Practical Perspectives on Variable Selection for Binary Longitudinal Data

Kefan Ping, Joyce Lin, Feng Ding, Yanjun Wu, Shi Qu

March 15, 2024

Abstract

Accounting for the correlation among responses is crucial for statistical analysis and reliable inferences in multivariate response data. In areas such as biomedical studies, binary longitudinal data frequently arises. Focusing on this, we review variable selection methods based on GEE, GLMM, Zero-Inflated Negative Binomial and machine learning methods such as random forest and neural networks from a practicality point of view. The implementations as well as computational challenges of these methods in software R are addressed. We study and evaluate our methods first by simulations, and then illustrate them on the microbiome data set.

Keywords— Feature Selection, Binary Longitudinal Data, GEE, GLMM, Zero-Inflated Negative Binomial, Machine Learning

1 Introduction

In areas like clinical trials, observational studies, and biomedical studies, it's common to gather measurements of a response variable from the same subject over time, constituting longitudinal data. In such data set, responses from a single subject are usually correlated, and incorporating this correlation into model construction is often necessary for reliable statistical inferences. Binary longitudinal data frequently arises in practice, where the response variable often represents whether a subject is infected with a certain disease. It is then of interest to

gain insights to the way how multiple drivers cooperate to cause a specific disease. Since many such mechanisms are still not well understood but it is often plausible to assume that they are multifactorial. Therefore, empirical data collection and multivariable analysis via statistical methods are important contributors to knowledge generation [1]. However, it is not known beforehand which covariates should be included in a model, and often we are confronted with too many candidate covariates to include them all in the model.

Variable selection is therefore one of the most important steps in model development. Especially when there are large numbers of candidate predictors, a good variable selection method should effectively pick out the important variables and keep the computation efficient [2]. However, responses from the same subject are often correlated in longitudinal data, which in turn renders the likelihood-based parameter estimation computationally difficult due to the complexity of the joint distribution of response variables. Such burden receives little attention in the current literature. It is often unclear what is a practical method that a researcher could employ for the analysis. Trial and error with multiple methods is often time consuming, given one not only need to understand the model development, but also its corresponding software implementation. To make matters worse, it's often hard to compare the performance of these methods without a clearly defined metric.

In this paper, we review variable-selection methods based on GEE, GLMM, Zero-Inflated Negative Binomial and other related machine learning methods from a practicality point of view. The paper is organized as follows. In Section 2, we review both the classical and newly emerged machine learning methods that can be applied to binary longitudinal data. In addition to their mathematical development, their corresponding implementations in software R is also introduced and discussed. We also addressed and quantified the computationally difficulty that comes with the correlated structure of the dataset. In Section 3, we conducted a simulation study for the feasible methods introduced in the previous section. In Section 4, we applied the methods to the GvHD microbiome data set. Lastly, Section 5 summarized the results and concluded the study.

2 Methods

2.1 GEE based Methods

2.1.1 QIC_u forward on GEE

The Generalized Estimating Equations (GEE) [3] approach is also widely used in the modeling of longitudinal data. It is a marginal model assuming that we have the specification of the marginal mean and variance structure: $E[Y_{ij}] = \mu_{ij}$, $Var(Y_{ij}) = \phi a_{ij}^{-1} V(\mu_{ij})$, and a marginal mean model: $g(\mu_{ij}) = X_{ij}^T \beta$, where Y_{ij} and X_{ij} is the response and covariates vector for the j th observation within the i th subject, and β is the parameter vector that we are very interested in. Then, the Generalized Estimating Equations for estimating β is given by:

$$S(\beta) = \sum_{i=1}^{m} D_i^T W_i^{-1} (Y_i - \mu_i) = 0$$

where $D_i = \frac{\partial \mu_i}{\partial \beta}$ is a $n_i \times p$ matrix, W_i is a $n_i \times n_i$ working covariance matrix, $Y_i - \mu_i$ is a $n_i \times 1$ vector, for subject i, $i = 1, \cdots, m$, and the subject i has n_i correlated observations. For the working covariance matrix W_i, we can specify it as $W_i = V_i^{\frac{1}{2}} R_i(\alpha) V_i^{\frac{1}{2}}$, where $V_i = diag[\phi a_{ij}^{-1} V(\mu_{ij})]$ is the marginal variance of Y_i, and $R_i(\alpha)$ is the working correlation matrix, which can be independence, exchangeable, unstructured and so on.

There are several approaches to perform variable selection based on the GEE model. Pan [4] modified the Akaike's Information Criterion (AIC) in the GEE model and developed Quasi-likelihood under the independence model criterion (QIC) and its simplified approximation QIC_u:

$$QIC(R) = -2Q(\hat{\beta}(R); I) + 2trace(\hat{\Omega_I}^{-1} \hat{V_R})$$

$$QIC_u = -2Q(\hat{\beta}(R); I) + 2p$$

where R is the working correlation structure, I is the independence correlation structure used to compute the Quasi-likelihood; $\hat{\Omega_I}^{-1}$ and $\hat{V_R}$ are variance estimator obtained under correlation structure I and R, respectively. The QIC_u can be used to perform variable selection under the same working correlation structure, and models with smaller QIC_u are better. As a result, we can perform the classical forward/backward/both direction selection algorithms based on the criterion of QIC_u.

2.1.2 Penalized GEE

The penalized GEE [5] estimate the coefficients β by adding a penalization term to the original generalized estimating equations:

$$S^P(\beta) = \sum_{i=1}^{m} D_i^T W_i^{-1}(Y_i - \mu_i) - \frac{\partial P(\beta)}{\partial \beta}$$

where $P(\beta) = \lambda \sum |\beta_j|^\gamma$ is the penalty. It becomes Lasso when $\gamma = 1$ and Ridge when $\gamma = 2$. The penalized generalized estimating equations with Lasso is especially interesting to us, as the estimation of parameters can shrink to 0. As a result, this can be very useful when performing variable selection. The R package `LassoGEE` is capable of fitting generalized esimating equations with L1 regularization in terms of high dimensional longitudinal data by I-CGD algorithm and re-weighted least square algorithm. After cross validation procedure to choose a appropriate λ, we can fit the penalized GEE and perform variable selection accroding to the estimation of coefficients.

2.2 GLMM based Methods

The Generalized Linear Mixed Model (GLMM) [6] is a versatile statistical framework that extends the capabilities of the Generalized Linear Model (GLM) by incorporating random effects to account for correlation and non-independence in the data. GLMMs are particularly useful when analyzing nested or clustered data, longitudinal studies, or any data structure where observations are not independent.

Let Y_{ij} denote j th observation within the i th subject, where $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Furthermore, let $\boldsymbol{X}_{ij} \in \mathbb{R}^p$ be the covariate associated with fixed effects and $\boldsymbol{Z}_{ij} \in \mathbb{R}^q$ be the covariate associated with random effects. It is assumed that the observations Y_{ij} are conditionally independent with means $\mu_{ij} = E[Y_{ij}|\mathbf{b}_i, \boldsymbol{X}_{ij}, \boldsymbol{Z}_{ij}]$ and variance $Var(Y_{ij}|\mathbf{b}_i) = \phi\nu(\mu_{ij})$, where $\mathbf{b}_i \in \mathbb{R}^q$ is subject-specific random effect, ν is a known variance function, and ϕ is a scale parameter. The GLMM could be written as:

$$g(\mu_{ij}^{\mathbf{b}_i}) = \boldsymbol{X}_{ij}^\top \boldsymbol{\beta} + \boldsymbol{Z}_{ij}^\top \mathbf{b}_i,$$

where $\mathbf{b}_i \sim N(0, \mathbf{D}_0(\theta))$ with $q \times q$ covariance matrix $\mathbf{D}_0$, and $\boldsymbol{\beta} \in \mathbb{R}^p$ is the vector of fixed

effects.

The methods introduced below all requires one to fit a GLMM model first. The most popular way to do so is by the `glmer` function from the `lme4` package in R. This function allows for the fitting of GLMMs with various distributional assumptions for the response variable and incorporates both fixed and random effects. Additionally, in our GLMM analysis, we utilize the `glmerControl` function from the `lme4` package to specify the optimization method for fitting the models. Specifically, we use the optimizer `optimx` package with `nlminb` method [7]. The `optimx` optimizer is a flexible optimization framework from `optimx` that provides a unified interface to various optimization algorithms, allowing for efficient optimization of complex objective functions. The `nlminb` method is one of the optimization algorithms available within the `optimx` framework, which is particularly well-suited for constrained optimization problems with bounded parameters. This method is commonly used in statistical modeling to find the maximum likelihood estimates of the parameters in GLMMs while satisfying any constraints imposed on the parameter space.

2.2.1 Adjusted p-value on GLMM

The most straight forward feature selection method is perhaps simply choose a set of covariates which all have p-values smaller than some pre-set threshold. However, as Benjamin and Hochberg pointed out, the unguarded use of single-inference procedures results in a greatly increased false positive (significance) rate [8]. This is particularly concerning for models that contains large amount of covariates. Therefore, for more reliable inferences, one should consider multiple hypotheses testing procedures. Many such procedures have been developed over the years, we will mainly focus on controlling the false discovery rate and Holm's method. For discussions on the development of these methods, see [8] and [9].

