic Regression and Regularized SVM, with Optimal λ.

4.2 Selection of Related OTUs

4.2.1 The Process of Selection

In addition to the predictive performance of various models, the selection of significant OTUs is of great importance. As mentioned in Section 2.1.4, L_1 penalty does feature selection by setting coefficients associated with non-significant variables to exactly zero. Elastic-net has a grouping effect on the basis of L_1 penalty. Next, we pay attention to the coefficients of each variable. Intuitively, the larger absolute value of a coefficient is, the more significant the associated variable is. One thing should be noted is that we conduct LOOCV, which makes coefficients of variables differ in each fold. In our dataset, 327 observations mean that we have 327 folds in LOOCV, and then, each variable has 327 coefficients. For a certain variable, the absolute value of coefficient in a fold may be zero, while in another fold, it might be non-zero. Thus, if the median of absolute values of coefficients from all folds is zero, then the associated variable is eliminated; if the median of absolute values of coefficients is non-zero, the associated variable is retained. In each regularized model with optimal λ, different OTUs are retained. In logistic models, with L_1 penalty, the number of retained OTUs is 25, and with elastic-net penalty, the number is 196. SVM with L_1 retained only 2 OTUs, while SVM with elastic-net retained 27 OTUs.

The number of remaining OTUs varies so much. As we discussed in Section 4.1.2, the solution of SVM only relies on a subset of training data, named as support vectors. The sparseness of solution is obtained by solving out optimization problems conditional on KKT. Therefore, the solution of SVM tends to be sparser than that of logistic models. Besides, in Section 2.1.5 we have mentioned that if there is a group of highly related variables, L_1 penalty tends to choose one of them and ignore others, while elastic-net often considers all of them in the group. Thus, under the same model, the solution of L_1 penalty will be sparser than that of elastic-net penalty.

We rank the top 20 remaining OTUs from most significant to less significant, by ranking the median. Boxplots in Figures 4.5a, 4.6a, 4.7a, and 4.8a show distributions of absolute values of coefficients associated with top 20 variables. Boxplots in Figures 4.5b, 4.6b, 4.7b, and 4.8b show the distributions of standardized log relative abundance of selected OTUs. Here, log relative abundances are from Formula 2.18. We can see that for most selected OTUs, the size and length of blue boxes are different from those of red boxes, which means that the distribution of relative abundance differs between controls and PD cases. For some selected OTUs, the shape of blue box is similar as the red box, but they have outliers. These outliers are very likely to have an impact on the prediction of PD.

4.2.2 Consistency with Other Studies

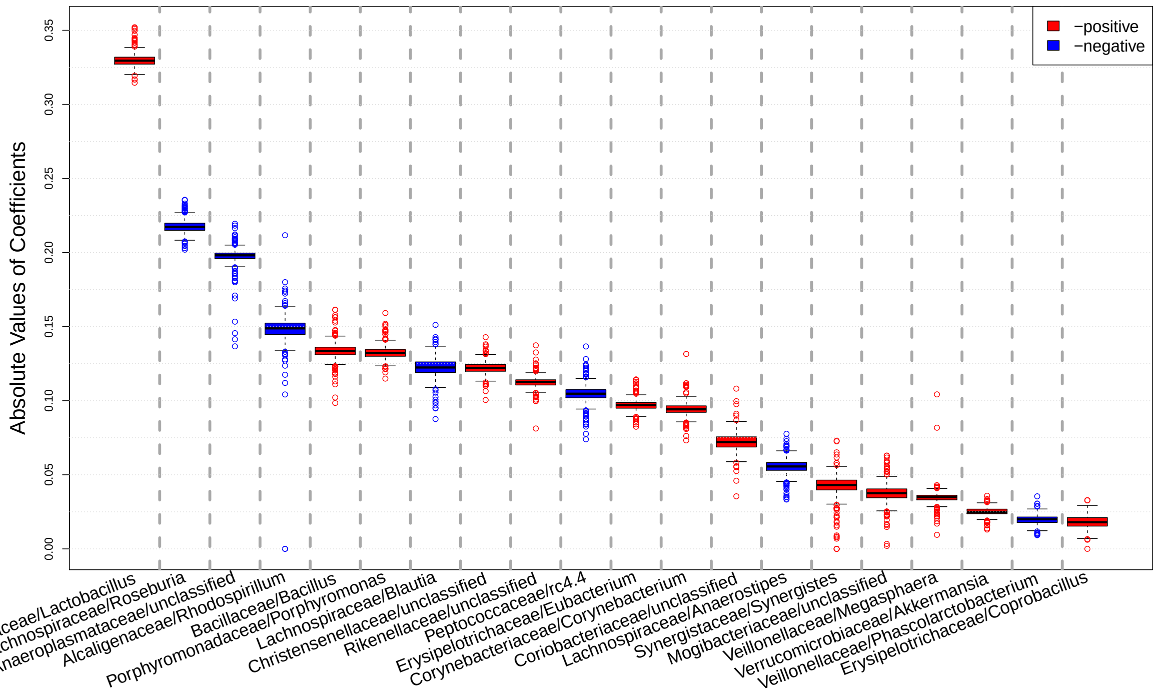
In our study, the sparsest model is SVM with L_1, that retained only 2 OTUs as predictors, Roseburia and Lactobacillus. These two genera are also at the top positions in rankings of selected OTUs by other models. Lactobacillus is ranked first by logistic regression with L_1 penalty, sixth by logistic regression with elastic-net penalty, and first by SVM with elastic-net penalty. Roseburia is ranked second by logistic regression with L_1, 5th by SVM with elastic-net, and 27th by logistic regression with elastic-net. There are some evidences indicating that in PD patients, the level of relative abundances of Roseburia in gut is decreased [41][15], while some studies conclude that Lactobacillus is more abundant in PD patients [42][17].

Ali Keshavarzian, Stefan Green, et al [15] and Vjacheslav Petrov, Irina Saltykova, et al [17] also imply that the levels of relative abundances of Bluatia, Akkermansia and Bifidobacterium are significantly changed in the course of PD. PD patients tend to have a higher level of Akkermansia and Bifidobacterium, but a lower level of Bluatia. In Figures 4.5b, 4.6b and 4.8b, we can see that Blautia is ranked 7th by logistic regression with L_1, 19th by logistic regression with elastic-net, and 16th by SVM with elastic-net. The shapes of boxplots show that Blautia in PD patients is relatively more abundant than in healthy controls. For Bifidobacterium and Akkermansia, we can see that only SVM with elastic-net ranks Bifidobacterium in the top 20, at 10th, and only logistic regression with L_1 ranks Akkermansia in the top 20, at 18th. However, we have already mentioned in Section 4.2.1 that logistic regression with elastic-net has 196 OTUs retained, that logistic regression with L_1 has 25 OTUs retained, and that SVM with elastic-net has 27 OTUs retained. We only display the boxplots of the top 20. There are OTUs that were selected by our models but not displayed in the figures. Bifidobacterium is ranked 24th and 98th, by logistic regression with L_1 and elastic-net, respectively. Akkermansia is ranked 23rd and 127th by SVM with elastic-net and logistic regression with elastic-net, respectively. They are all in the list of retained predictors for each model.

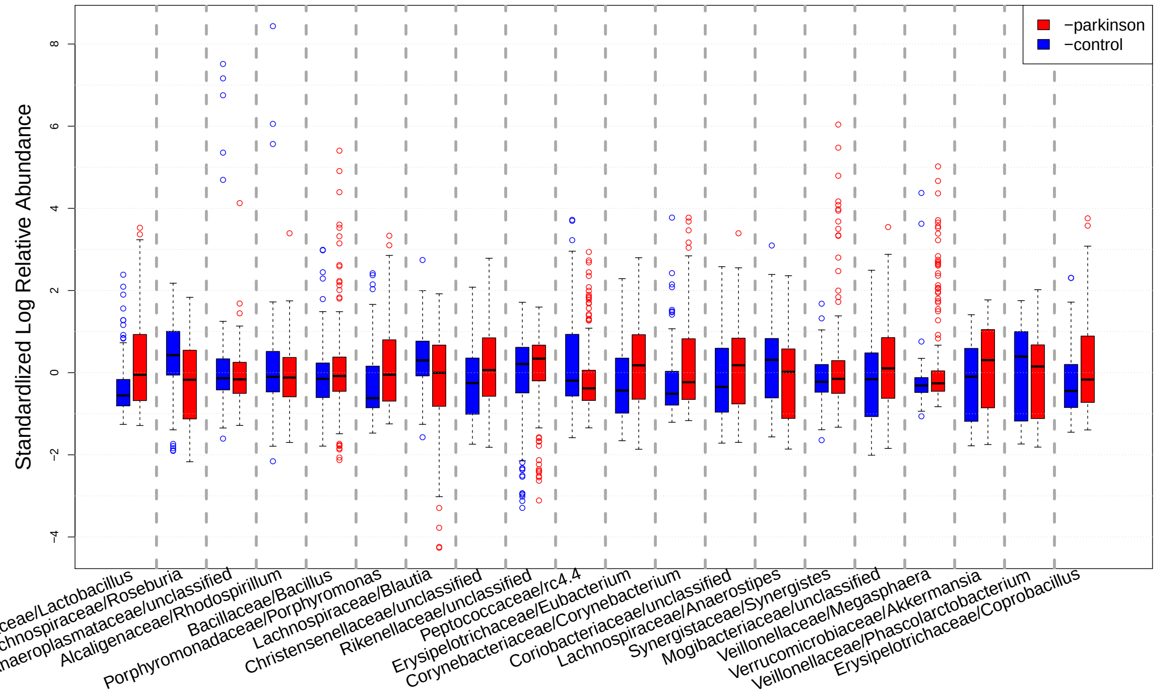
Additionally, a study [43] that used the same dataset as ours, also explored the relation between PD and gut microbiome. Though methodologies are different, our study still has some consistencies with theirs. [43] summarizes 8 genera that are highly related with PD. Lactobacillus, Blautia, Roseburia, Bifidobacterium, Akkermansia are all in the list of these 8 genera. One of these 8 genera is a genus without a specific name, under the family Christensenellaceae. In our results, all the lists include this genus with the designation Christensenellaceae/unclassified. It is ranked 8th and 12th by logistic regression with L_1 and SVM with elastic-net, respectively. We can see it from Figures 4.5b and 4.8b. It is ranked 47th by logistic regression with elastic-net, so we cannot see it in Figure 4.6b, but it is still in the list of retainment from logistic model with elastic-net.

All in all, the results of our study do have some consistencies with conclusions of other published studies.