Adjusted p-values in the context of GLMMs provide a means to account for multiple hypothesis testing while considering both fixed and random effects. It helps one to identify the significant fixed effects while taking correlation into account. To obtain the adjusted p-values, simply pass the p-values returned by `glmer` into the R function `p.adjust`.

2.2.2 AIC forward on GLMM

Initially introduced by Hirotogu Akaike in 1973 [10], the Akaike Information Criterion (AIC) has emerged as a pivotal tool for model selection in statistical analysis. The AIC represents a significant advancement in the realm of maximum likelihood estimation, providing a comprehensive framework for both parameter estimation and model selection. By incorporating measures of model complexity and goodness of fit, the AIC enables researchers to compare competing models and identify the most appropriate one for their data. Notably, for GLMM, the likelihood function is defined as:

$$\exp(\ell(\beta,\theta)) \propto |\mathbf{D}| \int \exp\left(\sum_{i=1}^{n} \ell_i(Y_i|\mathbf{b};\beta) - \frac{1}{2}\mathbf{b}^\top \mathbf{D}^{-1}\mathbf{b}\right) d\mathbf{b}$$

The Akaike Information Criterion (AIC) is computed as:

$$AIC(\mathcal{M}) = -2\ell(\mathcal{M}) + 2\dim(\mathcal{M}),$$

where $\mathcal{M}$ represents the GLMM, and $\ell(\mathcal{M})$ denotes the maximum value of the log-likelihood of model $\mathcal{M}$. Models with smaller AIC values are preferred. Subsequently, a classical forward selection algorithm based on the AIC criterion can be performed to identify the most suitable model.

2.2.3 GLMM with Lasso regularization

Generalized Linear Mixed Models with Lasso regularization (GLMM Lasso) [11] is a variable selection approach for GLMM by L1-penalized estimation. The method integrates the advantages of GLMMs, which are capable of handling non-normal response variables and accounting for random effects, with the Lasso regularization technique, which promotes sparsity in the model coefficients by imposing a penalty on the absolute values of the coefficients. By introducing the penalty term $\lambda \sum_{i=1}^{p} |\beta_i|$ into the log-likelihood function, we obtain the penalized log-likelihood:

$$\ell^{\text{pen}}(\beta,\theta) = \ell(\beta,\theta) - \lambda \sum_{i=1}^{p} |\beta_i|,$$

where $\lambda_i = 0$ is chosen for unpenalized parameters. This penalty term is weighted by a tuning parameter, lambda, which controls the degree of regularization applied to the model. As lambda

increases, more coefficients are shrunk towards zero, leading to a sparser model with fewer predictors. Once the optimal lambda value is determined, the GLMM Lasso model is fit to the entire dataset using this lambda value. The resulting model provides estimates of the fixed effects parameters, indicating the strength and direction of the relationships between the predictors and the response variable, while simultaneously performing variable selection by shrinking some coefficients to zero.

The `glmmLasso` package in R facilitates the implementation of the GLMM Lasso method. This package employs a gradient ascent algorithm specifically designed for generalized linear mixed models, incorporating variable selection through L1-penalized estimation. In the final re-estimation step, a model is fitted containing only the variables corresponding to the non-zero fixed effects using simple Fisher scoring.

2.2.4 Boosting GLMM

Boosting Generalized Linear Mixed Models (Boosting GLMM) [12] is a statistical approach aimed at enhancing the predictive performance of GLMMs through iterative model fitting and ensemble learning techniques. The method combines the flexibility of GLMMs in handling non-normal response variables and accounting for random effects with the boosting algorithm's ability to improve predictive accuracy by sequentially fitting models to the residuals of the previous model.

The Boosting GLMM algorithm begins by initializing with the fitting of an initial GLMM to the dataset. Sequentially, subsequent models are iteratively fitted to the residuals of the preceding model. Each of these models aims to capture the unexplained variance in the response variable left by its predecessors. After fitting each model, predictions are obtained and combined through weighted averaging, with the weights assigned based on the performance of each model on the training data. This iterative process continues for a predetermined number of iterations or until a stopping criterion is met, such as reaching a maximum number of models or achieving satisfactory performance on a validation dataset. Through this iterative refinement, Boosting GLMM effectively enhances the predictive performance of the model, iteratively improving its ability to capture complex relationships within the data.

In R, the `bGLMM` package provides a framework for fitting Generalized Linear Mixed Models (GLMMs) with variable selection using information criteria such as Akaike Information

Criterion (AIC) or Bayesian Information Criterion (BIC). This package enables users to automatically select the most relevant predictors while considering random effects in the data structure. However, the efficiency of this function is time-consuming, as depicted in Figure 1. On a 16-core machine with approximately 100 percent CPU usage for all cores, fitting a model with 20 covariates takes nearly 20 minutes. Consequently, fitting models with multiple covariates may require a significant amount of time to complete the computation. In conclusion, while the `bGLMM` package offers powerful functionality for GLMM fitting and variable selection, users should be mindful of the computational resources and time required for larger models.

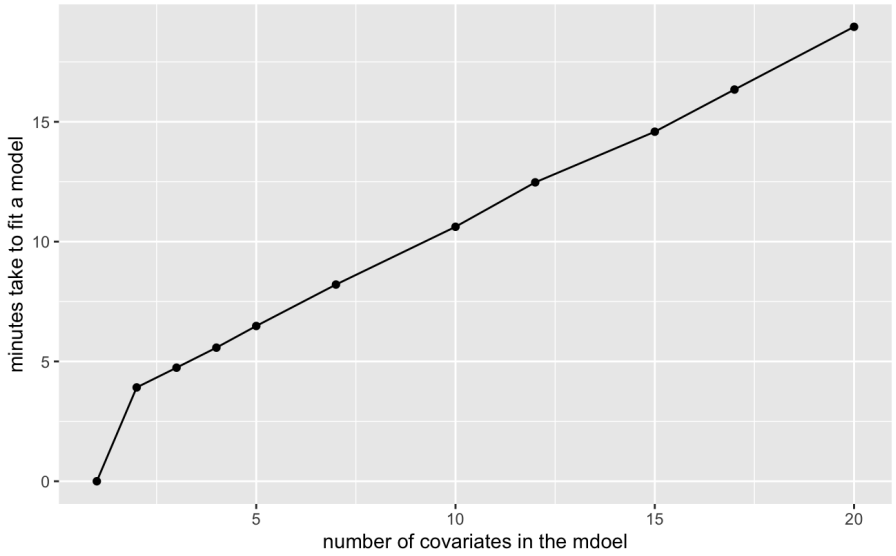

Figure 1: `bGLMM` fitting time with varying numbers of covariates.

2.2.5 GAMM

Generalized Additive Mixed Models (GAMMs) are an additive extension of GLMMs. It's essentially a GLMM with part of the linear predictor is specified in terms of smooth functions of covariates [13]. Extending the notations of GLMM defined above, a GAMM model can be written as

$$g(\mu_{ij}^{\mathbf{b}_i}) = \boldsymbol{X}_{ij}^{\top}\boldsymbol{\beta} + \sum_{k=1}^{L} \alpha_{(k)}(u_{ijk}) + \boldsymbol{Z}_{ij}^{\top}\mathbf{b}_i,$$

where the term $\sum_{k=1}^{L} \alpha_{(k)}(u_{ijk})$ is an additive term that depends on covariates $(u_{ij1}, \ldots, u_{ijL})$ and unspecified influence functions $\alpha_{(1)}, \ldots, \alpha_{(L)}$. It's common to take u_{ijk} equal to the corresponding x_{ijk}. There are many ways to estimate the unknown functions $\alpha_{(j)}$. While Lin and Zhang [13] considered natural cubic smoothing splines, in recent years, the method of regression

splines has become more popular. In the latter, the unknown functions $\alpha_{(j)}$ are approximated by linear combinations of basis functions. One example of such is B-spline basis of degree d, given by

$$\alpha_{(j)}(u) = \sum_{l=1}^{S} \alpha_l^{(j)} B_l^{(j)}(u; d)$$

where $B_l^{(j)}(u; d)$ denotes the l-th basis function for variable j. For a discussion of the details of the development of GAMMs, a good reference book can be found at [14].

There are two well-maintained packages in R for fitting GAMMs: the `gamm` function in the `mgcv` package, which implements the popular penalized quasi-likelihood (PQL) method, and the `gamm4` function within its corresponding package, which avoids PQL by utilizing the modular fitting functions provided in `lme4`. This methodological difference not only affects computational efficiency but also impacts the availability of variable selection procedures. Notably, `gamm` may perform poorly with binary data due to its reliance on PQL [15]. Furthermore, `gamm` returns the log-likelihood of the working model at convergence of the PQL iteration, rather than the likelihood of the fitted GAMM, thereby limiting the applicability of log-likelihood-based model selection approaches such as forward/backward selection based on AIC/BIC [15].

Given these limitations, `gamm4` appears to be a more attractive option for analysis. It is known for its numerical stability and superior performance, especially with binary and low mean count data [16]. However, an aspect often overlooked in the literature is the model fitting time. Consider a simulated data set that contains 100 subjects with 10 repeated observations each. Further suppose each observation produces a single response associated with at most 20 covariates. The below figure shows the number of additive covariates included in the model and the corresponding model fitting time in hours.

The computations were done on a 16-core machine with around 100 percent CPU usage for all cores throughout the process. Observe that the model fitting time increases exponentially as the number of additive covariates in the model increases, and the fitting time takes around 3.5 hours when all covariates are present and estimated as additive effect. This substantial computational expense presents great challenges for fully automatic model selection procedures. For example, the worst case for forward/backward selections can take more than a week to finish on a 16-core machine. Consequently, in practical applications, one may opt for feature selection methods based on adjusted p-values, as they only require little further computation resources.

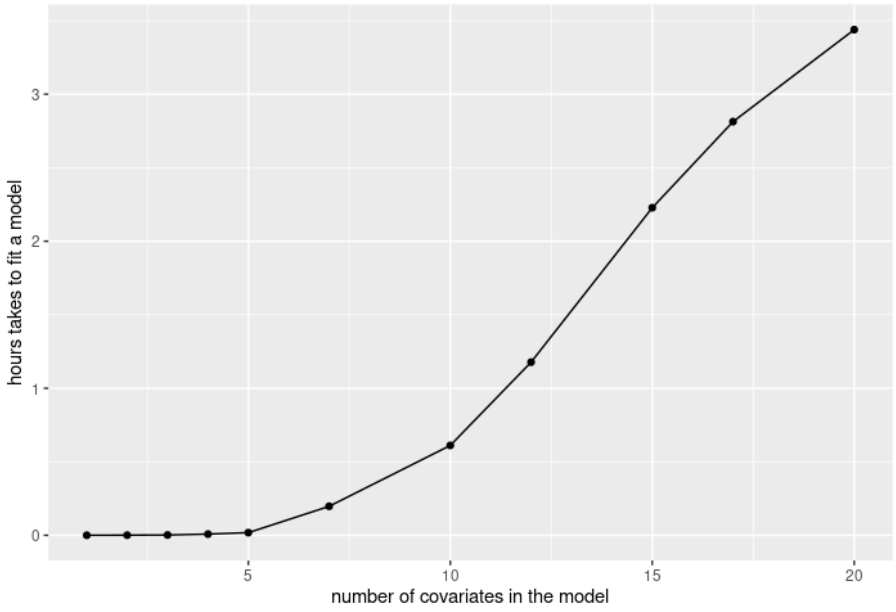

Figure 2: GAMM Model Fitting Time

It's worth noting that a shrinkage model selection method was developed for GAMs by Marra and Wood [17]. However, no similar approach was implemented for GAMMs to the best of our knowledge. In addition, the concept of likelihood-based boosting is extended to GAMMs by Groll and Tutz [18], which aimed for the selection of additive predictors.

2.3 Univariate Feature Selection Methods

Univariate feature selection method appears widely as a preliminary data preprocessing step in some high-dimensional dataset. The procedure examines the relationship between the response and covariates individually and independently, to determine which subset of the covariates has the strongest correlation with the target variable.

The zero-inflated distribution are commonly used to describe count data with more zeros than what would normally appears in a Poisson model or a negative binomial model. This distribution assumes the excess zeros originate from two different sources: a count distribution like Poisson or negative binomial that generates counts, and a zero distribution that only generates zero. The zero-inflated Poisson model is described as follows [19]:

$$P(Y = 0) = \Psi + (1 - \Psi)e^{-\lambda}$$

$$P(Y = y) = (1 - \Psi)\frac{\lambda^y e^{-\lambda}}{y!}, \qquad y = 1, 2, 3...$$

The zero-inflated negative binomial is described as [20]:

$$P(Y = 0) = \Psi + (1 - \Psi)p^r$$

$$P(Y = y) = (1 - \Psi)\frac{(y + r - 1)!}{y!(r - 1)!}p^r(1 - p)^y$$

where in the previous formula, Ψ is the probability of zero distribution, and others are respective parameters for Poisson and negative binomial distribution. When deciding between two models, one typically examines the association between mean and variance. If equi-dispersion is observed, a Poisson model is preferred; otherwise, one should stick to negative binomial model for over-dispersion.

The method here apply zero-inflated regression models directly to univariate feature selection on those sparse dataset with excess zero entries. It fits a zero-inflated regression model of selected features on the target variable, to calculate the statistical significance like p-value. Given the underlying true data distribution being zero-inflated, we can boost the model's performance compared to traditional Pearson-correlation-based or mutual-information-based feature selection.

One thing to note about the univariate feature selection method is that it makes the assumption that covariates are all independent because it does not take the inter-correlations between covariates into consideration.

2.4 Machine Learning and Sequence Learning based Methods

2.4.1 Random Forest based methods

Random Forest is a very powerful ensemble learning method developing from weak learners including classification and regression trees. We can utilize Random Forest, following the methodology outlined by [21]. The Random Forest method involves randomly sampling from the entire training set using Bootstrap to create multiple bootstrap training sets. For each of these bootstrap training sets, the method constructs a classification tree. During the fitting of these classification trees, this method randomly selects a subset of features to consider as potential splitting features at each node. This process significantly reduces the correlation between classification trees. Then the random forest can perform prediction based on the majority vote

of the constructed classification trees.

There are a variety of variable selection methods based on the standard random forest model. The feature importance measure can be obtained by performing random permutation of the values of the feature. If the average loss of accuracy of all the classification trees after random permutation of the values of the feature is very high, then it might indicate that this feature is very important.

The Boruta method [22] is a widely used variable selection method based on the random forest model. This method firstly adds copies of each feature (called shadow attributes), then it removes the correlations between the response and the shadow attributes by performing random permutation of the values of the shadow attributes. Next, a random forest is trained on the extended dataset, and the feature importance measures are computed. Afterwards, a two-sided test is performed for every feature against the maximum value of importances of all shadow attributes to decide whether the feature is important or not, and it also remove the unimportant features from the full data. The above procedure is performed repeatedly until every feature is decided or the algorithm has reached the stopping rules. The R package `Boruta` can perform the variable selection based on the Boruta algorithm.

The recursive feature elimination methods [23] is another extensively used variable selection approach based on the random forest model. This algorithm starts with fitting a random forest model on all the features, then it removes a certain proportion of features from the dataset which have the least feature importance in the previous iteration and fits a new random forest model on the remaining features. The algorithm repeats the above procedure until a single feature is left. After fitting all the random forests, the algorithm compares the Out-of-Bag (OOB) error rates from the fitted random forests and choose the model with the smallest number of features whose OOB error rate is within a given range of smallest one. As a result, the recursive feature elimination method tends to select a minimum number of features. The R package `varSelRF` can perform the variable selection based on the recursive feature elimination algorithm.

There are also many other variable selection methods based on the random forest model as listed in [24].

2.4.2 Sequence Learning methods

In addition to the previously mentioned statistical methods designed for analyzing longitudinal data, we also explored two sequence learning methods for variable selection from longitudinal datasets. Unlike other methods, these methods utilize sequential covariates information to predict a subject's eventually outcome, without accounting for changes in the subject's outcome over time. For those two methods, data needs to be structured as sequences. Since we have longitudinal data, each sequence will correspond to a subject's covariates data over time, leading up to their sole outcome. In sequence learning, each subject is treated as a data point, and a row of data within a subject is considered a time point in the sequence. Both methods require uniform dimensions for the input data, both along the time axis and the covariates axis, therefore we need to preprocess to standardize sequence lengths across the dataset. This is achieved by aligning each time point along a common time axis and padding shorter sequences with the last observed value to ensure all sequences have the same length. Due to such changes and the characteristic of the two methods not considering the changes in the subject's condition over time (thus turn the generated correlated responses into something trivial), we will not use these two methods in simulations trying different dispersion parameters.

A Long Short-Term Memory (LSTM) network [25] is a type of recurrent neural network designed for processing sequential data. It can remember both the entire sequential information over long periods and the most recent information from the immediate context of a sequence. LSTMs achieve this through a complex architecture that includes multiple gates (input, into, forget, and output) to regulate the sequential information flow. These gates determine which relevant information should be retained or discarded as the network processes a sequence of data, enabling the network to dynamically update and maintain its state over time.