Figure 4.5: Selected OTUs of Logistic Regression with L_1

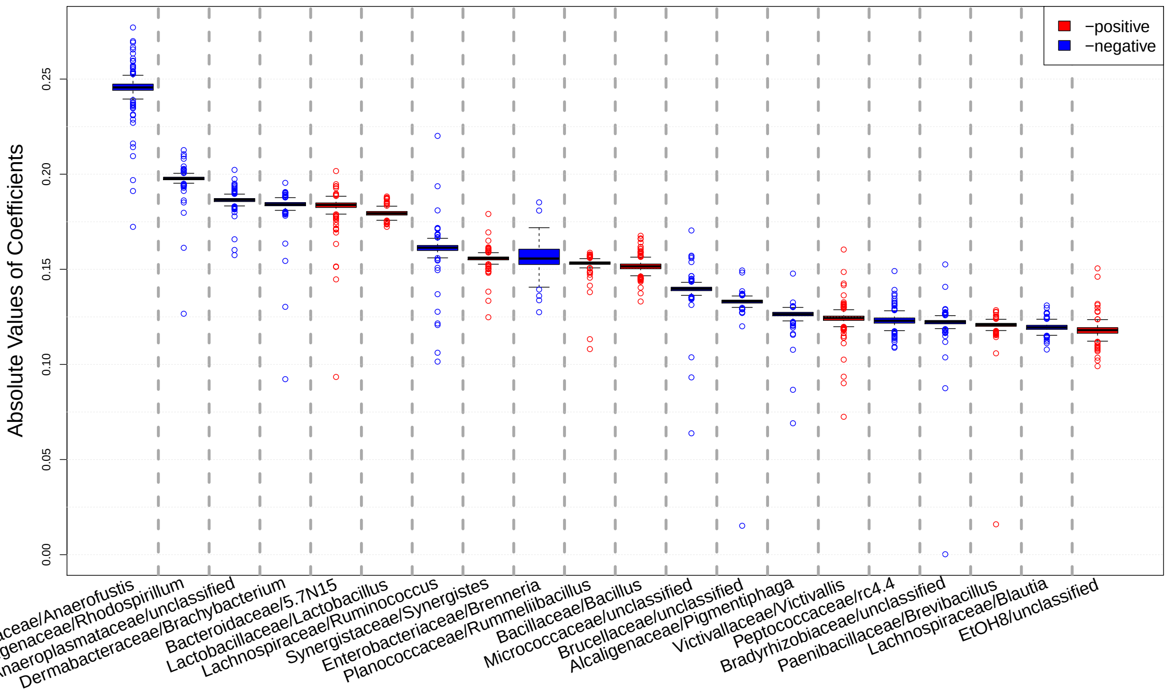


(a) Absolute Coefficients of Selected OTUs by LR with L_1

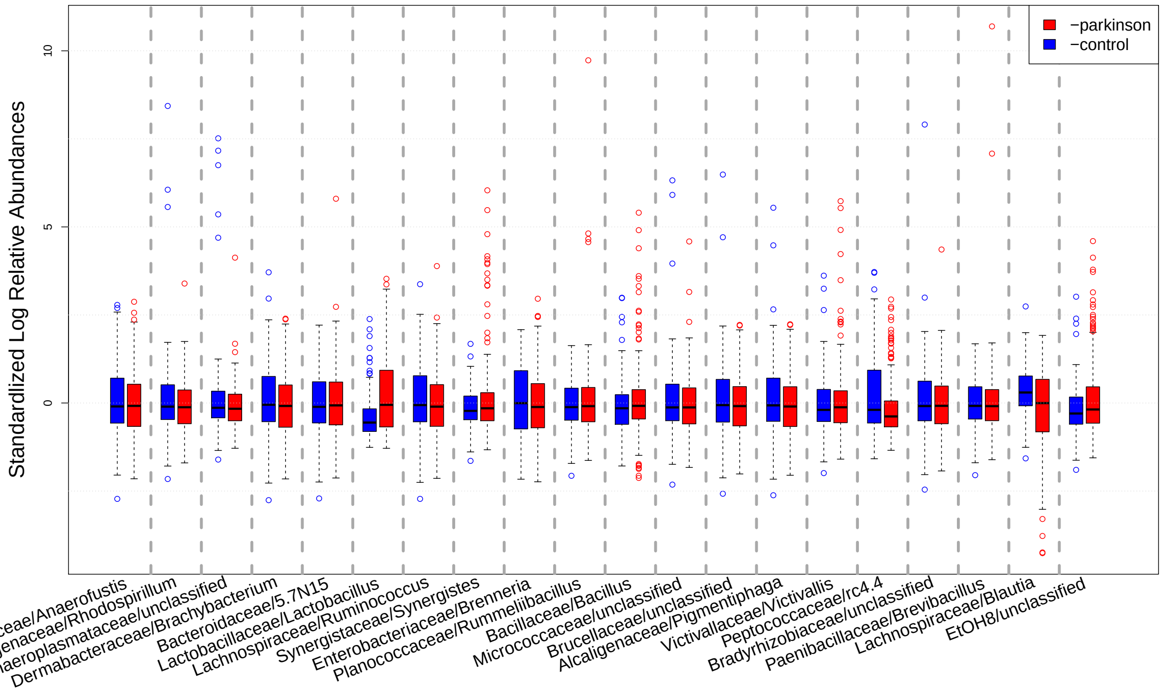


(b) Relative Abundances of Selected OTUs by LR with L_1

Figure 4.6: Selected OTUs of Logistic Regression with Elastic-net

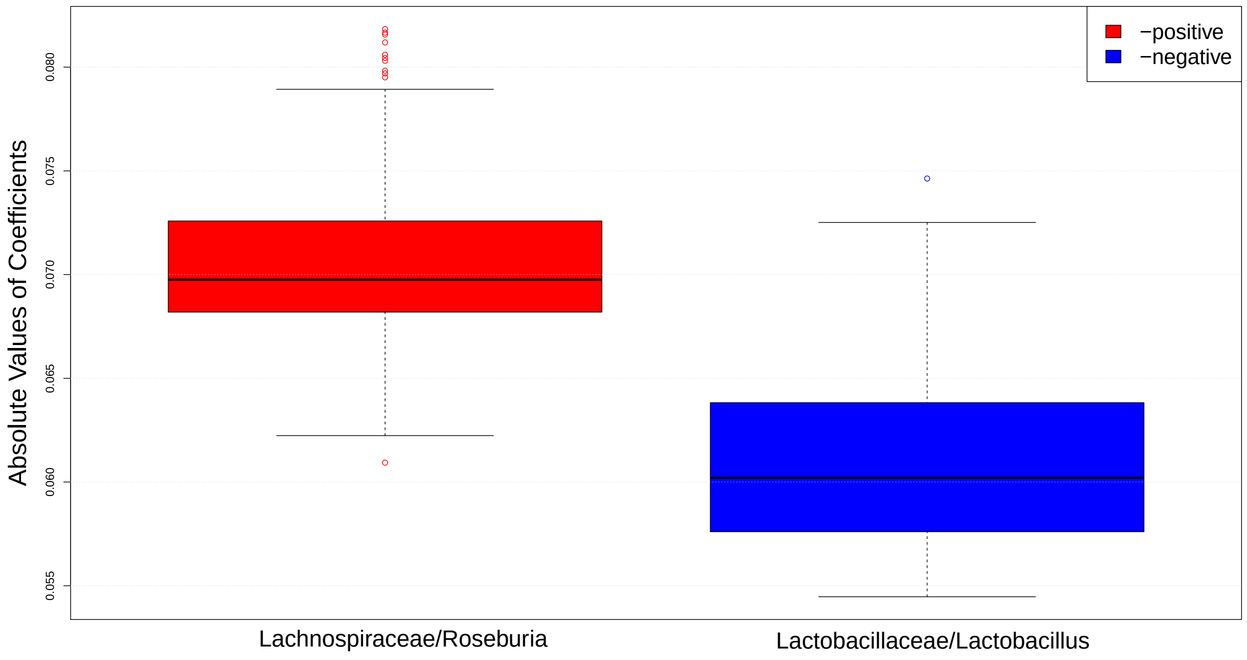


(a) Absolute Coefficients of Selected OTUs by LR with EN

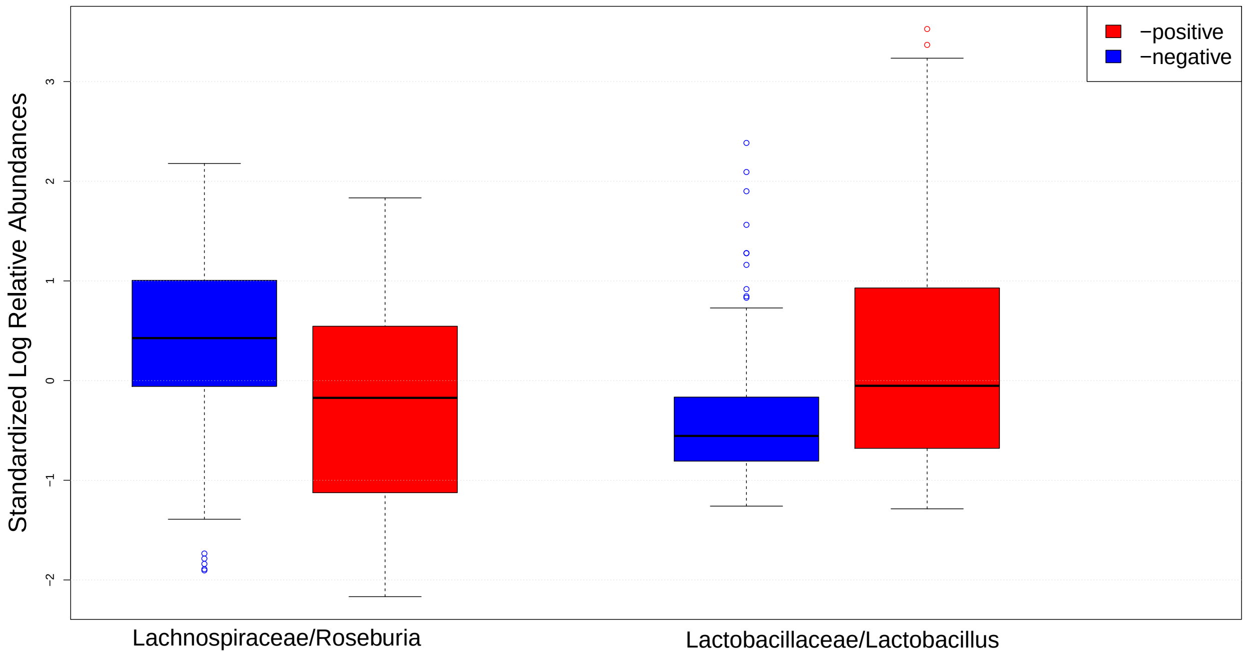


(b) Relative Abundances of Selected OTUs by LR with EN

Figure 4.7: Selected OTUs of SVM with L_1

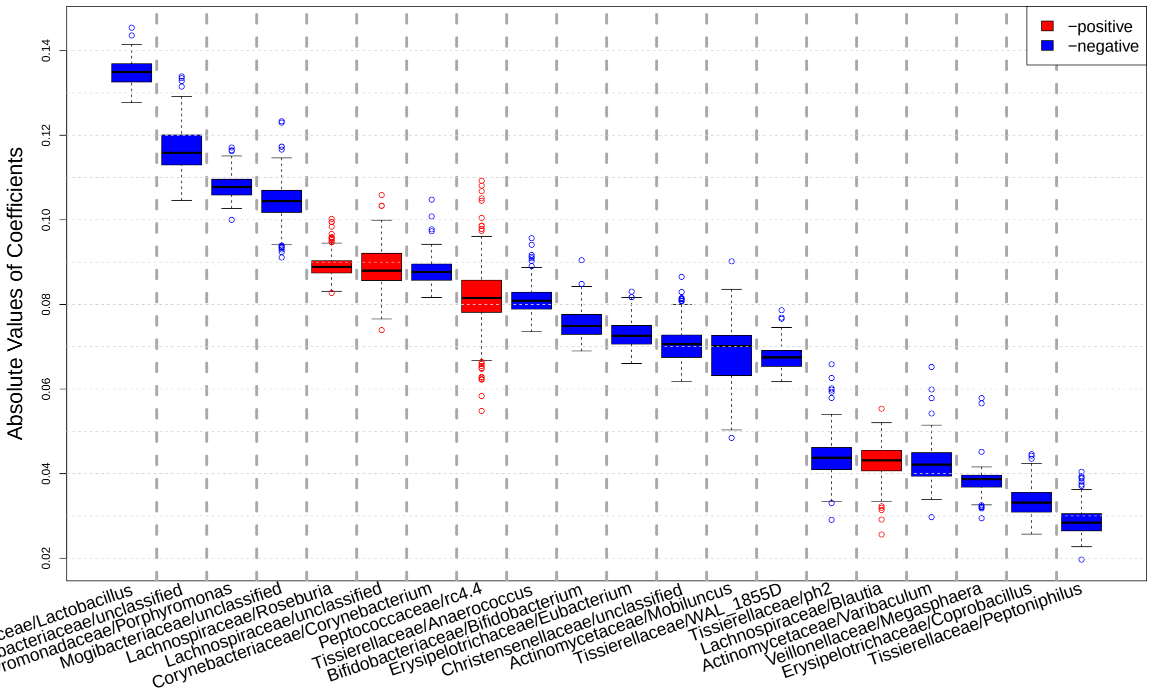


(a) Absolute Coefficients of Selected OTUs by SVM with L_1

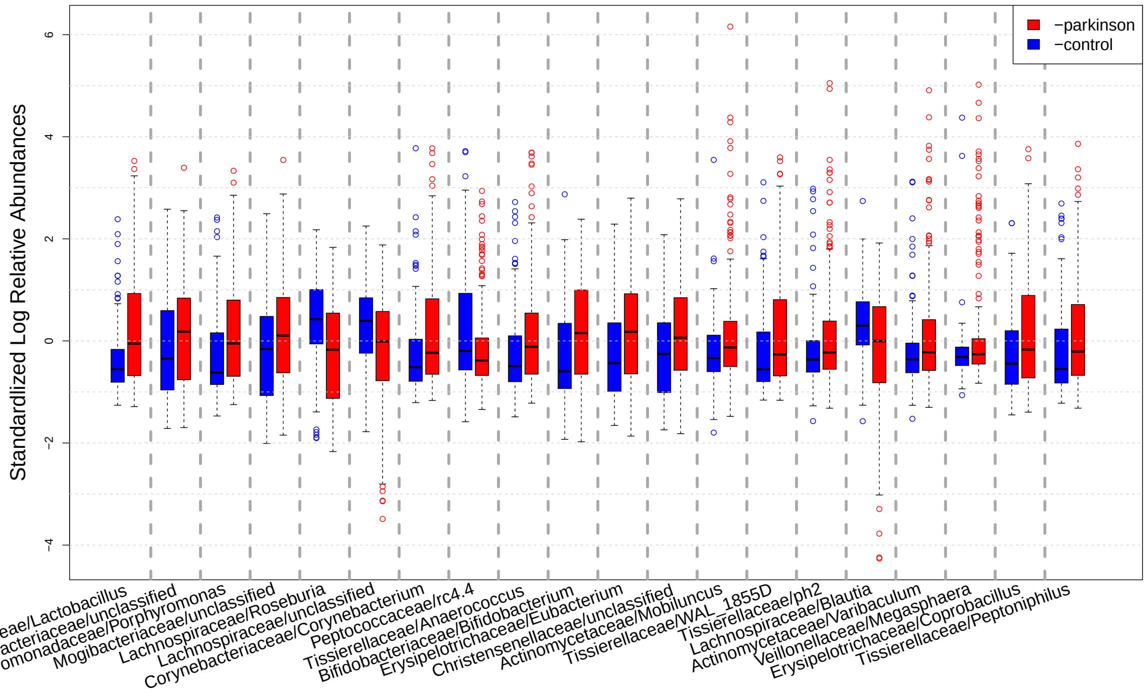


(b) Relative Abundances of Selected OTUs by SVM with L_1

Figure 4.8: Selected OTUs of SVM with Elastic-net



(a) Absolute Coefficients of Selected OTUs by SVM with EN



(b) Relative Abundances of Selected OTUs by SVM with EN

Chapter 5
Conclusion and Future Work

In this thesis, we explore the relationship of PD and gut microbiota by fitting SVM and logistic regression models to predict PD, with OTUs as predictors. To select significant OTUs, regularizations are applied into predictive models, including L_1 regularization and elastic-net regularization. LOOCV is conducted to choose the optimal values of the tuning parameter λ in regularizations. We also discuss CV, k-fold CV and LOOCV.

Then, we select significant OTUs related with PD, according to the median of absolute values of coefficients associated OTU variables. If absolute values of coefficients have a non-zero median, then the corresponding variable is retained.

After selection of significant OTUs, we compare our results with reported studies. Other studies have found that abundances of some gut genera in PD patients differ significantly between patients and controls. Selected genera in our study are in accordance with most of their results, including Lactobacillus, Blautia, Roseburia, Bifidobacterium, Akkermansia and a genus without a specific name, under the family Christensenellaceae.

Although the consistency with other studies exists, we still fail to exactly figure out a small group of specific OTUs that are highly related with PD. Our body is a complex system. Each OTU has complicated interactions with gut microbial environment in the progress of PD. It is hard to figure out all details and interactions. In those published studies, there are some other genera recognized as very significant, but in our results, they are not selected as significant variables. It means our models may have apparent drawbacks. Furthermore, correlation does not imply causation. We confirm that some patterns of gut microbiota are related with PD, but we still cannot know what causes changes of gut mocrobiota in the course of PD. There are many other models suitable for OTU data, such as zero-inflated regression models. We can further explore our dataset using other models in the future.

We can use larger datasets and combine more variables like body mass, height, medications, the models may be more accurate, to make our models more persuasive. Moreover, we can consider time-series analysis by using a dataset including data for different PD stages, which can bring us a better understanding for the changes of relative abundance of the gut OTUs during the process of PD. Besides, some medical experiments can be also done by injecting related genera into animal's gut, and observing the changes.

References

[1] Parkinson's Foundation. Understanding parkinson's-statistics-who has parkinson's. *https://www.parkinson.org/Understanding-Parkinsons/Statistics*.

[2] National Institute of Neurological Disorders and Stroke. Parkinson's disease information page. *https://www.ninds.nih.gov/Disorders/All-Disorders/Parkinsons-Disease-Information-Page#disorders-r1*, 2016.

[3] Sigurlaug Sveinbjornsdottir. The clinical symptoms of parkinson's disease. *Journal of Neurochemistry*, 139(1):318–324, 2016.

[4] Lorraine Kalia and Anthony Lang. Parkinson's disease. *The Lancet*, 386(9996):896–912, 2015.

[5] Healthline. Everything you want to know about parkinson's disease. *https://www.healthline.com/health/parkinsons*.

[6] Maria Cersosimo, Gabriela Raina, et al. Gastrointestinal manifestations in parkinson's disease: prevalence and occurrence before motor symptoms. *Journal of Neurology*, 260(5):1332–13328, 2013.

[7] David Devos, Thibaud Lebouvier, et al. Colonic inflammation in parkinson's disease. *Neurobiology of Disease*, 50:42–48, 2013.

[8] Gwen Falony, Marie Joossens, et al. Population-level analysis of gut microbiome variation. *Science*, 352(6285):560–564, 2016.

[9] Alexandra Zhernakova and Alexander Kurilshikov. Population-based metagenomics analysis reveals markers for gut microbiome composition and diversity. *Science*, 352(6285):565–569, 2016.

[10] Ömrüm Aydin, Max Nieuwdorp, and Victor Gerdes. The gut microbiome as a target for the treatment of type 2 diabetes. *Current Diabetes Reports*, 18(8), 2018.

[11] Patrice Cani. Human gut microbiome: hopes, threats and promises. *Gut*, 67(9):1716—-1725, 2018.

[12] Andrew Shreiner, John Kao, and Vincent Young. The gut microbiome in health and in disease. *Current Opinion in Gastroenterology*, 31(1):69—-75, 2015.

[13] Maria Carmen Cénita, Vasiliki Matzaraki, et al. Rapidly expanding knowledge on the role of the gut microbiome in health and disease. *Neurobiology of Disease*, 1842(10):1981–1992, 2014.

[14] Filip Scheperjans, Velma Aho, et al. Gut microbiota are rlated to parkinson's disease and clinical phenotype. *Movement Disorders*, 30(3):350–358, 2015.

[15] Ali Keshavarzian, Stefan Green, et al. Colonic bacterial composition in parkinson's disease. *Movement Disorders*, 30(10):1351–1360, 2015.

[16] Marcus Unger, Jorg Spiegel, et al. Short chain fatty acids and gut microbiota differ between patients with parkinson's disease and age-matched controls. *Parkinsonism and Related Disorders*, 32:66–72, 2016.

[17] Vjacheslav Petrov, Irina Saltykova, et al. Analysis of gut microbiota in patients with parkinson's disease. *Bulletin of Experimental Biology and Medicine volume*, 162:734–737, 2017.

[18] Herbert Ross et al. Principles of numerical taxonomy. *Systematic Biology*, 13:106–108, 1964.

[19] Mary Ann Tolson Carl Boyd and Wayne Copes. Evaluating trauma care: The triss method. trauma score and the injury severity score. *The Journal of Trauma*, 27(4):370–378, 1987.

[20] Murat Kologlu, Doruk Elker, et al. Validation of mpi and pia ii in two different groups of patients with secondary peritonitis. *Hepato-Gastroenterology*, 48(37):147–151, 2001.