The LSTM network comprises multiple LSTM units, each connected sequentially to represent different time points. Figure 3 illustrates the structure of an LSTM unit, where each letter represents a matrix. Specifically, $\mathbf{x_t}$ represents the input covariate vector at time t, a row of data in our cases (e.g. a subject's microbiome vector at time t). $\mathbf{c_{t-1}}$ represents the long-term memory, initialized as $\mathbf{0}$. $\mathbf{h_{t-1}}$ represents the short-term memory, also initialized as $\mathbf{0}$. $\mathbf{W}$ is a weight matrix to be trained. $\mathbf{f}$ stands for the forget gate, determines what percentage of the long-term memory is remembered; $\mathbf{i}$ stands for the input gate, controls the potential memory; $\mathbf{g}$ stands for the into gate, determines what percentage of potential memory to add to the

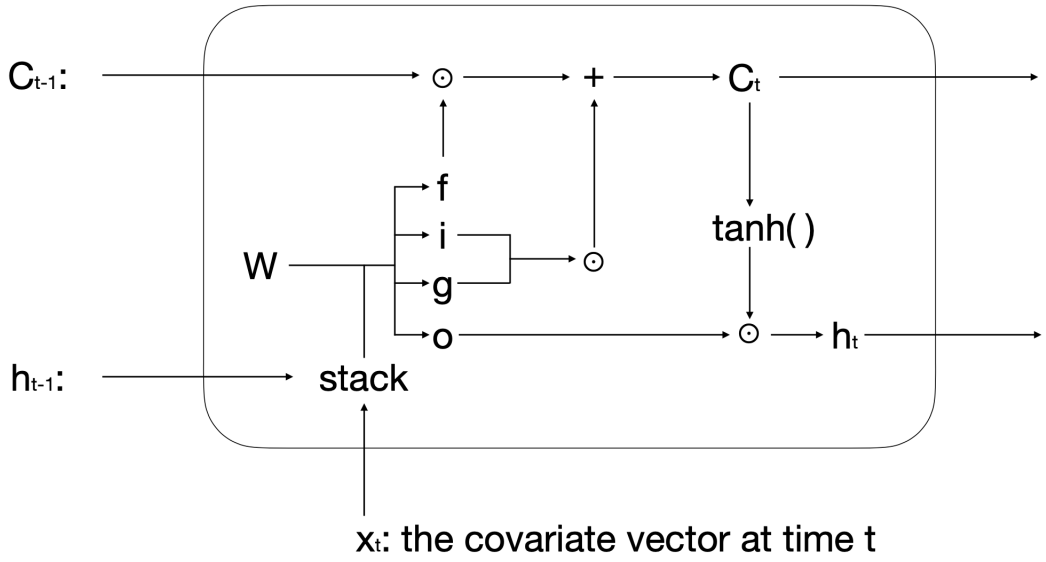

Figure 3: A unit of an LSTM model.

long-term memory; **o** stands for the output gate, controlling percentage of potential memory to add to the short-term memory. The final $\mathbf{h_t}$ is the output of the LSTM network (e.g., the probability of either eventuality of GvHD or not, to be fed into a fully connected network for binary classification).

To compute the output from the LSTM network, we simply update the components through equations below at each time points:

$$\begin{bmatrix} \mathbf{i} \\ \mathbf{f} \\ \mathbf{o} \\ \mathbf{g} \end{bmatrix} = \begin{bmatrix} sigmoid() \\ sigmoid() \\ sigmoid() \\ tanh() \end{bmatrix} \mathbf{W} \begin{bmatrix} \mathbf{h_{t-1}} \\ \mathbf{x_t} \end{bmatrix}$$

$$\mathbf{c_t} = \mathbf{f} \odot \mathbf{c_{t-1}} + \mathbf{i} \odot \mathbf{g}$$

$$\mathbf{h_t} = \mathbf{o} \odot tanh(\mathbf{c_t})$$

For the implementation of LSTM, the TensorFlow package provides us with an excellent framework. We can establish a sequence of LSTM units with just one line of code. By stacking multiple sequences of LSTM units with standard neural network components, such as fully connected layers and dropout, we can assemble our model as code. For training, since there is no inherent F1 score function in TensorFlow, we have to implement it ourselves. We also have

to manually tune the hyperparameters through cross-validation.

Self-Attention based transformer methods [26] consider the entire sequence of a subject's history at once, but they do not do so in a recurrent manner like LSTM. Each element in the sequence is processed in relation to all other elements simultaneously, rather than sequentially. This allows transformers to capture complex relationships within the data without being constrained by the sequential processing.

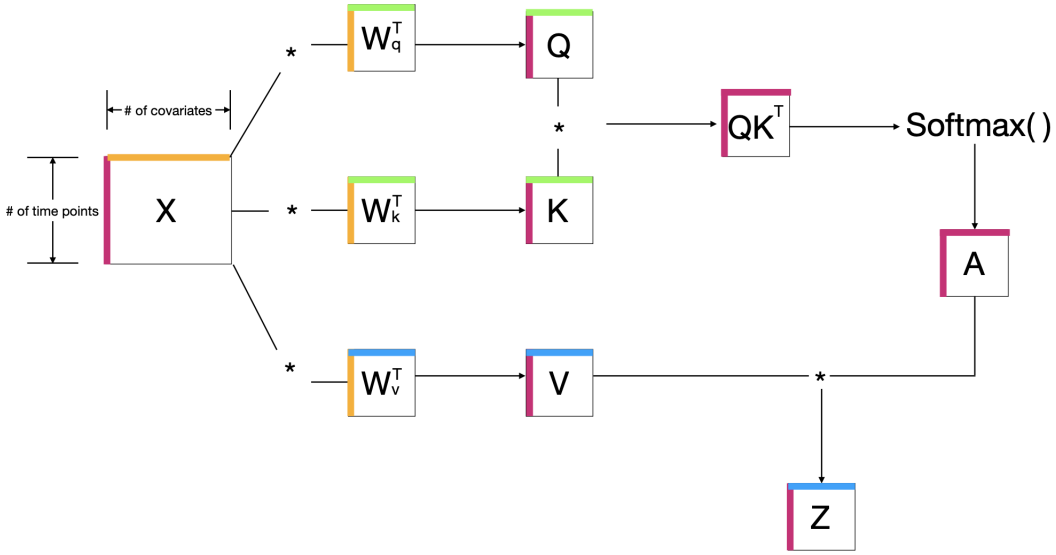

Figure 4: A unit of a self-attention model.

As illustrated in Figure 4, where dimensions are highlighted in distinct colors, the complete sequential data $\mathbf{X}$ for each subject is fed into the model. The process involves training three matrices: $\mathbf{W_q}$ (query), $\mathbf{W_k}$ (key), and $\mathbf{W_v}$ (value). These matrices undergo matrix multiplication followed by the application of softmax functions to produce a context matrix $\mathbf{Z}$. This procedure is integrated with additional components such as fully connected layers to construct a transformer layer, as detailed in the original paper [26]. The architecture comprises multiple such transformer layers. Following these, the data can be processed through fully connected layers, culminating in a binary output prediction. This process iteratively refines the input information through the transformer architecture, using attention mechanisms to enhance information extraction and training, ultimately making binary classifications.

For the implementation of the Self-Attention-based Transformer, although the TensorFlow package offers useful components for the model, such as Self-Attention units, the high complexity and customizability require us to define the model's structure on our own. This part involves a significant amount of work. For training, since there is no inherent F1 score function for tensors,

we must implement it ourselves again. We also need to manually tune the hyperparameters through cross-validation.

Neither of these two sequential methods has a built-in feature importance output. It's worth to note that we cannot use attention for feature importance because attention weights do not directly correlate to feature importance. Permutation importance is a technique used to measure the importance of each feature in models, and can be applied to any model. The general idea is to evaluate the impact of each feature on model performance by shuffling each feature in the dataset and observing how performance degrades. This process is completed one feature at a time while keeping all other features unchanged. First, the performance of the model is evaluated on the testing dataset to establish a baseline performance. Then, for each feature in the dataset, the feature's values are shuffled across all data points. This shuffling nulls the association between the feature and outcome. The performance of the model is then evaluated again. The importance of a feature is determined by the decrease of model performance: A large drop in performance indicates that the feature is very important, while a small drop suggests that the feature is not very important to the model's predictions. Repeat this process for each feature in the dataset to calculate the ranked importance of all features.

For the implementation of permutation importance, as there is no package for this, we have to implement it ourselves from scratch.

3 Simulation Study

In the following simulation study, we implement and compare most of the variable selection methods discussed in Section 2. Our main objective here is to assess the performance of univariate feature selection and multiple variable selection approaches, encompassing GLMM, GEE, and Random Forest-based methods, using simulated data. Since Tutz and Groll (2010) have already conducted simulation studies on Boosting GLMM and GAMM methods [12], and as mentioned earlier, the computational efficiency of these two methods is extremely low, we refrain from conducting additional simulations in our study. Additionally, since LSTM and Self-Attention models do not consider changes in the subject's condition over time, we excluded them from our simulation study.

3.1 Univariate Feature Selection

For the method of univariate feature selection based on zero-inflated distribution, we conducted a simulation study on a fictional correlated dataset where observations follows different zero-inflated negative binomial distributions (abbreviated as **ZINB** for later on). The goal of this simulation study is to test the accuracy of selecting true dependent covariates from noises.