[21] Sebastiano Biondo, Emilio Ramos, et al. Prognostic factors for mortality in left colonic peritonitis: a new scoring system. *Journal of the American College of Surgeons*, 191(6):635–642, 2000.

[22] John Marshall, Deborah Cook, et al. Multiple organ dysfunction score: A reliable descriptor of a complex clinical outcome. *Critical Care Medicine*, 23(10):1638–1652, 1995.

[23] Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (saps ii) based on a european/north american multicenter study. *The Journal of the American Medical Association*, 270(24):2957–2963, 1993.

[24] Andrew Zisserman. Lecture 2 ”the svm classifier”, c19 machine learning. *http://www.robots.ox.ac.uk/ az/lectures/ml/lect2.pdf*, 2015.

[25] Christopher M. Bishop. Chapter 7. sparse kernel machines. *Pattern Recognition and Machine Learning*, 2006.

[26] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal, Series B (Methodological)*, 58(1):267—-288, 1996.

[27] Jerome Friedman, Rob Tibshirani, and Trevor Hastie. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1—-22, 2010.

[28] Li Wang, Ji Zhu, and Hui Zou. The doubly regularized support vector machine. *Statistica Sinica*, 16:589–610, 2006.

[29] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1–22, 2010.

[30] Gareth James, Daniela Witten, Trevor Hastie, et al. Chapter 5. resampling methods. *An Introduction to Statistical Learning with Applications in R*, 2017.

[31] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[32] Carl Woese and George Fox. Phylogenetic structure of the prokaryotic domain: The primary kingdoms. *roceedings of the National Academy of Sciences of the United States of America*, 74(11):5088–5090, 1977.

[33] Pablo Yarza, Pelin Yilmaz, et al. Uniting the classification of cultured and uncultured bacteria and archaea using 16s rrna gene sequences. *Nature Reviews. Microbiology*, 12(9):635–645, 2014.

[34] Jean Baldus Patel. 16s rrna gene sequencing for bacterial pathogen identification in the clinical laboratory. *Molecular Diagnosis*, 6(4):313–321, 2001.

[35] Michael Janda and Sharon Abbott. 16s rrna gene sequencing for bacterial identification in the diagnostic laboratory: pluses, perils, and pitfalls. *Journal of Clinical Microbiology*, 45(9):2761–2764, 2007.

[36] Justin Kuczynski, Jesse Stombaugh, et al. Using qiime to analyze 16s rrna gene sequences from microbial communities. *Current Protocols in Bioinformatics*, page Doi:10.1002/0471250953.bi1007s36, 2011.

[37] Evguenia Kopylova, Laurent Noe, and Helene Touzet. Sortmerna: fast and accurate filtering of ribosomal rnas in metatranscriptomic data. *Bioinformatics*, 28(24):3211–3217, 2012.

[38] Daniel McDonald, Morgan Price, et al. An improved greengenes taxonomy with explicit ranks for ecological and evolutionary analyses of bacteria and archaea. *Multidisciplinary Journal of Microbial Ecology*, 6(3):610–618, 2012.

[39] Steffen Heinz, Justin Zobel, and Hugh Williams. Burst tries: A fast,efficient data structure for string keys. *ACM Transactions on Information Systems*, 20:192–223, 2002.

[40] Congrui Yi and Jian Huang. Semismooth newton coordinate descent algorithm for elastic-net penalized huber loss regression and quantile regression. *Journal of Computational and Graphical Statistics*, 26(3):547–557, 2017.

[41] Janis Bedarf, Falk Hildebrand, et al. Functional implications of microbial and viral gut metagenome changes in early stage l-dopa-naive parkinson's disease patients. *Genome Medicine*, 9(1), 2017.

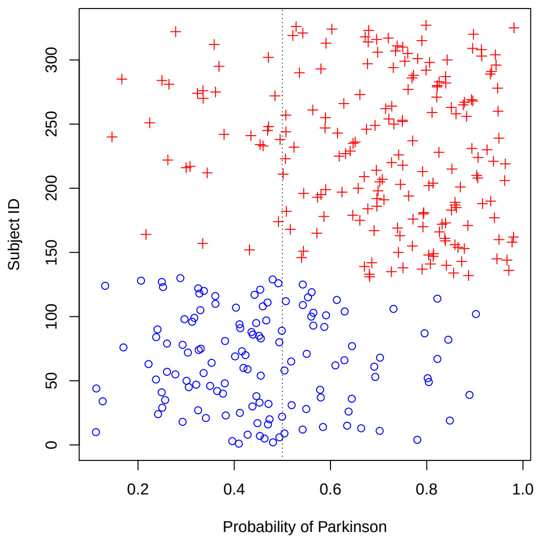
[42] Satoru Hasegawa, Sae Goto, et al. Intestinal dysbiosis and lowered serum lipopolysaccharide-binding protein in parkinson's disease. *PLoS One*, 10(11):e0142164, 2015.

[43] Erin Hill-Burns, Justine Debelius, et al. Parkinson's disease and parkinson's disease medications have distinct signatures of the gut microbiome. *Movement Disorders*, 32(5):739–749, 2017.

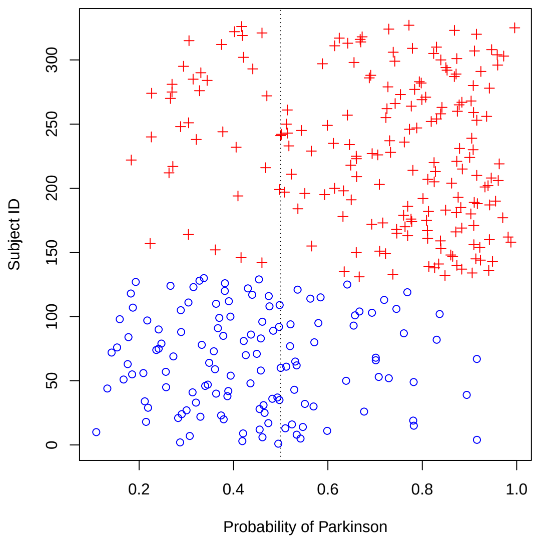
Appendix

Plots of Probability of Getting PD

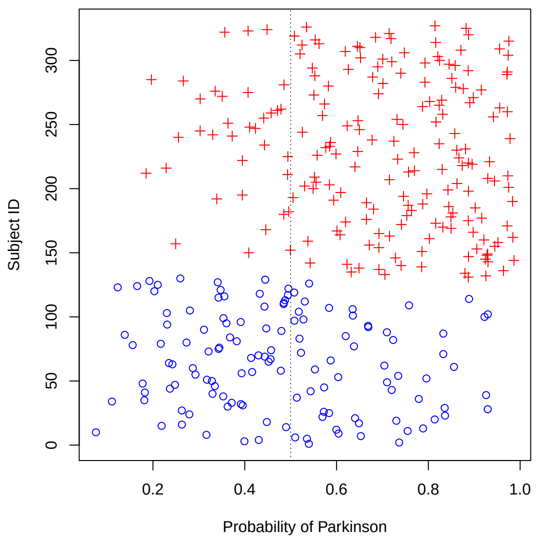
Figure A.1 can be treated as a visual illustration of the confusion matrix for these models with optimal λ. Red crosses and blue circles are positive observations and negative observations, respectively. X-coordinate of a point is the predicted probability of the corresponding observation being a PD patient. Y-coordinates just mean the index of points. Here, the threshold is 0.5. If an observation's predicted probability is no less than 0.5, then the predicted label is positive. Otherwise, the predicted label is negative. The middle vertical line at x=0.5 is the probabilistic decision line. From these figures, we can conclude that logistic models have higher sensitivity and specificity than SVM models, because logistic models have more red crosses located in right of threshold line, and more blue circles in left. Red crosses in left side are positive observations that are incorrectly classified as negative, and blue circles in right are negative samples that are falsely classified as positive. Logistic regression models have a less amount of red crosses in left and blue circles in right. Thus, error rates of logistic models are smaller than those of SVM models.



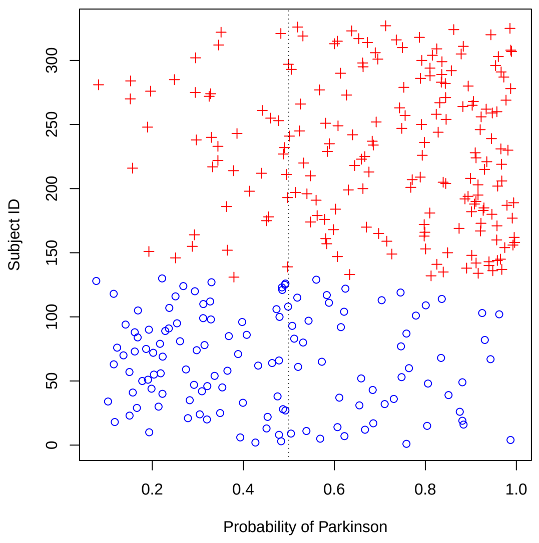
(a) Predicted Probabilities of LR with L_1



(b) Predicted Probabilities of LR with EN



(c) Predicted Probabilities of SVM with L_1



(d) Predicted Probabilities of SVM with EN

Figure A.1: Probability of Parkinson of Regularized Logistic Regression and Regularized SVM, with Optimal λ.

Appendix

Lists of OTUs Retained

B.1 List of Top 25 OTUs Selected by LR+L_1

```
1 Lactobacillaceae/Lactobacillus
2 Lachnospiraceae/Roseburia
3 Anaeroplasmataceae/unclassified
4 Alcaligenaceae/Rhodospirillum
5 Bacillaceae/Bacillus
6 Porphyromonadaceae/Porphyromonas
7 Lachnospiraceae/Blautia
8 Christensenellaceae/unclassified
9 Rikenellaceae/unclassified
10 Peptococcaceae/rc4.4
11 Erysipelotrichaceae/Eubacterium
12 Corynebacteriaceae/Corynebacterium
13 Coriobacteriaceae/unclassified
14 Lachnospiraceae/Anaerostipes
15 Synergistaceae/Synergistes
16 Mogibacteriaceae/unclassified
17 Veillonellaceae/Megasphaera
18 Verrucomicrobiaceae/Akkermansia
19 Veillonellaceae/Phascolarctobacterium
20 Erysipelotrichaceae/Coprobacillus
21 Lachnospiraceae/unclassified
22 Bacteroidales/unclassified/unclassified
23 Staphylococcaceae/Staphylococcus
24 Bifidobacteriaceae/Bifidobacterium
25 Enterobacteriaceae/Brenneria
```

B.2 List of Top 196 OTUs Selected by LR+EN

```
1 Eubacteriaceae/Anaerofustis
2 Alcaligenaceae/Rhodospirillum
3 Anaeroplasmataceae/unclassified
4 Dermabacteraceae/Brachybacterium
5 Bacteroidaceae/5.7N15
6 Lactobacillaceae/Lactobacillus
7 Lachnospiraceae/Ruminococcus
8 Synergistaceae/Synergistes
9 Enterobacteriaceae/Brenneria
10 Planococcaceae/Rummeliibacillus
11 Bacillaceae/Bacillus
12 Micrococcaceae/unclassified
13 Brucellaceae/unclassified
14 Alcaligenaceae/Pigmentiphaga
```

```
15 Victivallaceae/Victivallis
16 Peptococcaceae/rc4.4
17 Bradyrhizobiaceae/unclassified
18 Paenibacillaceae/Brevibacillus
19 Lachnospiraceae/Blautia
20 EtOH8/unclassified
21 Erysipelotrichaceae/Bulleidia
22 Bacillaceae/Oceanobacillus
23 Lachnospiraceae/unclassified
24 Staphylococcaceae/Staphylococcus
25 Bifidobacteriaceae/unclassified
26 Nocardiopsaceae/Prauseria
27 Lachnospiraceae/Roseburia
28 Bifidobacteriaceae/Scardovia
29 Succinivibrionaceae/unclassified
30 Rhizobiaceae/Agrobacterium
31 Synergistaceae/unclassified
32 Microbacteriaceae/Leucobacter
33 Victivallaceae/unclassified
34 Coriobacteriaceae/Eggerthella
35 Rhodocyclaceae/Thauera
36 Flavobacteriaceae/unclassified
37 Veillonellaceae/Megasphaera
38 Paraprevotellaceae/YRC22
39 Eubacteriaceae/Pseudoramibacter_Eubacterium
40 Coriobacteriaceae/unclassified
41 Pseudonocardiaceae/Pseudonocardia
42 Micrococcaceae/Rothia
43 unclassified/unclassified
44 Planococcaceae/Lysinibacillus
45 Anaeroplasmataceae/Anaeroplasma
46 Erysipelotrichaceae/Eubacterium
47 Christensenellaceae/unclassified
48 Coriobacteriaceae/Slackia
49 Porphyromonadaceae/Dysgonomonas
50 unclassified/unclassified
51 unclassified/unclassified
52 Dethiosulfovibrionaceae/Jonquetella
53 Porphyromonadaceae/Porphyromonas
54 Coriobacteriaceae/Adlercreutzia
55 Corynebacteriaceae/Corynebacterium
56 Lactobacillaceae/Pediococcus
57 Rikenellaceae/unclassified
58 Lachnospiraceae/Anaerostipes
59 Comamonadaceae/Comamonas
60 Lachnospiraceae/Epulopiscium
61 Comamonadaceae/Variovorax
62 Leptotrichiaceae/Sneathia
63 Desulfovibrionaceae/unclassified
64 Actinomycetaceae/Mobiluncus
65 Mogibacteriaceae/unclassified
66 Synergistaceae/Cloacibacillus
67 Nocardiaceae/Rhodococcus
68 Actinomycetaceae/unclassified
```

```
69 Sphingobacteriaceae/Sphingobacterium
70 Bacillaceae/unclassified
71 Turicibacteraceae/Turicibacter
72 Paraprevotellaceae/Prevotella
73 Brevibacteriaceae/Brevibacterium
74 unclassified/unclassified
75 Comamonadaceae/Alicycliphilus
76 Tissierellaceae/Anaerococcus
77 Methanomassiliicoccaceae/Methanomassiliicoccus
78 Christensenellaceae/Christensenella
79 unclassified/unclassified
80 Sphingomonadaceae/Sphingobium
81 Alicyclobacillaceae/Alicyclobacillus
82 Alcaligenaceae/Oligella
83 Clostridiaceae/SMB53
84 Nocardioidaceae/unclassified
85 Carnobacteriaceae/Carnobacterium
86 Enterobacteriaceae/Erwinia
87 Leuconostocaceae/Leuconostoc
88 Actinomycetaceae/Arcanobacterium
89 Odoribacteraceae/Odoribacter
90 Erysipelotrichaceae/Coprobacillus
91 Lachnospiraceae/Oribacterium
92 Veillonellaceae/Phascolarctobacterium
93 Aerococcaceae/Aerococcus
94 Comamonadaceae/Acidovorax
95 Paraprevotellaceae/unclassified
96 unclassified/unclassified
97 Rhodobacteraceae/Paracoccus
98 Bifidobacteriaceae/Bifidobacterium
99 Desulfovibrionaceae/Bilophila
100 Weeksellaceae/Elizabethkingia
101 Alteromonadaceae/Cellvibrio
102 Coriobacteriaceae/Collinsella
103 Erysipelotrichaceae/unclassified
104 Lachnospiraceae/Lachnospira
105 Moraxellaceae/unclassified
106 Enterobacteriaceae/Proteus
107 Lachnospiraceae/Coprococcus
108 Ruminococcaceae/unclassified
109 Tissierellaceae/ph2
110 Porphyromonadaceae/Parabacteroides
111 Bacteroidaceae/Bacteroides
112 Elusimicrobiaceae/unclassified
113 Tissierellaceae/WAL_1855D
114 Lachnospiraceae/Clostridium
115 Streptococcaceae/Lactococcus
116 Pasteurellaceae/Haemophilus
117 Flavobacteriaceae/Flavobacterium
118 Tissierellaceae/Helcococcus
119 Brucellaceae/Ochrobactrum
120 unclassified/unclassified
121 Ruminococcaceae/Butyricicoccus
122 Rikenellaceae/Rikenella
```

```
123 Rikenellaceae/Alistipes
124 Enterobacteriaceae/Serratia
125 Tissierellaceae/Finegoldia
126 Bifidobacteriaceae/Alloscardovia
127 Verrucomicrobiaceae/Akkermansia
128 Erysipelotrichaceae/Allobaculum
129 Ruminococcaceae/Faecalibacterium
130 Weeksellaceae/Wautersiella
131 Actinomycetaceae/Actinomyces
132 Enterobacteriaceae/Sodalis
133 Enterobacteriaceae/unclassified
134 Lactobacillaceae/unclassified
135 Tissierellaceae/Gallicola
136 Oxalobacteraceae/unclassified
137 Methanobacteriaceae/Methanobrevibacter
138 Veillonellaceae/Veillonella
139 Fusobacteriaceae/Fusobacterium
140 Peptococcaceae/unclassified
141 Brucellaceae/Pseudochrobactrum
142 Leuconostocaceae/unclassified
143 Cerasicoccaceae/unclassified
144 Caulobacteraceae/unclassified
145 Veillonellaceae/Succiniclasticum
146 Ruminococcaceae/Ruminococcus
147 unclassified/unclassified
148 Odoribacteraceae/Butyricimonas
149 Enterobacteriaceae/Pantoea
150 Tissierellaceae/Parvimonas
151 unclassified/unclassified
152 Micrococcaceae/Micrococcus
153 Comamonadaceae/Limnohabitans
154 Enterococcaceae/Enterococcus
155 Geodermatophilaceae/Blastococcus
156 Veillonellaceae/Anaerovibrio
157 Barnesiellaceae/Barnesiella
158 Rhodocyclaceae/unclassified
159 Microbacteriaceae/Pseudoclavibacter
160 Prevotellaceae/Prevotella
161 Planococcaceae/unclassified
162 unclassified/unclassified
163 Tissierellaceae/1.68
164 Pseudomonadaceae/Pseudomonas
165 Bifidobacteriaceae/Gardnerella
166 S247/unclassified
167 Thermicanaceae/Thermicanus
168 Pasteurellaceae/Gallibacterium
169 Sphingomonadaceae/unclassified
170 Moraxellaceae/Enhydrobacter
171 Comamonadaceae/Rubrivivax
172 Moraxellaceae/Acinetobacter
173 Enterobacteriaceae/Raoultella
174 Staphylococcaceae/Jeotgalicoccus
175 Streptococcaceae/Streptococcus
176 Ruminococcaceae/Anaerotruncus
```