We generated $m = 10, 20, 50$ or 100 subjects, each with $n = 10, 20$, or 40 observations. Built on this frame, we subsequently generated observation instances following ZINB of a fixed mean μ for each subject (which follows $N(1000, 200)$), varying dispersion parameter $\theta = 0.1, 0.2, ..., 0.9$, and varying probability of zero distribution $p = 0.1, 0.2, ..., 0.9$. For each subject, there is a binary 'state' variable of either 0 or 1, indicating the target variable. To distinguish between two type of covariates, one ZINB distribution is made conditional on the 'state' variable for true covariates and the other one is independent on the 'state' variable for noise covariates. Then we use the *zeroinfl* function from 'pscl' R package [27] to do ZINB regression of covariates on 'state' variable, selecting the statistically significant results based on p-values. Below is the accuracy plot:

The result shows a positive correlation between accuracy and number of observations. For example, for a dataset with m = 100 subjects and n=20 observations per subject, we reached an accuracy level of 0.907. This increasing accuracy indicates that this method could be well-applied to the microbiome dataset in Section 4, which presents a case study.

3.2 Multiple Variables Selection

The aim of the simulation study on multiple variable selection is to evaluate the performance of variable selection methods, which include GLMM, GEE, and Random Forest-based methods, on binary longitudinal data across different scenarios. The underlying true model is represented by a marginal logistic model:

$$\text{logit}(\mu_{ij}) = \boldsymbol{X}_{ij}^{\top}\boldsymbol{\beta},$$

where $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_p) \in \mathbb{R}^p$ with the effects given by intercept $\beta_0 = 1$ (the first column of $\boldsymbol{X}_{ij}^{\top}$ is a vector of elements of 1); $\beta_1 = -1$, $\beta_2 = -1.5$, $\beta_3 = \beta_4 = 0.5$, $\beta_5 = 1$ and $\beta_j = 0, j = 6, \ldots, k$. This means that the first 5 features are the features that should be

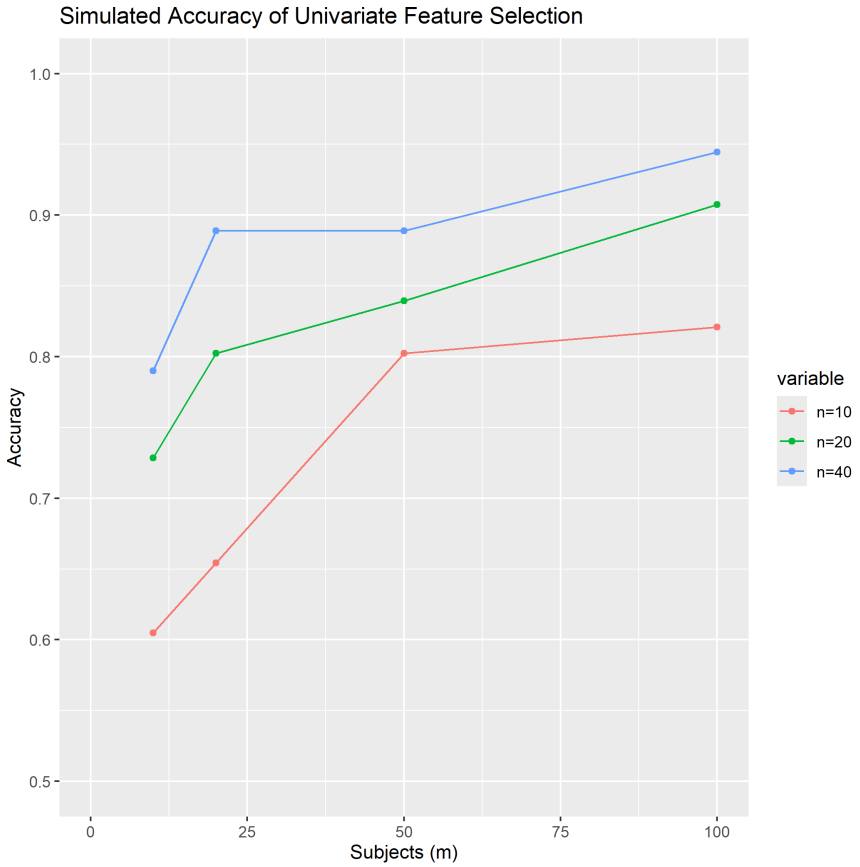

Figure 5: Zero-inflated Model Feature Selection Accuracy

selected, and other features do not have effect on our actual model, i.e., variables that should not be selected. We conduct 100 simulation datasets, each with 100 subjects ($m = 100$) and 10 observations per subject ($m = 10$). We consider different numbers of predictors ($k = 10, 20, 30$) to evaluate the scalability of the methods. Additionally, we vary the correlation coefficient (ρ) between 0.2, 0.5, 0.8 to examine the impact of correlation strength on the performance of the methods. To generate correlated response variables Y_{ij}, we utilize the `SimCorMultRes` package [28], leveraging the NORTA (Non-Overlapping Range Transformation) method. This approach ensures that the simulated datasets accurately reflect the specified correlations among the variables while preserving their marginal distributions. The true parameter vector $\boldsymbol{\beta}$ consists of 5 non-zero coefficients, reflecting the underlying true model. Our metrics for evaluating the performance of different methods include accuracy, recall, precision, and false positive rate (FPR), where accuracy= $\frac{TP+TN}{TP+TN+FP+FN}$, recall= $\frac{TP}{TP+FP}$, precision= $\frac{TP}{TP+FN}$, FPR= $\frac{FP}{FP+TN}$; predicted positive means that the variable is selected by the method, and predicted negative means that the variable is not selected by the method; TP means that the predicted positive is actually should be selected; FP means that the predicted positive is actually should not be

selected; TN means that the predicted negative is actually should not be selected; FN means that the predicted negative is actually should be selected.

As depicted in Figure 6, we observe that the mean accuracy generally increases with an increase in k for most methods, except for GLMM Lasso, where it decreases. Conversely, the mean recall tends to decrease as k increases across many methods, while the methods of adjusted p-value by Holm's and adjusted p-value by FDR stay steady. Mean precision and mean FPR exhibit minimal variation with changing k. Moreover, as ρ increases, both mean accuracy and FPR demonstrate improved performance compared to lower ρ. Notably, despite being considered naive methods, adjusted p-values on GLMM with Holm's procedure and the FDR method exhibit superior performance compared to other variable selection methods. While GLMM Lasso may not excel in metrics such as accuracy, recall, and FPR, it does demonstrate notable performance in terms of FPR.

Next, we came to the GEE based methods and Random Forest based methods. In terms of the FPR, the mean FPR of GEE based methods stays stable and below 0.2 across different values of k and different values of ρ, which means that they tend not to select the irrelevant variables. Moving to random forest methods of Boruta and recursive feature elimination (recursive), we can see that the Boruta does not perform very well when k is small, while recursive performs excellently across different simulation settings. This can be partly credited to the mechanism of the recursive feature elimination method, as it tends to select the minimal set of important features. So, we can anticipate that the mean FPR of recursive feature elimination method is very low.

Recall evaluates the capability to correctly select the important variables from all the variables that should be selected. As k increases, the GEE based methods' and Boruta's mean recall gradually drops. The recursive feature elimination method stays rather stable.

Precision evaluates the proportion of positive predictions that are actually true positives. We can see that the Boruta method does not perform very well, while the recursive feature elimination method stays between 0.7 and 0.8, while is acceptable. The lasso GEE and GEE QICu forward methods behave stable.