177 Enterobacteriaceae/Rahnella
178 Comamonadaceae/Pelomonas
179 Erysipelotrichaceae/cc_115
180 unclassified/unclassified
181 Paenibacillaceae/Paenibacillus
182 Rickettsiaceae/Rickettsia
183 Aeromonadaceae/unclassified
184 Dehalobacteriaceae/Dehalobacterium
185 Enterobacteriaceae/Citrobacter
186 Comamonadaceae/Rhodoferax
187 Tsukamurellaceae/Tsukamurella
188 SUP05/unclassified
189 Carnobacteriaceae/Granulicatella
190 Ruminococcaceae/Clostridium
191 Clostridiaceae/Clostridium
192 Methanomassiliicoccaceae/vadinCA11
193 Sphingomonadaceae/Sphingomonas
194 Enterobacteriaceae/Trabulsiella
195 Lachnospiraceae/Lachnobacterium
196 Lachnospiraceae/Ruminococcus

B.3 List of Top 2 OTUs Selected by SVM+L_1

1 Lachnospiraceae/Roseburia
2 Lactobacillaceae/Lactobacillus

B.4 List of Top 27 OTUs Selected by SVM+EN

1 Lactobacillaceae/Lactobacillus
2 Coriobacteriaceae/unclassified
3 Porphyromonadaceae/Porphyromonas
4 Mogibacteriaceae/unclassified
5 Lachnospiraceae/Roseburia
6 Lachnospiraceae/unclassified
7 Corynebacteriaceae/Corynebacterium
8 Peptococcaceae/rc4.4
9 Tissierellaceae/Anaerococcus
10 Bifidobacteriaceae/Bifidobacterium
11 Erysipelotrichaceae/Eubacterium
12 Christensenellaceae/unclassified
13 Actinomycetaceae/Mobiluncus
14 Tissierellaceae/WAL_1855D
15 Tissierellaceae/ph2
16 Lachnospiraceae/Blautia
17 Actinomycetaceae/Varibaculum
18 Veillonellaceae/Megasphaera
19 Erysipelotrichaceae/Coprobacillus
20 Tissierellaceae/Peptoniphilus
21 Ruminococcaceae/unclassified
22 Desulfovibrionaceae/Bilophila
23 Verrucomicrobiaceae/Akkermansia

```
24 Tissierellaceae/1.68
25 Rikenellaceae/unclassified
26 Ruminococcaceae/Anaerotruncus
27 Methanobacteriaceae/Methanobrevibacter
```

Appendix

R Code

C.1 Fit Logistic Regression of Parkinson and Age, Sex

```
library(nnet)
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

if (!exists("irep")) irep <- 1
i <- irep #qusb folds=327 times

traindata<-dataset[-folds.index[[i]],]
testdata<-dataset[folds.index[[i]],]
logistic<-glm(parkinson~age+sex,data<-traindata,family = binomial(logit))
testyes<-predict(logistic,testdata,type='response')
trainyes<-predict(logistic,traindata,type='response')

Pr_xgiven_En_log_test<-unname(cbind(log(1-testyes),log(testyes)))
Pr_xgiven_En_log_train<-unname(cbind(log(1-trainyes),log(trainyes)))
case_predicition_non_otu<-which.max(Pr_xgiven_En_log_test)
saveRDS(Pr_xgiven_En_log_test,file = paste0('./fold_',i,'/log_prior_test','.rds'))
saveRDS(Pr_xgiven_En_log_train,file = paste0('./fold_',i,'/log_prior_train','.rds'))
saveRDS(case_predicition_non_otu,file = paste0('./fold_',i,'/case_predicition_non_otu','.rds'))
```

C.2 Fit Regularized Logistic Regression and Regularized SVM

C.2.1 Fit Logistic Regression with L_1

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/"


setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
```

```
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(glmnet)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1
nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
Y <- dataset$parkinson
lambda <- scan("/home/xiw378/canola/lassoglmm/realdata/ethnicity/lassoout/lambda.txt")
if (!exists("irep")) irep <- 1
i <- irep #folds=327
X_tr <- X[-i, ]
Y_tr <- as.numeric(Y[-i])
X_ts <- X[i, ]
Y_ts <- as.numeric(Y[i])

for(j in 1:200){
#####fit lasso
lasso.fit <- glmnet(x=X_tr, y=Y_tr, lambda=lambda[j], family = "binomial")
#####calculate the training result
tr_pred_yes <- predict(lasso.fit, newx = X_tr, type="response")
tr_pred_no <- 1 - tr_pred_yes
tr_pred_matrix <- cbind(tr_pred_no , tr_pred_yes)
tr_prediction <- max.col( tr_pred_matrix)
tr_er <- 1-mean(tr_prediction==Y_tr)
#####calculate the validation result
test_pred_yes <- t(predict(lasso.fit, newx = t(X_ts), type = "response"))
test_pred_no <- 1 - test_pred_yes
ts_pred_matrix <- cbind(test_pred_no, test_pred_yes)
ts_prediction <- max.col(ts_pred_matrix)
num_otu_retained <- length(which(lasso.fit$beta[-c(1:2),]!=0))
index_otu_retained<-which(lasso.fit$beta[-c(1:2),]!=0)
beta<-lasso.fit$beta[-c(1:2),]

saveRDS(tr_pred_matrix,file = paste0("./fold_",i,"/lambda_",j,"train_prob.rds"))
saveRDS(ts_pred_matrix,file = paste0("./fold_",i,"/lambda_",j,"test_prob.rds"))
saveRDS(tr_prediction,file = paste0("./fold_",i,"/lambda_",j,"train_point.rds"))
saveRDS(ts_prediction,file =paste0 ("./fold_",i,"/lambda_",j,"test_point.rds"))
saveRDS(num_otu_retained,file = paste0("./fold_",i,"/lambda_",j,"num_otu_retained.rds"))
saveRDS(index_otu_retained,file = paste0("./fold_",i,"/lambda_",j,"index_otu_retained.rds"))
saveRDS(tr_er,file = paste0("./fold_",i,"/lambda_",j,"train_error_rate.rds"))
saveRDS(beta,file = paste0("./fold_",i,"/lambda_",j,"beta.rds"))
}
```

C.2.2 Fit Logistic Regression with Elastic-net

For logistic regression with elastic-net penalty, after we use the existing set of λ, all performances are not good, with relatively high error rates. So, we choose the appropriate group of λ first, using cv.glemnt(), and then use the chosen group as tuning parameter in main model.

C.2.2.1 Try a Set of α

```
library(nnet)
dataset_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/"
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(glmnet)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1
nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")
fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
Y <- dataset$parkinson
set.seed(1234)
alpha<-seq(0,1,0.005)
if (!exists("irep")) irep <- 1
i <- irep #folds=201
c<-cv.glmnet(x=X, y=Y,nfolds =327, family = "binomial",alpha=alpha[i])
g<-glmnet(x=X,y=as.numeric(Y),lambda=c$lambda.min,alpha=alpha[i],family = 'binomial')
pp<-predict(g,newx=X,type="response")
p<-cbind(1-pp,pp)
po<-max.col(p)
er<-1-mean(po==as.numeric(Y))
l<-c$lambda
saveRDS(l,paste0( "./_choose/_lambda_of_alpha_",i,".rds"))
saveRDS(er,paste0( "./_choose/_er_of_alpha_",i,".rds"))
```

C.2.2.2 Choose Best Alpha and The Corresponding Group of λ

```
alpha<-seq(0,1,0.005)
ers<-vector()
for(i in 1:length(alpha)){
ers[i]<-readRDS(paste0( "./_choose/_er_of_alpha_",i,".rds"))
}
ind<-which.min(ers)
#####best lambda
bestlambda<-readRDS(paste0( "./_choose/_lambda_of_alpha_",ind,".rds"))
bestalpha<-alpha[ind]
saveRDS(bestlambda,"./_lambda.rds")
saveRDS(bestalpha,"./best_alpha.rds")
```

C.2.2.3 Logistic Regression with Elastic-net

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/"

setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(glmnet)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x)))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")
fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
Y <- dataset$parkinson
lambda<-readRDS("./_lambda.rds")
alpha_fit<-readRDS("./best_alpha.rds")
if (!exists("irep")) irep <- 1
i <- irep #folds=327

X_tr <- X[-i, ]
Y_tr <- as.numeric(Y[-i])
X_ts <- X[i, ]
Y_ts <- as.numeric(Y[i])

for(j in 1:length(lambda)){
lasso.fit <- glmnet(x=X_tr, y=Y_tr, lambda=lambda[j], family = "binomial",alpha=alpha_fit)
#####calculate the training result
tr_pred_yes <- predict(lasso.fit, newx = X_tr, type="response")
tr_pred_no <- 1 - tr_pred_yes
tr_pred_matrix <- cbind(tr_pred_no , tr_pred_yes)
tr_prediction <- max.col( tr_pred_matrix)
tr_er <- 1-mean(tr_prediction==Y_tr)
#####calculate the validation result
test_pred_yes <- t(predict(lasso.fit, newx = t(X_ts), type = "response"))
test_pred_no <- 1 - test_pred_yes
ts_pred_matrix <- cbind(test_pred_no, test_pred_yes)
ts_prediction <- max.col(ts_pred_matrix)
num_otu_retained <- length(which(lasso.fit$beta[-c(1:2),]!=0))
index_otu_retained<-which(lasso.fit$beta[-c(1:2),]!=0)
beta<-lasso.fit$beta[-c(1:2),]

saveRDS(tr_pred_matrix,file = paste0("./fold_",i,"/lambda_",j,"train_prob.rds"))
saveRDS(ts_pred_matrix,file = paste0("./fold_",i,"/lambda_",j,"test_prob.rds"))
saveRDS(tr_prediction,file = paste0("./fold_",i,"/lambda_",j,"train_point.rds"))
saveRDS(ts_prediction,file =paste0 ("./fold_",i,"/lambda_",j,"test_point.rds"))
saveRDS(num_otu_retained,file = paste0("./fold_",i,"/lambda_",j,"num_otu_retained.rds"))
saveRDS(index_otu_retained,file = paste0("./fold_",i,"/lambda_",j,"index_otu_retained.rds"))
saveRDS(tr_er,file = paste0("./fold_",i,"/lambda_",j,"train_error_rate.rds"))
saveRDS(beta,file = paste0("./fold_",i,"/lambda_",j,"beta.rds"))
}
```

C.2.3 Fit SVM with L_1

For SVM with L_1 penalty, we need tuning parameter λ. After we use the existing set of λ, performances are not so good, with relatively high error rates. So, we choose the appropriate group of λ, using cv.sparseSVM(). Here, $\alpha = 1$.

C.2.3.1 Choose The Appropriate Group of λ

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/"

setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(sparseSVM)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1
nsample <- n

Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
```

```
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson
#####the arguments of best model in our trials, with lowest minimum error rate
cv.fit <- cv.sparseSVM(X,
2*as.numeric(Y)-3, nfolds = 10, ncores = 2, seed = 1234,gamma=1,
alpha=1, preprocess = "standardize")
#####record the appropriate group of lambda as tuning parameter for fitting the main model:
lambda<-cv.fit$lambda
saveRDS(lambda,file="./_lambda.rds")
```

C.2.3.2 SVM with L_1

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/"

setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(sparseSVM)
library(nnet)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1
nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")
fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
Y <- dataset$parkinson
#X_c <- sweep(X, 2, colMeans(X), "-")
# X_s<- sweep(X_c,2, apply(X_c,2,sd), "/")
lambda <- readRDS("/home/mac744/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/_lambda.rds")
nlambda<-length(lambda)
if (!exists("irep")) irep <- 1
i <- irep #folds=327
X_tr <- X[-i, ]
Y_tr <- Y[-i]
X_ts <- X[i, ]
Y_ts <-Y[i]
for(j in 1:nlambda){
L1<-sparseSVM(X_tr, Y_tr, alpha = 1, gamma =1,lambda=lambda[j], preprocess = "standardize")
L1_tr_pred<-as.vector(predict(L1, X_tr))
L1_tr_er<-1-mean(L1_tr_pred==as.character(Y_tr))
L1_ts_pred<-as.vector(predict(L1, X_ts))
L1_intercept<-L1$weights[1]
L1_coef<-L1$weights[-1]
names(L1_coef)<-colnames(X_tr)
L1_otu_retained<-L1_coef[-c(1:2)]
L1_num_otu_retained <- length(which(L1_otu_retained!=0))
L1_index_otu_retained<-which(L1_coef[-c(1:2)]!=0)
L1_intercept<-L1$weights[1]
tr_relative_distance<-X_tr%*%L1_coef+L1_intercept
ts_relative_distance<-X_ts%*%L1_coef+L1_intercept
#####logitics regression
#####input:relative distance,
#####output:parkinson
SL_traindata<-data.frame(relative_distance=tr_relative_distance,parkinson=Y_tr)
SL_testdata<-data.frame(relative_distance=ts_relative_distance,parkinson=Y_ts)
S_logistic<-glm(parkinson~relative_distance,data=SL_traindata,family = binomial(logit))
trainyes<-predict(S_logistic,SL_traindata,type = "response")
Pr_of_PD_train<-unname(cbind(1-trainyes,trainyes))
train_pred_point<-max.col(Pr_of_PD_train)
S_L_tr_er<-mean(train_pred_point!=as.numeric(Y_tr))
testyes<-predict(S_logistic,SL_testdata,type = "response")
Pr_of_PD_test<-unname(cbind(1-testyes,testyes))
saveRDS(L1_tr_pred,file = paste0("./fold_",i,"/SVM//lambda_",j,"train_point.rds"))
saveRDS(L1_ts_pred,file =paste0 ("./fold_",i,"/SVM/lambda_",j,"test_point.rds"))
saveRDS(L1_num_otu_retained,file = paste0("./fold_",i,"/SVM/lambda_",j,"num_otu_retained.rds"))
saveRDS(L1_index_otu_retained,file = paste0("./fold_",i,"/SVM/lambda_",j,"index_otu_retained.rds"))
saveRDS(L1_tr_er,file = paste0("./fold_",i,"/SVM/lambda_",j,"train_error_rate.rds"))
saveRDS(L1_otu_retained,file = paste0("./fold_",i,"/SVM/lambda_",j,"beta.rds"))
saveRDS(L1_coef,file = paste0("./fold_",i,"/SVM/lambda_",j,"coef.rds"))
saveRDS(L1_intercept,file = paste0("./fold_",i,"/SVM/lambda_",j,"intercept.rds"))
saveRDS(tr_relative_distance,file = paste0("./fold_",i,"/SVM/lambda_",j,"tr_relative_distance.rds"))
saveRDS(ts_relative_distance,file = paste0("./fold_",i,"/SVM/lambda_",j,"ts_relative_distance.rds"))
#####logistic result:
saveRDS(Pr_of_PD_train,file = paste0("./fold_",i,"/S_L/lambda_",j,"Pr_of_PD_train.rds"))
saveRDS(Pr_of_PD_test,file = paste0("./fold_",i,"/S_L/lambda_",j,"Pr_of_PD_test.rds"))
saveRDS(S_L_tr_er,file = paste0("./fold_",i,"/S_L/lambda_",j,"train_error.rds"))}
```

C.2.4 Fit SVM with Elastic-net

For SVM with Elastic-net, after we use the existing set of λ, performances are not so good, with relatively high error rates. So, we also choose a $\alpha = 1$, and the appropriate group of λ, usin cv.sparseSVM(). Here.