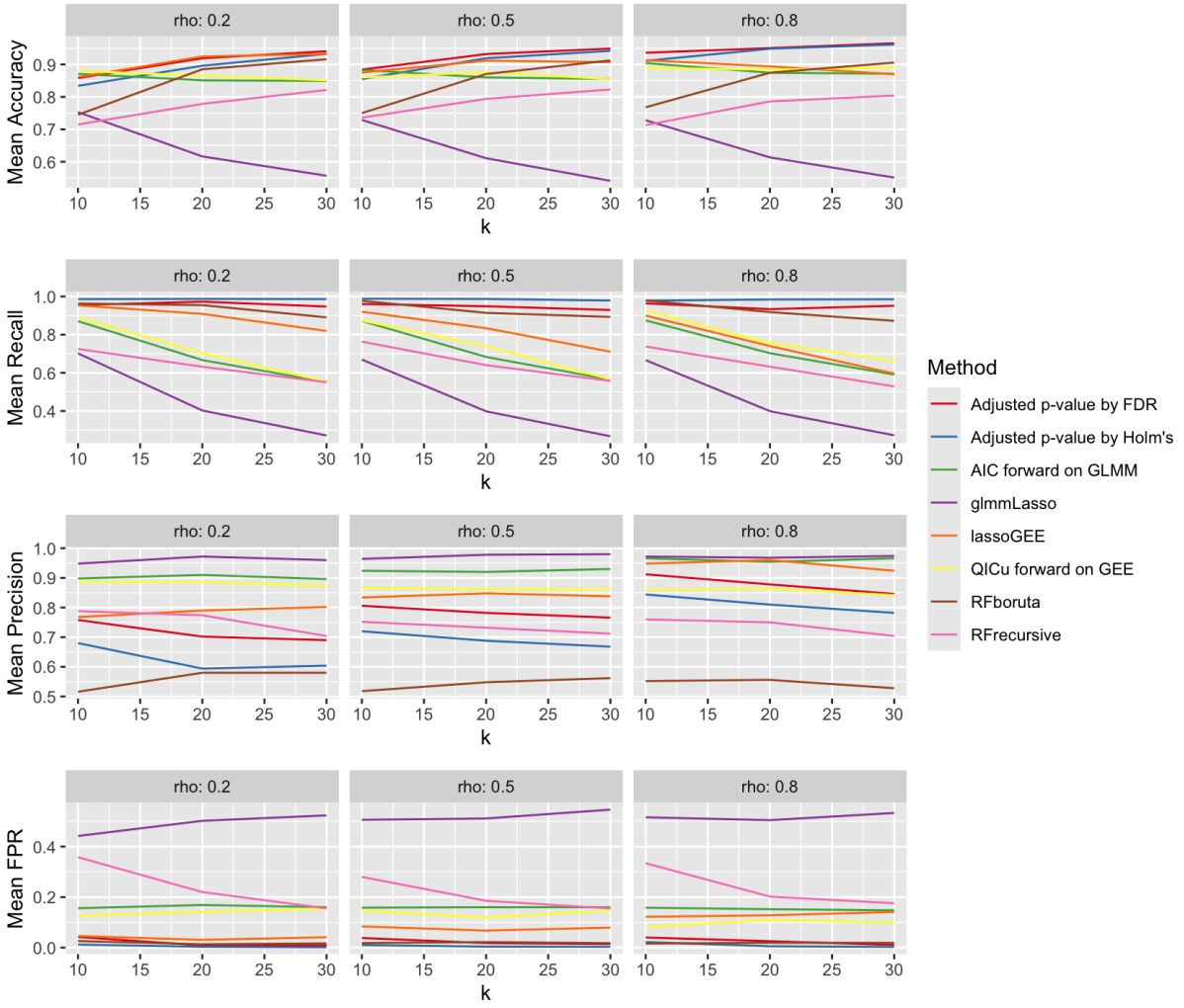

Figure 6: Simulation results on evaluating the performance of GLMM, GEE, and Random Forest-based methods.

4 Case Study

In this section, we will apply the methods reviewed in Section 2 on the GvHD microbiome data set. GvHD, which stands for "graft reacts against the host disease", is a possible complication of a stem cell or bone marrow transplant from another person. Acute GvHD generally happens within the first 100 days after a transplant and it mainly affects the skin, gut and liver. With that in mind, the data was obtained by following patients from before transplant to 100 days after the transplant, and regular stool sample was collected for each patient over time. For each sample, microbiome profiling was performed. The data set contains 2046 samples and around 800 microbiome species count. A detailed description of the data set is available in the attachment of the paper.

Studies have shown that the diversity of bacterial populations inhabit the gastrointestinal tract is associated with survival outcomes [29]. It is then of interest to gain more insights on the set of microbiomes that are associated with the onset of GvHD. This can be framed as a variable selection problem by first modeling the relationship between the onset of GvHD with microbiome species profiles, and then employ feature selection methods. To this end, we chose to omit the specific involvement of skin, liver, or gastrointestinal tract. Instead, we aggregate the variable "agvhgrd" into a binary response variable where 0 stands for the patient is not effected by GvHD at the time the sample was taken.

Some microbiomes are inherently rare and only appeared in a few samples. We chose to omit these specific species from all of our modeling since they are of little use when it comes to the consistent trend among all individuals. The specific cutoff was established to exclude species from the model if their occurrence frequency falls below 5 percent of the total number of samples. Furthermore, even though the data is longitudinal in nature, around twenty patients only have one sample available. We decided to include these individuals only when we fit non-longitudinal models.

In the simulation study where the non-zero coefficients were known beforehand, we defined various metrics based on the true coefficients to evaluate the performance of different methods. The true relationship between each microbiome species and the onset of GvHD is not yet known to us, which makes defining a performance metric on the species level difficult. However, previous studies have shown microbiome genus: Eubacterium, Blautia, Anaerostipes, Coprococcus and Ruminococcus are associated with the onset of GvHD [30]. Then let M_g denotes the set of

the above five genus, one can consider the following simple metric:

$$a_i = \frac{|S_i \cap M_g|}{|M_g|}$$

where S_i is the set of features selected by method i, $|\cdot|$ denotes the number of elements in a finite set. The metric simply stands for the fraction of the study-verified-genus that a method selected out. It is worth noting that after omitting the rare species, the data set contains around 110 aggregated genus. Follow from our discussion in Section 2, computations under such dimensionality proves cumbersome for models like GAMM and boosting-based algorithms. Therefore, for these methods, we consider the corresponding family of the above five genus, which yields: Eubacteriaceae, Lachnospiraceae, Ruminococcaceae. Then a similar metric on family level can be defined as

$$b_i = \frac{|S_i \cap M_f|}{|M_f|}$$

where M_f denotes the set of the above three families.

4.1 Univariate Feature Selection on Microbime Data

To apply the univariate feature selection method using zero-inflated regression model into microbiome data, additional data preprocessing is required. Such is the nature of microbiome data that the count data is not directly applicable into the regression model. One common practice to solve this problem is the fix the overall size of the microbiome and calculate the relative abundance for each bacteria. This initial process transformed the count data into a real number between 0 and 1. Then we re-transform the relative abundance into count data by multiplicating and rounding. Consequently, the dataset after processing can be modeled as count data, and simultaneously represented as relative abundance.

We started the feature selection process on two different levels of bacteria: *genus* and *species*. For each level, we used the ZINB to regress bacteria count on target variable (agvhgrd) and other covariates including subject id(patient_ID). Here's a snapshot of what we have filtered during this process using the p-values:

Some of the bacteria to note about includes *Blautia*, *Anaerostipes*, and *Akkermansia*, which also appeares in relevant study of gut microbiome and aGVHD [31]. Based on the filtering of p-values, the diagram provides further study a basis of what could be chosen as the preliminary

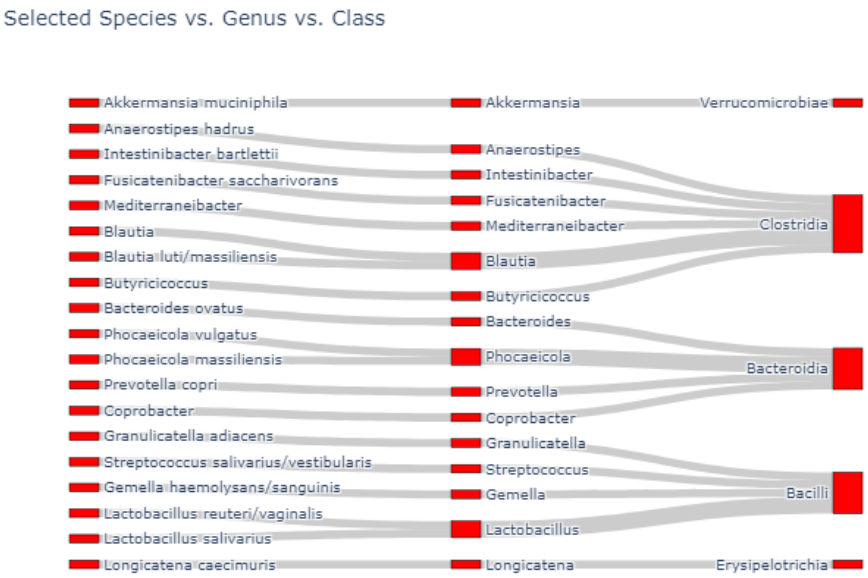

Figure 7: Diagram of selected features on three different levels

features for statistical learning.

4.2 Multiple Variable Selection on Microbime Data

4.2.1 GEE based methods

In this section, we employed the GEE forward variable selection with QIC_u method and Lasso GEE in the microbiome data. The GEE forward method selects 23 features, and two of them are confirmed in the previous study: Anaerostipes and Coprococcus. So, $a_{geeforward} = 0.4$.

The Lasso GEE method selects 66 features, and four of them are confirmed in the previous study: Anaerostipes, Ruminococcus, Coprococcus, and Eubacterium. So, $a_{lassogee} = 0.8$.

The Lasso GEE selects more confirmed features than the GEE forward method, though the GEE forward gives a much smaller range of potential important features related to the GvHD disease. Further, the main drawback of the GEE forward method is that its nature of greedy algorithm may keep it staying at the local optimum instead of the global optimum. Also, as the dimension grows, the computation time of forward algorithm may become unacceptable, and the local optimum that the algorithm stays at may be far from the true global optimum.