C.2.4.1 Choose Best α and The Corresponding Group of λ

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/"

setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
```

```
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(sparseSVM)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1
nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson
#####after input a series of alpha between (0,1),
#####we find alpha=0.5 performs a little better than others.

cv.fit <- cv.sparseSVM(X, 2*as.numeric(Y)-3, nfolds = 10, ncores = 2, seed = 1234,
alpha=0.5,gamma=1, preprocess = "standardize")

lambda<-cv.fit$lambda
saveRDS(lambda,file="./_lambda.rds")
```

C.2.4.2 SVM with Elastic-net

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/"

setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(sparseSVM)
library(nnet)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
```

```
X_cov<-c("age","sex")
fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson
lambda <- readRDS("/home/mac744/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/_lambda.rds")
#lambda <- scan("/home/xiw378/canola/lassoglmm/realdata/ethnicity/lassoout/lambda.txt")

nlambda<-length(lambda)
if (!exists("irep")) irep <- 1
i <- irep #folds=327

X_tr <- X[-i, ]
Y_tr <- Y[-i]
X_ts <- X[i, ]
Y_ts <-Y[i]
for(j in 1:nlambda){
ELA<-sparseSVM(X_tr, Y_tr, alpha = 0.5, gamma =1,lambda=lambda[j], preprocess = "standardize")
ELA_tr_pred<-as.vector(predict(ELA, X_tr))
ELA_tr_er<-1-mean(ELA_tr_pred==as.character(Y_tr))
ELA_ts_pred<-as.vector(predict(ELA, X_ts))
ELA_intercept<-ELA$weights[1]
ELA_coef<-ELA$weights[-1]
names(ELA_coef)<-colnames(X_tr)
ELA_otu_retained<-ELA_coef[-c(1:2)]
ELA_num_otu_retained <- length(which(ELA_otu_retained!=0))
ELA_index_otu_retained<-which(ELA_coef[-c(1:2)]!=0)
ELA_intercept<-ELA$weights[1]
tr_relative_distance<-X_tr%*%ELA_coef+ELA_intercept
ts_relative_distance<-X_ts%*%ELA_coef+ELA_intercept
#####logitics regression
#####input:relative distance,
#####output:parkinson
SL_traindata<-data.frame(relative_distance=tr_relative_distance,parkinson=Y_tr)
SL_testdata<-data.frame(relative_distance=ts_relative_distance,parkinson=Y_ts)
S_logistic<-glm(parkinson~relative_distance,data=SL_traindata,family = binomial(logit))
trainyes<-predict(S_logistic,SL_traindata,type = "response")
Pr_of_PD_train<-unname(cbind(1-trainyes,trainyes))
train_pred_point<-max.col(Pr_of_PD_train)
S_L_tr_er<-mean(train_pred_point!=as.numeric(Y_tr))
testyes<-predict(S_logistic,SL_testdata,type = "response")
Pr_of_PD_test<-unname(cbind(1-testyes,testyes))
saveRDS(ELA_tr_pred,file = paste0("./fold_",i,"/SVM//lambda_",j,"train_point.rds"))
saveRDS(ELA_ts_pred,file =paste0 ("./fold_",i,"/SVM/lambda_",j,"test_point.rds"))
saveRDS(ELA_num_otu_retained,file = paste0("./fold_",i,"/SVM/lambda_",j,"num_otu_retained.rds"))
saveRDS(ELA_index_otu_retained,file = paste0("./fold_",i,"/SVM/lambda_",j,"index_otu_retained.rds"))
saveRDS(ELA_tr_er,file = paste0("./fold_",i,"/SVM/lambda_",j,"train_error_rate.rds"))
saveRDS(ELA_otu_retained,file = paste0("./fold_",i,"/SVM/lambda_",j,"beta.rds"))
saveRDS(ELA_coef,file = paste0("./fold_",i,"/SVM/lambda_",j,"coef.rds"))
saveRDS(ELA_intercept,file = paste0("./fold_",i,"/SVM/lambda_",j,"intercept.rds"))
saveRDS(tr_relative_distance,file = paste0("./fold_",i,"/SVM/lambda_",j,"tr_relative_distance.rds"))
saveRDS(ts_relative_distance,file = paste0("./fold_",i,"/SVM/lambda_",j,"ts_relative_distance.rds"))
######logistic result:
saveRDS(Pr_of_PD_train,file = paste0("./fold_",i,"/S_L/lambda_",j,"Pr_of_PD_train.rds"))
saveRDS(Pr_of_PD_test,file = paste0("./fold_",i,"/S_L/lambda_",j,"Pr_of_PD_test.rds"))
saveRDS(S_L_tr_er,file = paste0("./fold_",i,"/S_L/lambda_",j,"train_error.rds"))}
```

C.3 Calculate Test Error Rates of Models with Different λ

C.3.1 Test Error Rates of Regularized Logistic Regression with Different λ

```
######for logistic regression with L1:
dataset_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/"
lambda <- scan("/home/xiw378/canola/lassoglmm/realdata/ethnicity/lassoout/lambda.txt")
######for logistic regression with Elastic-net:
dataset_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/"
lambda<-readRDS(paste0(result_path,"_lambda.rds"))
########################################################################
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(glmnet)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1
nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")
fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson

if (!exists("irep")) irep <- 1
j <- irep #200
test_point<-vector()
for (i in 1:folds){
test_point[i]<-readRDS(paste0 ("./fold_",i,"/lambda_",j,"test_point.rds"))
}

test_error<-1-mean(test_point==as.numeric(Y))
saveRDS(test_error,file = paste0(result_path,"_test_error/lambda_",j,".rds"))
```

C.3.2 Test Error Rates of Regularized SVM with Different λ

```
#####SVM with L1
dataset_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/"
#####################################################################
#####SVM with Elastic-net
dataset_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/"
#####################################################################
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(glmnet)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson
lambda <- readRDS(paste0(result_path,"_lambda.rds"))
nlambda<-length(lambda)
if (!exists("irep")) irep <- 1
j <- irep #100

SVM_test_point<-vector()
S_L_test_point<-vector()
for (i in 1:folds){
SVM_test_point[i]<-readRDS(paste0 ("./fold_",i,"/SVM/lambda_",j,"test_point.rds"))
S_L_test_point[i]<-max.col(readRDS(paste0("./fold_",i,"/S_L/lambda_",j,"Pr_of_PD_test.rds")))}

SVM_test_error<-1-mean(SVM_test_point==as.character(Y))
S_L_test_error<-1-mean(S_L_test_point==as.numeric(Y))
saveRDS(SVM_test_error,file = paste0("./_test_error/SVM_lambda_",j,".rds"))
saveRDS(SVM_test_point,file = paste0("./_test_error/SVM_lambda_",j,"_point.rds"))

saveRDS(S_L_test_error,file = paste0("./_test_error/S_L_lambda_",j,".rds"))
saveRDS(S_L_test_point,file = paste0("./_test_error/S_L_lambda_",j,"_point.rds"))
```

C.4 Modifications of Names of The Top Selected OTUs

```
#####logistic regression with L1
dataset_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/"
lambda <- scan("/home/xiw378/canola/lassoglmm/realdata/ethnicity/lassoout/lambda.txt")

#####logistic regression with L1
dataset_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/"
lambda <-readRDS(paste0(result_path,"_lambda.rds"))
############################################################
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
genus_names <- one_dataset$genus_names

otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(sparseSVM)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson
nlambda<-length(lambda)

test_error<-vector()
for(i in 1:nlambda){
test_error[i]<-readRDS(paste0("./_test_error/lambda_",i,".rds"))}

best_lam.n<-which.min(test_error)
beta<-matrix(NA,ncol=382,nrow=327)
for(i in 1:n){
beta[i,]<-readRDS(paste0("./fold_",i,"/lambda_",best_lam.n,"beta.rds"))
}
colnames(beta)<-names(readRDS(paste0("./fold_",i,"/lambda_",best_lam.n,"beta.rds")))
median_absolute<-apply(beta, 2, function(x) median(abs(x)))
medians<-apply(beta, 2, function(x) median(x))
```

```
non_zero_n<-sum(medians!=0)
non_zero_index<-which(medians!=0)
ab_me_order.n<-order(median_absolute,decreasing=TRUE)
ab_me_order<-colnames(beta)[ab_me_order.n]
sign_of_top_beta<-sign(medians)[ab_me_order.n[1:non_zero_n]]
selected_genus<-genus_names[ab_me_order.n[1:non_zero_n]]
top_otus<-vector()
A<-vector()
for(i in 1:length(selected_genus)){
a<-unlist(strsplit(selected_genus[i],"f__"))
A[i]<-a[length(a)]
if(is.na(unlist(strsplit(A[i],".g__"))[2])==TRUE)
{B<-c(unlist(strsplit(A[i],".g__")),"unclassified")}else{B<-unlist(strsplit(A[i],".g__"))}
top_otus[i]<-paste0(B[1],'/',B[2])
}

##Here we need to delete some unnecessary dots by hands. Finally:

saveRDS(top_otus,file = "./_top_genera_names.rds")
###########################################################


###########For OTUs selected by SVM with L1#################



dataset_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/"
#####################################################################
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
genus_names <- one_dataset$genus_names

otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index
##############################
library(sparseSVM)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
```

```
Y <- dataset$parkinson

lambda <- readRDS("/home/mac744/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/_lambda.rds")

nlambda<-length(lambda)
test_error<-vector()
for(i in 1:nlambda){
test_error[i]<-readRDS(paste0("./_test_error/S_L_lambda_",i,".rds"))}
############################
best_lam.n<-which.min(test_error)
beta<-matrix(NA,ncol=382,nrow=327)
for(i in 1:n){
beta[i,]<-readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds"))
}

colnames(beta)<-names(readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds")))
median_absolute<-apply(beta, 2, function(x) median(abs(x)))
medians<-apply(beta, 2, function(x) median(x))
non_zero_n<-sum(medians!=0)
non_zero_index<-which(medians!=0)
ab_me_order.n<-order(median_absolute,decreasing=TRUE)
ab_me_order<-colnames(beta)[ab_me_order.n]
sign_of_top_beta<-sign(medians)[ab_me_order.n[1:non_zero_n]]
selected_genus<-genus_names[ab_me_order.n[1:non_zero_n]]
top_otus<-vector()
A<-vector()
for(i in 1:length(selected_genus)){a<-unlist(strsplit(selected_genus[i],"f__"))
A[i]<-a[length(a)]
if(is.na(unlist(strsplit(A[i],".g__"))[2])==TRUE){B<-c(unlist(strsplit(A[i],".g__")),"unclassified")}els
top_otus[i]<-paste0(B[1],'/',B[2])
}

##Here we need to delete some unnecessary dots by hands. Finally:

saveRDS(top_otus,file = "./_top_genera_names.rds")
##############################################################################################
################For OTUs selected by SVM with Elastic-net########################


dataset_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/"
#####################################################################
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
genus_names <- one_dataset$genus_names

otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
```

```
folds.index<-one_dataset$folds.index
##############################
library(sparseSVM)
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1


nsample <- n

Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson
lambda <- readRDS("/home/mac744/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/_lambda.rds")


nlambda<-length(lambda)
test_error<-vector()

for(i in 1:nlambda){
test_error[i]<-readRDS(paste0("./_test_error/S_L_lambda_",i,".rds"))}


############################
best_lam.n<-which.min(test_error)
beta<-matrix(NA,ncol=382,nrow=327)
for(i in 1:n){
beta[i,]<-readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds"))
}
colnames(beta)<-names(readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds")))
median_absolute<-apply(beta, 2, function(x) median(abs(x)))
medians<-apply(beta, 2, function(x) median(x))
non_zero_n<-sum(medians!=0)
non_zero_index<-which(medians!=0)
ab_me_order.n<-order(median_absolute,decreasing=TRUE)
ab_me_order<-colnames(beta)[ab_me_order.n]
sign_of_top_beta<-sign(medians)[ab_me_order.n[1:non_zero_n]]
selected_genus<-genus_names[ab_me_order.n[1:non_zero_n]]
top_otus<-vector()
A<-vector()
for(i in 1:length(selected_genus)){a<-unlist(strsplit(selected_genus[i],"f__"))
A[i]<-a[length(a)]
if(is.na(unlist(strsplit(A[i],".g__"))[2])==TRUE){B<-c(unlist(strsplit(A[i],".g__")),"unclassified")}els
top_otus[i]<-paste0(B[1],'/',B[2])
}
##Here we need to delete some unnecessary dots by hands. Finally:

saveRDS(top_otus,file = "./_top_genera_names.rds")
```

C.5 Evaluations of Models, Visualization of Metrics, Plots

C.5.1 Results of Logistic Regression with L_1

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_lasso_loocv_logbyT/"
#####################################################################
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index
##############################
library(glmnet)
library(pROC)
library(ROCR)
library(PRROC)
library(reshape2)
#library (HTLR)
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]

Y <- dataset$parkinson
lambda <- scan("/home/xiw378/canola/lassoglmm/realdata/ethnicity/lassoout/lambda.txt")
nlambda<-length(lambda)

X_c <- sweep(X, 2, colMeans(X), "-")
X_n <- sweep(X_c,2, apply(X_c,2,sd), "/")
exprSet=data.frame(X_n[,-c(1:2)],parkinson=Y)
sample_order<-order(Y)
test_error<-vector()
for(i in 1:nlambda){
test_error[i]<-readRDS(paste0("./_test_error/lambda_",i,".rds"))}

nlambda <- length (lambda)
n <- one_dataset$n
```

```
pred_matrix <- array (0, dim = c(n,2,nlambda))
for (i in 1:n) {
for (j in 1:nlambda){
pred_matrix[i,,j] <- readRDS(paste0(result_path,"fold_",i,"/lambda_",j,"test_prob.rds"))}}
saveRDS(pred_matrix, file = paste0(result_path, "allpred_matrix.rds"))
pred_matrix<-readRDS(paste0(result_path, "allpred_matrix.rds"))

non_otu_predmatrix<-matrix(NA,ncol=2,nrow=327)
for(i in 1:327){non_otu_predmatrix[i,]<-readRDS(paste0("./fold_",i,"/log_prior_test.rds"))}
saveRDS(non_otu_predmatrix,file = "./non_otu_predmatrix.rds")
non_otu_predmatrix<-exp(readRDS("./non_otu_predmatrix.rds"))
n <- one_dataset$n

## find predictive metrics against lambda
y_true <- as.numeric(one_dataset$dataset$parkinson)
non_otu_er<-mean(y_true!=max.col(non_otu_predmatrix))
amlp_lasso <- rep(0, nlambda)
er_lasso <- rep(0, nlambda)

for (i in 1:nlambda) {
eval <- evaluate_pred (pred_matrix[,,i],y_true)
er_lasso[i] <- eval$er
amlp_lasso [i] <- eval$amlp
}

base<-matrix(0.5,nrow=327,ncol=2)
amlp_non_otu<-evaluate_pred (non_otu_predmatrix,y_true)$amlp
er_non_otu<-evaluate_pred (non_otu_predmatrix,y_true)$er
amlp_base<-evaluate_pred (base,y_true)$amlp
AUC<-vector()
AU_PR<-vector()
for (i in 1:nlambda){
AUC[i]<-roc(y_true,pred_matrix[,2,i])$auc
scores <- data.frame(pre=pred_matrix[,2,i], lab=y_true)
AU_PR[i] <- pr.curve(scores.class0=scores[scores$lab=="2",]$pre,
scores.class1=scores[scores$lab=="1",]$pre,
curve=T)$auc.integral}
scores_nonotu <- data.frame(pre=exp(non_otu_predmatrix[,2]), lab=y_true)
AU_PR_non_otu <- pr.curve(scores.class0=scores_nonotu[scores_nonotu$lab=="2",]$pre,
scores.class1=scores_nonotu[scores_nonotu$lab=="1",]$pre,
curve=T)$auc.integral

AUC_non_otu<-roc(y_true,non_otu_predmatrix[,2])$auc

which.max(AUC)
which.min(er_lasso)
which.min(amlp_lasso)
best_lam.n<-which.min(er_lasso)

pred <- prediction(pred_matrix[,2,best_lam.n], y_true)
non_otu_pred<-prediction(non_otu_predmatrix[,2],y_true)
perf <- performance(pred,"tpr","fpr")
perf1 <- performance(pred, "prec", "rec")
non_otu_perf<-performance(non_otu_pred, "prec", "rec")
```

```
##########AUC/AUPR#########
pdf(file=paste0("./__final_plots/AUC_and_AUC-PR.pdf"),width=6,height = 6)
par(mar=c(4,4,2,1))
par(mar=c(3,4,2,4))
plot(log(lambda), AUC,col="1",type="l",lwd=2,xlab=" ",ylab=" ",lty=1)
par(new=TRUE)
plot(log(lambda), AU_PR,col="2",type="l",lwd=2,axes=F,ylab=" ",xlab="",lty=4)
axis(side=4,col.axis="black")
mtext(side=1, line=2, expression(log(lambda)), cex=1)
mtext(side=4, line=2, "AUPRC", cex=1.2)
mtext(side=2, line=2, "AUC", cex=1.2)
abline(h=seq(0.7,0.88,by=0.005),v=seq(-12,-2,by=1),col="grey",lty=3)
legend('topleft', c('AUPRC', 'AUC'),col=c('red', 'black'),lty=c(4,1),bty = "o",cex=0.8,bg='white')
dev.off()

#########Error raete#######
pdf(file=paste0("./__final_plots/Lasso_Error_Rate.pdf"),width=6,height = 6)
par(mar=c(3,4,1,4))
plot(log(lambda), er_lasso,col="blue",type="l",lwd=2,ylim=c(0.2,0.40),xlab="", ylab="")
abline(h=non_otu_er,col="darkgreen",lty=2,lwd=2)
abline(h=130/327,col="red",lty=3,lwd=2.5)
axis(side = 4, at=seq(0,130/327,by=130/327/10), labels=seq(1,0,by=-0.1))
mtext(side=4, line=2, expression(R^2), cex=1.5,las=2)
mtext(side=2, line=2, "ER", cex=1.5,las=2)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0.2,0.44,by=0.01),v=seq(-12,-3,by=1),col="grey",lty=3)
legend('bottomleft', c('OTU+age+sex', 'age+sex','baseline'),col =c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

#########amlp#######
pdf(file=paste0("./__final_plots/lasso_logistic_amlp.pdf"),width=6,height = 6)
par(mar=c(3,4,1,4))
plot(log(lambda), amlp_lasso,col="blue",type="l",lwd=2,xlab="", ylab="")
abline(h=amlp_non_otu,col="darkgreen",lty=2,lwd=2)
abline(h=amlp_base,col="red",lty=3,lwd=2.5)
mtext(side=2, line=2, "amlp", cex=1.5,las=3)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0,4.2,by=0.2),v=seq(-12,-1,by=1),col="grey",lty=3)
legend('topright', c('OTU+age+sex', 'age+sex','baseline'),col=c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

### Probability of Parkinson######
pdf(file=paste0("./__final_plots/Probability of Parkinson.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
par(mfrow=c(1,1))
PDorder<-order(y_true)
fr<-data.frame(p=pred_matrix[sample_order,,best_lam.n], l=y_true)
plot(pred_matrix[sample_order,2,best_lam.n],c(1:327),pch=2*y_true[sample_order]-1,
col=-2*y_true[PDorder]+6,xlab="Probability of Parkinson",ylab="Subject ID")
abline(v=0.5,lty=3,lwd=1)
dev.off()
```

```
#########ROC#########
pdf(file=paste0("./__final_plots/roc.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(roc(y_true,pred_matrix[,2,best_lam.n]),col="blue",main = "",lty=1,lwd=2)
lines(roc(y_true,exp(non_otu_predmatrix[,2])),col='darkgreen',lty=2,lwd=2)
legend('bottomright', c('OTU+age+sex, AUC=0.83', 'age+sex, AUC=0.66'),col  =c('blue', 'darkgreen'),
bty = "o",cex=0.8,lty=c(1,2),lwd=2)
dev.off()

######### P_R curve #########
pdf(file=paste0("./__final_plots/PRC.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(perf1,col="blue",main  = "",ylim=c(0.6,1),lty=1,lwd=2)
lines(non_otu_perf@x.values[[1]],non_otu_perf@y.values[[1]],col="darkgreen",lty=2,lwd=2)
abline(h=197/327,lty=3,col="red",lwd=2)
legend('topright', c('OTU+age+sex, AUPR=0.88', 'age+sex, AUPR=0.72, ','baseline'),
col  =c('blue', 'darkgreen','red'),
bty = "o",cex=0.8,lty=c(1,2,3),lwd=2)
dev.off()

#############beta################
beta<-matrix(NA,ncol=382,nrow=327)
for(i in 1:n){
beta[i,]<-readRDS(paste0("./fold_",i,"/lambda_",best_lam.n,"beta.rds"))
}
colnames(beta)<-names(readRDS(paste0("./fold_",i,"/lambda_",best_lam.n,"beta.rds")))
median_absolute<-apply(beta, 2, function(x) median(abs(x)))
medians<-apply(beta, 2, function(x) median(x))
non_zero_n<-sum(medians!=0)
non_zero_index<-which(medians!=0)
ab_me_order.n<-order(median_absolute,decreasing=TRUE)
ab_me_order<-colnames(beta)[ab_me_order.n]
sign_of_top_beta<-sign(medians)[ab_me_order.n[1:25]]
which(sign_of_top_beta==-1)
top_number<-25
rank<-ab_me_order.n
ordered_exprSet <- exprSet[sample_order, c("parkinson",ab_me_order)]
top_genera<-readRDS("./_top_genera_names.rds")
b_data<-data.frame(beta[,rank[1:top_number]],id=rownames(dataset))
beta_data<-melt(b_data,id.vars = c('id') )

##top 20 beta
pdf(file=paste0("./__final_plots/top 20 beta_absolute.pdf"),width=18,height = 10)
boxplot(abs(b_data[1:20]),col = -1*sign_of_top_beta+3,outcol=-1*sign_of_top_beta+3,
names=rep("",time=20),par(mar=c(8.6,11.3,1,1)),xaxt='n')
labels <- top_genera[1:20]
text(seq(1,20,by=1), par("usr")[3] - 0.005, srt = 21, adj = 1, labels =labels, xpd = TRUE,cex=1.5)
mtext(side=2, line=3, "Absolute Values of Coefficients", cex=1.8)
abline (v=seq(1.5,20,by = 1), col = "darkgrey", lty=2,lwd=4)
abline (h=seq(0,0.35,0.025), col = "lightgrey", lty=3,lwd=1)
legend('topright', c('-positive', '-negative'),fill =c('red', 'blue'),bty = "o",cex=1.4,bg="white")
dev.off()
```

```
##############################
ordered_exprSet_add<-matrix(NA,nrow=328,ncol=382)
for (i in 1:382){ordered_exprSet_add[,i]<-c(ordered_exprSet[,i+1],NA)}
for (i in 1:382){ordered_exprSet_add[,i]<-as.numeric(ordered_exprSet_add[,i])}
parkinson_add<-c(as.numeric(ordered_exprSet[,1]),3)
parkinson_add<-as.factor(parkinson_add)
fit<-data.frame(parkinson_add,ordered_exprSet_add)
colnames(fit)<-colnames(ordered_exprSet)
oadd_data1<-melt(fit[,1:(top_number+1)],id.vars = c("parkinson") )
oadd_data2<-melt(fit[,1:(20+1)],id.vars = c("parkinson") )

######### boxplot pf normalized OTU ############
pdf(file=paste0("./__final_plots/boxplot of top 20 OTU.pdf"),width=18,height = 10)
boxplot(value~parkinson+variable, data=oadd_data2,col=rep(c(4,2,1),20),
names=rep("",time=60),
par(mar=c(8.6,11.7,1,1)),outcol=rep(c(4,2,1),20),xaxt="n")
labels <- top_genera[1:20]
ylab="Standardlized Log Relative Abundances"
text(seq(1.5,60,by=3), par("usr")[3] - 0.12, srt = 22, adj = 1, labels =labels, xpd = TRUE,cex=1.5)
mtext(side=2, line=3, "Standardlized Log Relative Abundance", cex=1.8)
abline (v=seq(3,57,by = 3), col = "darkgrey", lty=2,lwd=4)
abline (h=seq(-4,8,by = 1), col = "lightgrey", lty=3,lwd=1)
legend('topright', c('-parkinson', '-control'),fill =c('red', 'blue'),bty = "o",cex=1.4,bg='white')
dev.off()
```

C.5.2 Results of Logistic Regression with Elastic-net

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_elastic_loocv_logbyT/"

setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(glmnet)
library(pROC)
library(ROCR)
library(PRROC)
library(reshape2)
#library (HTLR)
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
```

```
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
Y <- dataset$parkinson
lambda<-readRDS("./_lambda.rds")
nlambda<-length(lambda)

X_c <- sweep(X, 2, colMeans(X), "-")
X_n <- sweep(X_c,2, apply(X_c,2,sd), "/")
exprSet=data.frame(X_n[,-c(1:2)],parkinson=Y)
sample_order<-order(Y)
test_error<-vector()

for(i in 1:nlambda){
test_error[i]<-readRDS(paste0("./_test_error/lambda_",i,".rds"))}
non_otu_predmatrix<-matrix(NA,ncol=2,nrow=327)
for(i in 1:327){non_otu_predmatrix[i,]<-readRDS(paste0("./fold_",i,"/log_prior_test.rds"))}
saveRDS(non_otu_predmatrix,file = "./non_otu_predmatrix.rds")
non_otu_predmatrix<-exp(readRDS(paste0(result_path,"non_otu_predmatrix.rds")))
nlambda <- length (lambda)
n <- one_dataset$n

pred_matrix <- array (0, dim = c(n,2,nlambda))
for (i in 1:n) {
for (j in 1:nlambda)
pred_matrix[i,,j] <- readRDS(paste0(result_path,"fold_",i,"/lambda_",j,"test_prob.rds"))}
saveRDS(pred_matrix, file = paste0(result_path, "all_pred_matrix.rds"))
pred_matrix<-readRDS(paste0(result_path, "all_pred_matrix.rds"))

n <- one_dataset$n
## find predictive metrics against lambda
y_true <- as.numeric(one_dataset$dataset$parkinson)
non_otu_er<-mean(y_true!=max.col(non_otu_predmatrix))
amlp_ela <- rep(0, nlambda)
er_ela <- rep(0, nlambda)
for (i in 1:nlambda) {
eval <- evaluate_pred (pred_matrix[,,i],y_true)
er_ela[i] <- eval$er
amlp_ela[i] <- eval$amlp
}
base<-matrix(0.5,nrow=327,ncol=2)
amlp_non_otu<-evaluate_pred (non_otu_predmatrix,y_true)$amlp
amlp_base<-evaluate_pred (base,y_true)$amlp

AUC<-vector()
AU_PR<-vector()
for (i in 1:nlambda){
AUC[i]<-roc(y_true,pred_matrix[,2,i])$auc
scores <- data.frame(pre=pred_matrix[,2,i], lab=y_true)
AU_PR[i] <- pr.curve(scores.class0=scores[scores$lab=="2",]$pre,
scores.class1=scores[scores$lab=="1",]$pre,
curve=T)$auc.integral}
scores_nonotu <- data.frame(pre=exp(non_otu_predmatrix[,2]), lab=y_true)
AU_PR_non_otu <- pr.curve(scores.class0=scores_nonotu[scores_nonotu$lab=="2",]$pre,
scores.class1=scores_nonotu[scores_nonotu$lab=="1",]$pre,
curve=T)$auc.integral
AUC_non_otu<-roc(y_true,non_otu_predmatrix[,2])$auc

which.max(AUC)
which.min(er_ela)
which.min(amlp_ela)
best_lam.n<-which.min(er_ela)

pred <- prediction( pred_matrix[,2,best_lam.n], y_true)
non_otu_pred<-prediction(non_otu_predmatrix[,2],y_true)
perf <- performance(pred,"tpr","fpr")
perf1 <- performance(pred, "prec", "rec")
non_otu_perf<-performance(non_otu_pred, "prec", "rec")

###########AUC/AUPR#########
pdf(file=paste0("./__final_plots/AUC_and_AUC-PR.pdf"),width=6,height = 6)
par(mar=c(4,4,2,1))
par(mar=c(3,4,2,4))
plot(log(lambda), AUC,col="1",type="l",lwd=2,xlab=" ",ylab=" ",lty=1)
par(new=TRUE)
plot(log(lambda), AU_PR,col="2",type="l",lwd=2,axes=F,ylab=" ",xlab="",lty=4)
axis(side=4,col.axis="black")
mtext(side=1, line=2, expression(log(lambda)), cex=1)
mtext(side=4, line=2, "AUPRC", cex=1.2)
mtext(side=2, line=2, "AUC", cex=1.2)
abline(h=seq(0.3,1.01,by=0.02),v=seq(-4,2,by=0.5),col="grey",lty=3)
legend('bottomleft', c('AUPRC', 'AUC'),col=c('red', 'black'),lty=c(4,1),bty = "o",
cex=0.8,bg="white")
dev.off()

#########Error raete#######
pdf(file=paste0("./__final_plots/elastic_logistic__Error_Rate.pdf"),width=6,height = 6)
par(mar=c(3,4,1,4))
plot(log(lambda), er_ela,col="blue",type="l",lwd=2,ylim=c(0.2,0.40),xlab="", ylab="")
abline(h=non_otu_er,col="darkgreen",lty=2,lwd=2)
abline(h=130/327,col="red",lty=3,lwd=2.5)
axis(side = 4, at=seq(0,130/327,by=130/327/10), labels=seq(1,0,by=-0.1))
mtext(side=4, line=2, expression(R^2), cex=1.5,las=2)
mtext(side=2, line=2, "ER", cex=1.5,las=2)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0.18,0.44,by=0.01),v=seq(-4,2,by=0.5),col="grey",lty=3)
legend('bottomleft', c('OTU+age+sex', 'age+sex','baseline'),col  =c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

#########amlp#######
pdf(file=paste0("./__final_plots/elastic_logistic_amlp.pdf"),width=6,height = 6)
par(mar=c(3,4,1,4))

plot(log(lambda), amlp_ela,col="blue",type="l",lwd=2,xlab="", ylab="",ylim=c(0.5,0.70))
```

```
abline(h=amlp_non_otu,col="darkgreen",lty=2,lwd=2)
abline(h=amlp_base,col="red",lty=3,lwd=2.5)
mtext(side=2, line=2, "amlp", cex=1.5,las=3)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0.48,0.8,by=0.01),v=seq(-4,2,by=0.5),col="grey",lty=3)
legend('bottomright', c('OTU+age+sex', 'age+sex','baseline'),col  =c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

#########Probability of Parkinson#######
pdf(file=paste0("./__final_plots/Probability of Parkinson.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
par(mfrow=c(1,1))
PDorder<-order(y_true)
fr<-data.frame(p=pred_matrix[sample_order,,best_lam.n], l=y_true)
plot(pred_matrix[sample_order,2,best_lam.n],c(1:327),pch=2*y_true[sample_order]-1,
col=-2*y_true[PDorder]+6,xlab="Probability of Parkinson",ylab="Subject ID")
abline(v=0.5,lty=3,lwd=1)
dev.off()

#########ROC#########
pdf(file=paste0("./__final_plots/roc.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(roc(y_true,pred_matrix[,2,best_lam.n]),col="blue",main = "",lty=1,lwd=2)
lines(roc(y_true,exp(non_otu_predmatrix[,2])),col='darkgreen',lty=2,lwd=2)
legend('bottomright', c(paste0('OTU+age+sex, AUC=',round(AUC[best_lam.n],2)
), paste0('age+sex, AUC=',round(AUC_non_otu,2))),col  =c('blue', 'darkgreen'),bty = "o",cex=0.8,
lty=c(1,2),lwd=2)
dev.off()

######### P_R curve #########
pdf(file=paste0("./__final_plots/PRC.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(perf1,col="blue",main  = "",ylim=c(0.6,1),lty=1,lwd=2)
lines(non_otu_perf@x.values[[1]],non_otu_perf@y.values[[1]],col="darkgreen",lty=2,lwd=2)
abline(h=197/327,lty=3,col="red",lwd=2)
legend('topright', c(paste0('OTU+age+sex, AUPR=',round(AU_PR[best_lam.n],2)),
paste0('age+sex, AUPR=',round(AU_PR_non_otu,2) ),'baseline'),
col  =c('blue', 'darkgreen','red'),bty = "o",cex=0.8,lty=c(1,2,3),lwd=2)
dev.off()

#############beta###############
beta<-matrix(NA,ncol=382,nrow=327)
for(i in 1:n){
beta[i,]<-readRDS(paste0("./fold_",i,"/lambda_",best_lam.n,"beta.rds"))
}
colnames(beta)<-names(readRDS(paste0("./fold_",i,"/lambda_",best_lam.n,"beta.rds")))
median_absolute<-apply(beta, 2, function(x) median(abs(x)))
medians<-apply(beta, 2, function(x) median(x))
non_zero_n<-sum(medians!=0)
non_zero_index<-which(medians!=0)
ab_me_order.n<-order(median_absolute,decreasing=TRUE)
ab_me_order<-colnames(beta)[ab_me_order.n]
sign_of_top_beta<-sign(medians)[ab_me_order.n[1:25]]
```

```
which(sign_of_top_beta==-1)
top_number<-25
rank<-ab_me_order.n
ordered_exprSet <- exprSet[sample_order, c("parkinson",ab_me_order)]
top_genera<-readRDS("./_top_genera_names.rds")
b_data<-data.frame(beta[,rank[1:top_number]],id=rownames(dataset))
beta_data<-melt(b_data,id.vars = c('id') )
#top 20 beta
pdf(file=paste0("./__final_plots/top 20 beta_absolute.pdf"),width=18,height = 10)
boxplot(abs(b_data[1:20]),col = -1*sign_of_top_beta+3,outcol=-1*sign_of_top_beta+3,
names=rep("",time=20),par(mar=c(8.2,10.3,1,1)),xaxt='n')
labels <- top_genera[1:20]
text(seq(1,20,by=1), par("usr")[3] - 0.004, srt = 21, adj = 1, labels =labels, xpd = TRUE,cex=1.5)
mtext(side=2, line=3, "Absolute Values of Coefficients", cex=1.8)
abline (v=seq(1.5,20,by = 1), col = "darkgrey", lty=2,lwd=4)
abline (h=seq(0,0.25,0.025), col = "lightgrey", lty=3,lwd=1)
legend('topright', c('-positive', '-negative'),fill =c('red', 'blue'),bty = "o",cex=1.4,bg="white")
dev.off()

#############################
ordered_exprSet_add<-matrix(NA,nrow=328,ncol=382)
for (i in 1:382){ordered_exprSet_add[,i]<-c(ordered_exprSet[,i+1],NA)}
for (i in 1:382){ordered_exprSet_add[,i]<-as.numeric(ordered_exprSet_add[,i])}
parkinson_add<-c(as.numeric(ordered_exprSet[,1]),3)
parkinson_add<-as.factor(parkinson_add)
fit<-data.frame(parkinson_add,ordered_exprSet_add)
colnames(fit)<-colnames(ordered_exprSet)
#ordered_exprSet_add[,1]<-factor(ordered_exprSet_add[,1])
oadd_data1<-melt(fit[,1:(top_number+1)],id.vars = c("parkinson") )
oadd_data2<-melt(fit[,1:(20+1)],id.vars = c("parkinson") )

######### boxplot pf normalized OTU ############
pdf(file=paste0("./__final_plots/boxplot of top 20 OTU.pdf"),width=18,height = 10)
boxplot(value~parkinson+variable, data=oadd_data2,col=rep(c(4,2,1),20),names=rep("",time=60),
par(mar=c(8.2,10.8,1,1)),outcol=rep(c(4,2,1),20),xaxt="n")
labels <- top_genera[1:20]
ylab="Standardlized Log Relative Abundance"
text(seq(1.5,60,by=3), par("usr")[3] - 0.22, srt = 21, adj = 1, labels =labels, xpd = TRUE,cex=1.5)
mtext(side=2, line=3, "Standardlized Log Relative Abundances", cex=1.8)
abline (v=seq(3,57,by = 3), col = "darkgrey", lty=2,lwd=4)
abline (h=seq(-5,10,2.5), col = "lightgrey", lty=3,lwd=1)
legend('topright', c('-parkinson', '-control'),fill =c('red', 'blue'),bty = "o",cex=1.4,bg='white')
dev.off()
```