4.2.2 GLMM based methods

In this section, we explore various GLMM-based variable selection methods for analyzing microbiome data, including adjusted p-values with Holm's procedure and false discovery rate (FDR), AIC forward selection, GLMM Lasso, and Boosting GLMM. To ensure the appropriate application of these methods to microbiome data, it's essential to rescale the varying values for each microbiome. Failure to do so could lead to computational issues such as matrix and gradient degeneration.

We begin by employing adjusted p-values with Holm's procedure on GLMM, which identifies 5 features. Among these, Anaerostipes is confirmed in previous studies, indicating $a_{GLMMholm} = 0.2$. Similarly, using adjusted p-values with Holm's procedure on GLMM selects 12 features, with Anaerostipes also confirmed, suggesting $a_{GLMMfdr} = 0.2$.

Additionally, AIC forward selection on GLMM identifies 42 microbiomes, although only Anaerostipes is confirmed in previous studies, resulting in $a_{GLMMforward} = 0.2$. GLMM Lasso selects 5 microbiomes, with Anaerostipes also confirmed in prior research, implying $a_{GLMMLasso} = 0.2$.

Finally, employing Boosting GLMM, implemented using the `bGLMM` package in R with 100 boosting interations, identifies 14 microbiomes, with Anaerostipes being the only confirmed species from previous studies, corresponding to $a_{bGLMM} = 0.2$. It's noteworthy that all GLMM-based methods select Anaerostipes.

4.2.3 Random Forest based methods

In this section, we employed two variable selection methods based on the Random Forest model: Boruta and recursive feature elimination. The Boruta method selects 68 features, and five of them are confirmed in the previous study: Blautia, Anaerostipes, Coprococcus, Ruminococcus, and Eubacterium. So, $a_{boruta} = 1$.

The recursive feature elimination method selects 59 features, and five of them are confirmed in the previous study: Blautia, Anaerostipes, Coprococcus, Ruminococcus, and Eubacterium. So, $a_{boruta} = 1$.

The recursive feature elimination tends to select the minimal set of important features. In this example, we can see that the recursive feature elimination selects less features than the Boruta method while keeping the performance of selecting those five confirmed features.

The methods based on the Random Forest model are particularly adaptive variable selection methods as they are suitable to many kinds of scenario and keep a stable performance. If we prefer to narrow down the range of potential important features, the recursive feature elimination can be a good choice in terms of easy-implementation, time complexity, and performance.

4.2.4 Sequence Learning methods

As mentioned previously, we are using the permutation importance method to select variables for these two sequential learning methods — LSTM and Self-Attention. When evaluating the impact of shuffled variables on model performance, we chose the F1-score as our metric. This involves shuffling each variable and calculating the difference in the F1-score before and after the shuffle to determine variable importance.

For the LSTM model, the selected variables are Ruminococcus, Faecalibacterium, and Flavonifractor (with other variables assigned zero importance). So $a_{LSTM} = 0.2$ as only Ruminococcus is captured.

For the Self-Attention model, the selected variables are Bacteroides, Blautia, Escherichia_Shigella, Roseburia, Citrobacter, Limosilactobacillus, Alistipes, Pediococcus, Lachnoclostridium, Hungatella, Pseudoflavonifractor/Clostridium, Massilimicrobiota, and Merdimonas (with other variables assigned zero importance). So $a_{selfattention} = 0.2$ as only Blautia is captured.

In applying the permutation importance method to these two models, we encountered the issue of many variables of zero importance. We believe this is due to the small size of our dataset after preprocessing (noting that we treated each subject's response as constant, based on whether the subject eventually developed GvHD) and the fact that our task involves binary classification, which is discrete. As a result, shuffling many variables still led the model to produce the same predictions and, consequently, the same F1 score and zero importance.

5 Discussion

For GEE model, we only need to specify the marginal mean and variance structure, marginal mean model, and correlation structure, which makes it relatively easy to implement and also robust. In the simulation study, the Lasso GEE performs better than the GEE foward

algorithm with QIC_u in most cases in terms of mean accuracy, mean recall, and mean FPR. In terms of mean precision, though the Lasso GEE does not perform as well as GEE foward when ρ is small, it becomes much better when ρ grows as large as 0.8. When it comes to the computation cost and robustness, it is clearly that the Lasso GEE is a better choice than the forward/backward/both directions algorithm. When k becomes much larger and the model structure becomes much complex, the computation cost of forward/backward/both directions algorithm is likely to become unacceptable. Also, as mentioned before, the main drawback of the GEE forward method is that its nature of greedy algorithm may keep it staying at the local optimum instead of the global optimum. This problem can be clear when looking at the results of the case study of microbime data: though the Lasso GEE selects a much larger range of features, it at least selects four features which have been confirmed as closely related to GvHD by former study, while the forward algorithm only selects two of them and costs much longer time. So, combining the GEE QIC_u criterion with other algorithms with randomness may be a better choice. For example, the idea of Markov chain Monte Carlo model composition MC^3 algorithm may be a potential choice to be modified. Overall, the Lasso GEE method can be a better choice than the GEE foward method.

GLMM is a classic approach for modeling correlated data. In our simulation study, simple GLMM variable selection methods, such as those using adjusted p-values via Holm's procedure or false discovery rate (FDR), exhibit strong performance in accuracy, recall, and false positive rate (FPR). Although precision falls short initially, it tends to improve with higher correlation (ρ) values. On the other hand, forward selection based on AIC for GLMM underperforms for both simulation and case study. While both adjusted p-values and AIC forward selection identify Anaerostipes as an important microbiome related to acute GvHD, the former methods select only a handful of variables, whereas the latter selects nearly 42 variables encompassing 110 microbiomes at the Genus level. In addition, as the number of predictors increases, fitting GLMMs using `glmer` in R becomes time-consuming and may yield convergence warnings or errors. Tweaking hyper-parameters of the model can mitigate these issues but it's in the cost of precision and computation time. In contrast, GLMM Lasso demonstrates favorable performance in accuracy, recall, FPR, and precision. Additionally, its efficient model fitting process further underscores its utility in microbiome data analysis, offering precise and efficient feature selection. Finally, boosting GLMM proves to be highly time-consuming due to its computational demands.

Despite its ability to successfully identify GvHD related microbiome Anaerostipes like other GLMM-based methods, conducting a simulation study with boosting GLMM is challenging.

Univariate Feature Selection based on ZINB distribution performs well in detecting meaningful covariates admist noise in the simulation scenario. It's a robust method, with its performance is positively correlated with the number of observations. Implementing such algorithm is computationally efficient with existing R packages. However, additional implentations are needed to apply it into real-world dataset, like the microbiome dataset. In this case study, although it successfully selected certain bacteria of significance, it also incorporates irrelevant covariates into the final outcome. There are ways to expand this method: like implementing a ZINB emission probability for Hidden Markov Model. Apart from this, this method relies heavily on the underlying assumption of data distribution being a zero-inflated distribution. Thus it lacks the ability to be generalized into diverse datasets.

The Random Forsest model is a very powerful and adaptive model. In the simulation study, we have two variable selection methods based on the Random Forest model: Boruta and recursive feature elimination. In terms of mean accuracy, mean recall, and mean FPR, the recursive feature elimination performs significantly better than the Boruta method across different values of k and rho. We can notice that the recursive feature elimination method is actually one of the methods that performs best with respect to the metrics of accuracy, recall, and FPR, even when k and ρ are large. In addition, if our aim is to ensure the precision of the variable selection of true positive feature, i.e. the true important feature, the Boruta method performs may be a better choice than the recursive feature elimination. In the case study, we can see that both these two methods select all the five features which have been confirmed as closely related to GvHD by former study mentioned before. However, if we prefer to narrow down the range of potential important features, the recursive feature elimination can be a good choice as it gives a minimal set of important features. These methods based on the Random Forest model have the advantage of easy-implementation and less computation cost. Also, they are very adaptive as they are suitable to all kinds of responses and features without the need to specify a statistical model. So, the recursive feature elimination method can be recommended for a variable selection task. In the future, we can combine these variable selection methods with modified random forest models which are more suitable to correlated data as mentioned in Hu and Szymczak (2023) [32], and it has the potential to further push the performance of

variable selection to a even higher level.