C.5.3 Results of SVM with L_1

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/"
#####################################################################
setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
```

```
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
genus_names<-one_dataset$genus_names
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index
##############################
library(glmnet)
library(pROC)
library(ROCR)
library(PRROC)
#library (HTLR)
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
Y <- dataset$parkinson
lambda <- readRDS("/home/mac744/ParkinsonBayes/dataset_real2_svml1_loocv_logbyT/_lambda.rds")
nlambda<-length(lambda)
X_c <- sweep(X, 2, colMeans(X), "-")
X_n <- sweep(X_c,2, apply(X_c,2,sd), "/")
exprSet=data.frame(X_n[,-c(1:2)],parkinson=Y)
sample_order<-order(Y)

test_error<-rep(0,times=nlambda)
for(i in 1:nlambda){
test_error[i]<-readRDS(paste0("./_test_error/S_L_lambda_",i,".rds"))}
non_otu_predmatrix<-matrix(NA,327,2)
for (i in 1:folds) {
non_otu_predmatrix[i,]<- exp(readRDS(paste0(result_path,"fold_",i,'/log_prior_test.rds')))
}
saveRDS(non_otu_predmatrix,paste0(result_path,"non_otu_predmatrix.rds"))
non_otu_predmatrix<-readRDS(paste0(result_path,"non_otu_predmatrix.rds"))

nlambda <- length (lambda)
n <- one_dataset$n
pred_matrix <- array (0, dim = c(n,2,nlambda))
for (i in 1:n) {
for (j in 1:nlambda)
pred_matrix[i,,j] <- readRDS(paste0(result_path,"fold_",i,"/S_L/lambda_",j,"Pr_of_PD_test.rds"))}
saveRDS(pred_matrix, file = paste0(result_path, "all_pred_matrix.rds"))
pred_matrix<-readRDS(paste0(result_path, "all_pred_matrix.rds"))

## find predictive metrics against lambda
```

```
y_true <- as.numeric(one_dataset$dataset$parkinson)
non_otu_er<-mean(y_true!=max.col(non_otu_predmatrix))
amlp_s_l <- rep(0, nlambda)
er_s_l <- rep(0, nlambda)

for (i in 1:nlambda) {
eval <- evaluate_pred (pred_matrix[,,i],y_true)
#auc_lasso[i]<-roc(y_true,pred_matrix[,2,i])$auc
#auc_lasso[i] <- eval$auc
er_s_l[i] <- eval$er
amlp_s_l[i] <- eval$amlp
}
base<-matrix(0.5,nrow=327,ncol=2)
amlp_non_otu<-evaluate_pred (non_otu_predmatrix,y_true)$amlp
amlp_base<-evaluate_pred (base,y_true)$amlp

AUC<-vector()
AU_PR<-vector()
for (i in 1:nlambda){
AUC[i]<-roc(y_true,pred_matrix[,2,i])$auc
scores <- data.frame(pre=pred_matrix[,2,i], lab=y_true)
AU_PR[i] <- pr.curve(scores.class0=scores[scores$lab=="2",]$pre,
scores.class1=scores[scores$lab=="1",]$pre,
curve=T)$auc.integral}
scores_nonotu <- data.frame(pre=exp(non_otu_predmatrix[,2]), lab=y_true)
AU_PR_non_otu <- pr.curve(scores.class0=scores_nonotu[scores_nonotu$lab=="2",]$pre,
scores.class1=scores_nonotu[scores_nonotu$lab=="1",]$pre,
curve=T)$auc.integral

AUC_non_otu<-roc(y_true,non_otu_predmatrix[,2])$auc

which.min(er_s_l)
which.min(amlp_s_l)
best_lam.n<-which.min(er_s_l)

pred <- prediction(pred_matrix[,2,best_lam.n], y_true)
non_otu_pred<-prediction(non_otu_predmatrix[,2],y_true)
perf <- performance(pred,"tpr","fpr")
perf1 <- performance(pred, "prec", "rec")
non_otu_perf<-performance(non_otu_pred, "prec", "rec")

##########AUC/AUPR#########
pdf(file=paste0("./__final_plots/L1SVM+logistic/AUC_and_AUC-PR.pdf"),width=6,height = 6)
par(mar=c(4,4,2,1))
par(mar=c(3,4,2,4))
plot(log(lambda), AUC,col="1",type="l",lwd=2,xlab=" ",ylab=" ",lty=1)
par(new=TRUE)
plot(log(lambda), AU_PR,col="2",type="l",lwd=2,axes=F,ylab=" ",xlab="",lty=4)
axis(side=4,col.axis="black")
mtext(side=1, line=2, expression(log(lambda)), cex=1)
mtext(side=4, line=2, "AUPRC", cex=1.2)
mtext(side=2, line=2, "AUC", cex=1.2)
#legend('topleft', c('AUPRC', 'AUC'),col=c('red', 'black'),bty = "o",cex=0.8,lty=1)
abline(h=seq(0.4,0.88,by=0.02),v=seq(-45,-1.5,by=0.25),col="grey",lty=3)
```

```
legend('topleft', c('AUPRC', 'AUC'),col=c('red', 'black'),lty=c(4,1),bty = "o",cex=0.8,bg="white")
dev.off()

#########Error raete#######
pdf(file=paste0("./__final_plots/L1SVM+logistic/L1SVM+logistic_Error_Rate.pdf"),width=6,height = 6)
par(mar=c(3,4,1,4))
plot(log(lambda), er_s_l,col="blue",type="l",lwd=2,ylim=c(0.275,0.40),xlab="", ylab="")
abline(h=non_otu_er,col="darkgreen",lty=2,lwd=2)
abline(h=130/327,col="red",lty=3,lwd=2.5)
axis(side = 4, at=seq(0,130/327,by=130/327/10), labels=seq(1,0,by=-0.1))
mtext(side=4, line=2, expression(R^2), cex=1.5,las=2)
mtext(side=2, line=2, "ER", cex=1.5,las=2)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0.2,0.42,by=0.005),v=seq(-4.5,-1,by=0.25),col="grey",lty=3)
legend('bottomleft', c('OTU+age+sex', 'age+sex','baseline'),col  =c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

#########amlp#######
pdf(file=paste0("./__final_plots/L1SVM+logistic/L1SVM+logistic_amlp.pdf"),width=6,height = 6)
par(mar=c(3,4,1,4))
plot(log(lambda), amlp_s_l,col="blue",type="l",lwd=2,xlab="", ylab="")
abline(h=amlp_non_otu,col="darkgreen",lty=2,lwd=2)
abline(h=amlp_base,col="red",lty=3,lwd=2.5)
mtext(side=2, line=2, "amlp", cex=1.5,las=3)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0.54,0.8,by=0.01),v=seq(-4.5,-1.5,by=0.25),col="grey",lty=3)
legend('bottomleft', c('OTU+age+sex', 'age+sex','baseline'),col  =c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

### Probability of Parkinson######
pdf(file=paste0("./__final_plots/L1SVM+logistic/Probability of Parkinson.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
par(mfrow=c(1,1))
PDorder<-order(y_true)
fr<-data.frame(p=pred_matrix[sample_order,,best_lam.n], l=y_true)
plot(pred_matrix[sample_order,2,best_lam.n],c(1:327),pch=2*y_true[sample_order]-1,
col=-2*y_true[PDorder]+6,xlab="Probability of Parkinson",ylab="Subject ID")
abline(v=0.5,lty=3,lwd=1)
dev.off()

#########ROC#########
pdf(file=paste0("./__final_plots/L1SVM+logistic/roc.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(roc(y_true,pred_matrix[,2,best_lam.n]),col="blue",main = "",lty=1,lwd=2)
lines(roc(y_true,exp(non_otu_predmatrix[,2])),col='darkgreen',lty=2,lwd=2)
legend('bottomright', c('OTU+age+sex, AUC=0.78', 'age+sex, AUC=0.66'),col  =c('blue', 'darkgreen'),
bty = "o",cex=0.8,lty=c(1,2),lwd=2)
dev.off()
######### P_R curve #########
pdf(file=paste0("./__final_plots/L1SVM+logistic/PRC.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(perf1,col="blue",main  = "",ylim=c(0.6,1),lty=1,lwd=2)
```

```
lines(non_otu_perf@x.values[[1]],non_otu_perf@y.values[[1]],col="darkgreen",lty=2,lwd=2)
abline(h=197/327,lty=3,col="red",lwd=2)
legend('topright', c('OTU+age+sex, AUPR=0.83', 'age+sex, AUPR=0.72, ','baseline'),
col  =c('blue', 'darkgreen','red'),bty = "o",cex=0.8,lty=c(1,2,3),lwd=2)
dev.off()

#############beta################
beta<-matrix(NA,ncol=382,nrow=327)
for(i in 1:n){
beta[i,]<-readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds"))
}
colnames(beta)<-names(readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds")))
median_absolute<-apply(beta, 2, function(x) median(abs(x)))
medians<-apply(beta, 2, function(x) median(x))
non_zero_n<-sum(medians!=0)
non_zero_index<-which(medians!=0)
ab_me_order.n<-order(median_absolute,decreasing=TRUE)
ab_me_order<-colnames(beta)[ab_me_order.n]
sign_of_top_beta<-sign(medians)[ab_me_order.n[1:non_zero_n]]
which(sign_of_top_beta==-1)
top_number<-2
rank<-ab_me_order.n
ordered_exprSet <- exprSet[sample_order, c("parkinson",ab_me_order)]
top_genera<-readRDS("./_top_genera_names.rds")
b_data<-data.frame(beta[,rank[1:top_number]],id=rownames(dataset))
beta_data<-melt(b_data,id.vars = c('id') )

###top 20
pdf(file=paste0("./__final_plots/L1SVM+logistic/top 20 beta_absolute.pdf"),width=18,height = 10)
boxplot(abs(b_data[1:2]),col = -1*sign_of_top_beta+3,outcol=-1*sign_of_top_beta+3,
names=rep("",time=2),par(mar=c(5,5,1,1)),xaxt='n')
labels <- top_genera[c(12,2)]
text(seq(1.3,2.7,by=1), par("usr")[3] - 0.001,  adj = 1, labels =labels, xpd = TRUE,cex=2)
mtext(side=2, line=3, "Absolute Values of Coefficients", cex=2.2)
abline (h=seq(0.055,0.080,0.005), col = "lightgrey", lty=3,lwd=1)
legend('topright', c('-positive', '-negative'),fill =c('red', 'blue'),bty = "o",cex=1.8,bg="white")
dev.off()

#############################
ordered_exprSet_add<-matrix(NA,nrow=328,ncol=382)
for (i in 1:382){ordered_exprSet_add[,i]<-c(ordered_exprSet[,i+1],NA)}
for (i in 1:382){ordered_exprSet_add[,i]<-as.numeric(ordered_exprSet_add[,i])}
parkinson_add<-c(as.numeric(ordered_exprSet[,1]),3)
parkinson_add<-as.factor(parkinson_add)
fit<-data.frame(parkinson_add,ordered_exprSet_add)
colnames(fit)<-colnames(ordered_exprSet)
oadd_data1<-melt(fit[,1:(top_number+1)],id.vars = c("parkinson") )

######### boxplot pf normalized OTU ############
pdf(file=paste0("./__final_plots/L1SVM+logistic/boxplot of top 20 OTU.pdf"),width=18,height = 10)
boxplot(value~parkinson+variable, data=oadd_data1,col=rep(c(4,2,1),2),names=rep("",time=6),
par(mar=c(5,5,1,1)),outcol=rep(c(4,2,1),2),xaxt="n")
labels <- top_genera[c(12,2)]
text(seq(2.2,7,by=3.2), par("usr")[3] - 0.2,  adj = 1, labels =labels, xpd = TRUE,cex=2)
```

```
mtext(side=2, line=3, "Standardized Log Relative Abundances", cex=2.2)
abline (h=seq(-2,3,1), col = "lightgrey", lty=3,lwd=1)
legend('topright', c('-positive', '-negative'),fill =c('red', 'blue'),bty = "o",cex=1.8,bg="white")
dev.off()
```

C.5.4 Results of SVM with Elastic-net

```
dataset_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/dataset_real2_loocv.rds"
result_path<-"~/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/"

setwd(result_path)
one_dataset<-readRDS(dataset_path)
dataset <- one_dataset$dataset
otu_names <- one_dataset$otu_names
otu.num<-one_dataset$otu.num
genus_names<-one_dataset$genus_names
var_interested_name <- "parkinson"
nl <- nlevels(dataset[1, var_interested_name])
n<-one_dataset$n
folds<-one_dataset$folds
folds.index<-one_dataset$folds.index

library(glmnet)
library(pROC)
library(ROCR)
library(PRROC)
library(reshape2)
#library (HTLR)
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")
trans<-apply(dataset[,1:382],1,function(x) log(x+1)-log(sum(x)+length(x) ))
trans1<-t(trans)
dataset[,1:382]<-trans1

nsample <- n
Y_name <- "parkinson"
X_name <- otu_names
X_cov<-c("age","sex")
fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_name), collapse = "+")))
X <- model.matrix(fit_formula,data=dataset)[,-1]
Y <- dataset$parkinson

lambda <- readRDS("/home/mac744/ParkinsonBayes/dataset_real2_svmela_loocv_logbyT/_lambda.rds")
nlambda<-length(lambda)
X_c <- sweep(X, 2, colMeans(X), "-")
X_n <- sweep(X_c,2, apply(X_c,2,sd), "/")
exprSet=data.frame(X_n[,-c(1:2)],parkinson=Y)
sample_order<-order(Y)

test_error<-rep(0,times=nlambda)
for(i in 1:nlambda){
test_error[i]<-readRDS(paste0("./_test_error/S_L_lambda_",i,".rds"))}

non_otu_predmatrix<-matrix(NA,327,2)
```

```
for (i in 1:folds) {
non_otu_predmatrix[i,]<- exp(readRDS(paste0(result_path,"fold_",i,'/log_prior_test.rds')))
}
saveRDS(non_otu_predmatrix,paste0(result_path,"non_otu_predmatrix.rds"))
non_otu_predmatrix<-readRDS(paste0(result_path,"non_otu_predmatrix.rds"))

nlambda <- length (lambda)
n <- one_dataset$n
pred_matrix <- array (0, dim = c(n,2,nlambda))
for (i in 1:n) {
for (j in 1:nlambda)
pred_matrix[i,,j] <- readRDS(paste0(result_path,"fold_",i,"/S_L/lambda_",j,"Pr_of_PD_test.rds"))}
saveRDS(pred_matrix, file = paste0(result_path, "all_pred_matrix.rds"))
pred_matrix<-readRDS(paste0(result_path, "all_pred_matrix.rds"))

## find predictive metrics against lambda
y_true <- as.numeric(one_dataset$dataset$parkinson)
non_otu_er<-mean(y_true!=max.col(non_otu_predmatrix))
amlp_s_l <- rep(0, nlambda)
er_s_l <- rep(0, nlambda)

for (i in 1:nlambda) {
eval <- evaluate_pred (pred_matrix[,,i],y_true)
#auc_lasso[i]<-roc(y_true,pred_matrix[,2,i])$auc
#auc_lasso[i] <- eval$auc
er_s_l[i] <- eval$er
amlp_s_l[i] <- eval$amlp
}
base<-matrix(0.5,nrow=327,ncol=2)
amlp_non_otu<-evaluate_pred (non_otu_predmatrix,y_true)$amlp
amlp_base<-evaluate_pred (base,y_true)$amlp
AUC<-vector()
AU_PR<-vector()
for (i in 1:nlambda){
AUC[i]<-roc(y_true,pred_matrix[,2,i])$auc

scores <- data.frame(pre=pred_matrix[,2,i], lab=y_true)
AU_PR[i] <- pr.curve(scores.class0=scores[scores$lab=="2",]$pre,
scores.class1=scores[scores$lab=="1",]$pre,
curve=T)$auc.integral}
scores_nonotu <- data.frame(pre=exp(non_otu_predmatrix[,2]), lab=y_true)
AU_PR_non_otu <- pr.curve(scores.class0=scores_nonotu[scores_nonotu$lab=="2",]$pre,
scores.class1=scores_nonotu[scores_nonotu$lab=="1",]$pre,
curve=T)$auc.integral
AUC_non_otu<-roc(y_true,non_otu_predmatrix[,2])$auc

which.min(er_s_l)
which.min(amlp_s_l)
best_lam.n<-which.min(er_s_l)

pred <- prediction(pred_matrix[,2,best_lam.n], y_true)
non_otu_pred<-prediction(non_otu_predmatrix[,2],y_true)
perf <- performance(pred,"tpr","fpr")
perf1 <- performance(pred, "prec", "rec")
```

```
non_otu_perf<-performance(non_otu_pred, "prec", "rec")

##########AUC/AUPR#########
pdf(file=paste0("./__final_plots/elasticSVM+logistic/AUC_and_AUC-PR.pdf"),width=6,height = 6)
par(mar=c(4,4,2,1))
par(mar=c(3,4,2,4))
plot(log(lambda), AUC,col="1",type="l",lwd=2,xlab=" ",ylab=" ",lty=1)
par(new=TRUE)
plot(log(lambda), AU_PR,col="2",type="l",lwd=2,axes=F,ylab=" ",xlab="",lty=4)
axis(side=4,col.axis="black")
mtext(side=1, line=2, expression(log(lambda)), cex=1)
mtext(side=4, line=2, "AUPRC", cex=1.2)
mtext(side=2, line=2, "AUC", cex=1.2)
#legend('topleft', c('AUPRC', 'AUC'),col=c('red', 'black'),bty = "o",cex=0.8,lty=1)
abline(h=seq(0.4,0.88,by=0.02),v=seq(-4,0,by=0.25),col="grey",lty=3)
legend('topleft', c('AUPRC', 'AUC'),col=c('red', 'black'),lty=c(4,1),bty = "o",cex=0.8,bg='white')
###
dev.off()

#########Error raete#######
pdf(file=paste0("./__final_plots/elasticSVM+logistic/elastic_SVM+logistic_Error_Rate.pdf"),
width=6,height = 6)
par(mar=c(3,4,1,4))
plot(log(lambda), er_s_l,col="blue",type="l",lwd=2,ylim=c(0.275,0.55),xlab="", ylab="")
abline(h=non_otu_er,col="darkgreen",lty=2,lwd=2)
abline(h=130/327,col="red",lty=3,lwd=2.5)
axis(side = 4, at=seq(0,130/327,by=130/327/10), labels=seq(1,0,by=-0.1))
mtext(side=4, line=2, expression(R^2), cex=1.5,las=2)
mtext(side=2, line=2, "ER", cex=1.5,las=2)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0.2,0.55,by=0.01),v=seq(-4,0,by=0.25),col="grey",lty=3,lwd=1)
legend('bottomleft', c('OTU+age+sex', 'age+sex','baseline'),col  =c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

#########amlp#######
pdf(file=paste0("./__final_plots/elasticSVM+logistic/elastic_SVM+logistic_amlp.pdf"),width=6,
height = 6)
par(mar=c(3,4,1,4))
plot(log(lambda), amlp_s_l,col="blue",type="l",lwd=2,xlab="", ylab="")
abline(h=amlp_non_otu,col="darkgreen",lty=2,lwd=2)
abline(h=amlp_base,col="red",lty=3,lwd=2.5)
mtext(side=2, line=2, "amlp", cex=1.5,las=3)
mtext(side=1, line=2, expression(log(lambda)), cex=1)
abline(h=seq(0.55,0.8,by=0.01),v=seq(-4.5,0,by=0.25),col="grey",lty=3)
legend('bottomleft', c('OTU+age+sex', 'age+sex','baseline'),col  =c('blue', 'darkgreen','red'),
cex=0.8,lty=c(1,2,3),lwd=c(2,2,2.5),bg="white")
dev.off()

### Probability of Parkinson######
pdf(file=paste0("./__final_plots/elasticSVM+logistic/Probability of Parkinson.pdf"),
width=6,height = 6)
par(mar=c(4,4,2,2))
par(mfrow=c(1,1))
```

```
PDorder<-order(y_true)
fr<-data.frame(p=pred_matrix[sample_order,,best_lam.n], l=y_true)
plot(pred_matrix[sample_order,2,best_lam.n],c(1:327),pch=2*y_true[sample_order]-1,
col=-2*y_true[PDorder]+6,xlab="Probability of Parkinson",ylab="Subject ID")
abline(v=0.5,lty=3,lwd=1)
dev.off()

########ROC#########
pdf(file=paste0("./__final_plots/elasticSVM+logistic/roc.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(roc(y_true,pred_matrix[,2,best_lam.n]),col="blue",main = "",lty=1,lwd=2)
lines(roc(y_true,exp(non_otu_predmatrix[,2])),col='darkgreen',lty=2,lwd=2)
legend('bottomright', c(paste0('OTU+age+sex, AUC=',round(AUC[best_lam.n],2)
), paste0('age+sex, AUC=',round(AUC_non_otu,2))),col  =c('blue', 'darkgreen'),bty = "o",
cex=0.8,lty=c(1,2),lwd=2)
dev.off()
######### P_R curve #########
pdf(file=paste0("./__final_plots/elasticSVM+logistic/PRC.pdf"),width=6,height = 6)
par(mar=c(4,4,2,2))
plot(perf1,col="blue",main  = "",ylim=c(0.6,1),lty=1,lwd=2)
lines(non_otu_perf@x.values[[1]],non_otu_perf@y.values[[1]],col="darkgreen",lty=2,lwd=2)
abline(h=197/327,lty=3,col="red",lwd=2)
legend('topright', c(paste0('OTU+age+sex, AUPR=',round(AU_PR[best_lam.n],2)),
paste0('age+sex, AUPR=',round(AU_PR_non_otu,2) ),'baseline'),col  =c('blue', 'darkgreen','red'),
bty = "o",cex=0.8,lty=c(1,2,3),lwd=2)
dev.off()

#############beta###############
beta<-matrix(NA,ncol=382,nrow=327)
for(i in 1:n){
beta[i,]<-readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds"))
}
colnames(beta)<-names(readRDS(paste0("./fold_",i,"/SVM/lambda_",best_lam.n,"beta.rds")))
median_absolute<-apply(beta, 2, function(x) median(abs(x)))
medians<-apply(beta, 2, function(x) median(x))
non_zero_n<-sum(medians!=0)
non_zero_index<-which(medians!=0)
ab_me_order.n<-order(median_absolute,decreasing=TRUE)
ab_me_order<-colnames(beta)[ab_me_order.n]
sign_of_top_beta<-sign(medians)[ab_me_order.n[1:non_zero_n]]
which(sign_of_top_beta==-1)
top_number<-25
rank<-ab_me_order.n
ordered_exprSet <- exprSet[sample_order, c("parkinson",ab_me_order)]
top_genera<-readRDS("./_top_genera_names.rds")
b_data<-data.frame(beta[,rank[1:top_number]],id=rownames(dataset))
beta_data<-melt(b_data,id.vars = c('id') )

#top 20 beta
pdf(file=paste0("./__final_plots/elasticSVM+logistic/top 20 beta_absolute.pdf"),width=18,height = 10)
boxplot(abs(b_data[1:20]),col = -1*sign_of_top_beta+3,outcol=-1*sign_of_top_beta+3,
names=rep("",time=20),par(mar=c(7.8,11.5,1,1)),xaxt='n')
labels <- top_genera[1:20]
text(seq(1,20,by=1), par("usr")[3] - 0.0018, srt = 19, adj = 1, labels =labels, xpd = TRUE,cex=1.5)
```

```
mtext(side=2, line=3, "Absolute Values of Coefficients", cex=1.8)
abline (v=seq(1.5,20,by = 1), col = "darkgrey", lty=2,lwd=4)
abline (h=seq(0.02,0.14,by = 0.01), col = "lightgrey", lty=2,lwd=1)
legend('topright', c('-positive', '-negative'),fill =c('red', 'blue'),bty = "o",cex=1.4,bg="white")
dev.off()

#############################
ordered_exprSet_add<-matrix(NA,nrow=328,ncol=382)
for (i in 1:382){ordered_exprSet_add[,i]<-c(ordered_exprSet[,i+1],NA)}
for (i in 1:382){ordered_exprSet_add[,i]<-as.numeric(ordered_exprSet_add[,i])}
parkinson_add<-c(as.numeric(ordered_exprSet[,1]),3)
parkinson_add<-as.factor(parkinson_add)
fit<-data.frame(parkinson_add,ordered_exprSet_add)
colnames(fit)<-colnames(ordered_exprSet)
#ordered_exprSet_add[,1]<-factor(ordered_exprSet_add[,1])
oadd_data1<-melt(fit[,1:(top_number+1)],id.vars = c("parkinson") )
oadd_data2<-melt(fit[,1:(20+1)],id.vars = c("parkinson") )

######### boxplot pf normalized OTU ############
pdf(file=paste0("./__final_plots/elasticSVM+logistic/boxplot of top 20 OTU.pdf"),width=18,height = 10)
boxplot(value~parkinson+variable, data=oadd_data2,col=rep(c(4,2,1),20),
names=rep("",time=60),par(mar=c(7.8,12,1,1)),outcol=rep(c(4,2,1),20),xaxt="n")
labels <- top_genera[1:20]
text(seq(1.5,60,by=3), par("usr")[3] - 0.18, srt = 19, adj = 1, labels =labels, xpd = TRUE,cex=1.5)
mtext(side=2, line=3, "Standardlized Log Relative Abundances", cex=1.8)
abline (v=seq(3,57,by = 3), col = "darkgrey", lty=2,lwd=4)
abline (h=seq(-4,6,by = 1), col = "lightgrey", lty=2,lwd=1)
legend('topright', c('-parkinson', '-control'),fill =c('red', 'blue'),bty = "o",cex=1.4,bg='white')
dev.off()
```