Both LSTM and Self-Attention models are cutting-edge, demonstrating versatility across various predictive applications, from time series data to i.i.d. tabular data. However, in this paper, we address longitudinal data, which emphasizes the relationships between outcomes within the same subject. Due to the constraints of our method design, as detailed in Section 2.4.2, we had to break this relationship, resulting in each subject having a constant outcome. Consequently, we utilized a single outcome per subject as the target variable for model fitting. This preprocessing led to a very small dataset, even smaller than the number of covariates, leading to very limited testing data (67 subjects). We observed that the binary predictions made by both sequential learning models on the testing data were completely identical. Furthermore, the variables selected by the models through feature importance analysis performed poorly according to our designated metric. This indicates that while these models are powerful, they cannot show interpretation from in datasets of small size. The uniformity of predictions and the failure to select meaningful variables suggest that, despite their advanced capabilities, both models encounter challenges in generating useful insights from small datasets. The training times for both models are short, taking less than 2 minutes each on the microbiome data. However, the process of calculating feature importance demands significantly more computation. This involves shuffling one variable each time and training five separate models on new data to average the decrease in performance (F1-score). This process repeated across 110 variables. Consequently, for each of the two methods, we have to conduct 110*5=550 such training and evaluations, which is quite time-intensive. Given the challenges in identifying meaningful variables, along with the long implementation and feature selection times, we do not recommend these two sequential learning methods. So it is important to emphasize that for longitudinal data, statistical methods mentioned earlier might be more suitable than these sequential learning methods, as they are specifically designed to account for the correlations inherent within such subject. To improve existing sequence learning-based methods in the future, we may explore more neural network-tailored methods (v.s. permutation importance), such as the gradient-based feature importance ranking methods mentioned in Wojtas and Chen (2020) [33].

In this research, we have been focused on the development of variable selection methods based on GEE, GLMM and machine learning methods for correlated longitudinal binary responses, with the consideration of method development time, computation cost and variable

selection accuracy. When it comes to modeling correlated binary responses, we recommend the following: (1) GEE models, in particular Lasso GEE to be a good starting point. Its outstanding performance, coupled with minimal model development and implementation, positioned it as a practical method suitable for a wide range of circumstances. (2) GLMM based methods and its extension GAMM should be employed with domain knowledge. The automated variable selection methods for these models investigated in this paper all requires significant computational expenses. Such burden can be mitigated when the researchers are certain of which subset of covariates should be included in the model, allowing for selective inclusion of the remaining variables. (3) Zero-Inflated Negative Binomial methods is robust and have strong performance. Though it requires minor additional modification of the data, it should be employed whenever the underlying assumptions are satisfied. (4) Adapted machine learning algorithms can be employed for further detailed analysis. We have demonstrated the strong performance of random forest and neural net models. These methods can be further modified to better incorporate the correlated structure of the data set. However, the method needs to be carefully designed, and the lack of existing software implementation of such methods can hinder the research progress.

Appendix

Table 1: Simulation results on evaluating the performance of GLMM, GEE, and Random Forest-based methods with $\rho = 0.2$

k	method	accuracy	recall	precision	FPR
10	Adjusted p-value by Holm's	0.834	0.987	0.68	0.012
	Adjusted p-value by FDR	0.858	0.955	0.758	0.042
	glmmLasso	0.753	0.702	0.948	0.442
	AIC forward on GLMM	0.871	0.871	0.898	0.156
	QICu forward on GEE	0.88	0.89	0.886	0.126
	RFboruta	0.745	0.963	0.516	0.026
	RFrecursive	0.715	0.725	0.788	0.358
	lassoGEE	0.861	0.953	0.768	0.046
20	Adjusted p-value by Holm's	0.896	0.988	0.594	0.003
	Adjusted p-value by FDR	0.919	0.973	0.702	0.009
	glmmLasso	0.617	0.403	0.972	0.502
	AIC forward on GLMM	0.851	0.667	0.91	0.169
	QICu forward on GEE	0.866	0.703	0.886	0.141
	RFboruta	0.885	0.955	0.58	0.013
	RFrecursive	0.778	0.632	0.774	0.22
	lassoGEE	0.924	0.909	0.79	0.031
30	Adjusted p-value by Holm's	0.933	0.987	0.604	0.002
	Adjusted p-value by FDR	0.941	0.947	0.69	0.009
	glmmLasso	0.557	0.272	0.96	0.524
	AIC forward on GLMM	0.849	0.552	0.896	0.16
	QICu forward on GEE	0.851	0.554	0.874	0.154
	RFboruta	0.916	0.891	0.58	0.017
	RFrecursive	0.821	0.551	0.704	0.155
	lassoGEE	0.933	0.819	0.802	0.041

Table 2: Simulation results on evaluating the performance of GLMM, GEE, and Random Forest-based methods with $\rho = 0.5$

k	method	accuracy	recall	precision	FPR
10	Adjusted p-value by Holm's	0.855	0.989	0.72	0.01
	Adjusted p-value by FDR	0.884	0.961	0.806	0.038
	glmmLasso	0.729	0.67	0.964	0.506
	AIC forward on GLMM	0.883	0.872	0.924	0.158
	QICu forward on GEE	0.86	0.874	0.866	0.146
	RFboruta	0.75	0.978	0.518	0.018
	RFrecursive	0.736	0.763	0.752	0.28
	lassoGEE	0.875	0.92	0.834	0.084
20	Adjusted p-value by Holm's	0.919	0.987	0.688	0.004
	Adjusted p-value by FDR	0.932	0.949	0.782	0.018
	glmmLasso	0.611	0.398	0.978	0.511
	AIC forward on GLMM	0.86	0.683	0.92	0.16
	QICu forward on GEE	0.876	0.739	0.862	0.119
	RFboruta	0.871	0.914	0.548	0.022
	RFrecursive	0.794	0.64	0.732	0.185
	lassoGEE	0.911	0.834	0.848	0.067
30	Adjusted p-value by Holm's	0.942	0.979	0.668	0.003
	Adjusted p-value by FDR	0.949	0.929	0.766	0.014
	glmmLasso	0.541	0.268	0.98	0.546
	AIC forward on GLMM	0.855	0.564	0.93	0.16
	QICu forward on GEE	0.855	0.566	0.862	0.147
	RFboruta	0.912	0.893	0.562	0.018
	RFrecursive	0.823	0.558	0.712	0.155
	lassoGEE	0.907	0.711	0.838	0.079

Table 3: Simulation results on evaluating the performance of GLMM, GEE, and Random Forest-based methods with $\rho = 0.8$

k	method	accuracy	recall	precision	FPR
10	Adjusted p-value by Holm's	0.911	0.978	0.844	0.022
	Adjusted p-value by FDR	0.936	0.964	0.912	0.04
	glmmLasso	0.728	0.667	0.972	0.516
	AIC forward on GLMM	0.904	0.875	0.966	0.158
	QICu forward on GEE	0.889	0.926	0.86	0.082
	RFboruta	0.768	0.979	0.552	0.016
	RFrecursive	0.713	0.738	0.76	0.334
	lassoGEE	0.913	0.9	0.948	0.122
20	Adjusted p-value by Holm's	0.949	0.985	0.81	0.005
	Adjusted p-value by FDR	0.951	0.934	0.878	0.025
	glmmLasso	0.614	0.399	0.968	0.505
	AIC forward on GLMM	0.875	0.703	0.954	0.152
	QICu forward on GEE	0.884	0.756	0.864	0.109
	RFboruta	0.875	0.918	0.556	0.019
	RFrecursive	0.786	0.632	0.75	0.202
	lassoGEE	0.894	0.74	0.96	0.128
30	Adjusted p-value by Holm's	0.961	0.986	0.782	0.003
	Adjusted p-value by FDR	0.965	0.952	0.846	0.011
	glmmLasso	0.551	0.273	0.974	0.533
	AIC forward on GLMM	0.871	0.591	0.966	0.148
	QICu forward on GEE	0.89	0.662	0.842	0.1
	RFboruta	0.906	0.873	0.528	0.019
	RFrecursive	0.804	0.529	0.704	0.176
	lassoGEE	0.87	0.596	0.924	0.141

References

[1] Georg Heinze, Christine Wallisch, and Daniela Dunkler. Variable selection - a review and recommendations for the practicing statistician. *Biometrical journal. Biometrische Zeitschrift*, 60(3):431–449, 2018.

[2] Dan Chen. *Variable Selection for Longitudinal Data with Correlated Binary Responses*. PhD thesis, ProQuest Dissertations and Theses, 2019. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2023-06-21.

[3] Kung-Yee Liang and Scott L Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22, 1986.

[4] Wei Pan. Akaike's information criterion in generalized estimating equations. *Biometrics*, 57(1):120–125, 2001.

[5] Wenjiang J Fu. Penalized estimating equations. *Biometrics*, 59(1):126–132, 2003.

[6] Norman E Breslow and David G Clayton. Approximate inference in generalized linear mixed models. *Journal of the American statistical Association*, 88(421):9–25, 1993.

[7] John C Nash and Ravi Varadhan. Unifying optimization algorithms to aid software system users: optimx for r. *Journal of Statistical Software*, 43:1–14, 2011.

[8] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[9] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[10] Hirotogu Akaike. Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer, 1998.

[11] Andreas Groll and Gerhard Tutz. Variable selection for generalized linear mixed models by l 1-penalized estimation. *Statistics and Computing*, 24:137–154, 2014.

[12] Gerhard Tutz and Andreas Groll. Generalized linear mixed models based on boosting. *Statistical modelling and regression structures: festschrift in honour of ludwig fahrmeir*, pages 197–215, 2010.

[13] Xihong Lin and Daowen Zhang. Inference in generalized additive mixed models by using smoothing splines. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 61(2):381–400, 1999.

[14] Simon Wood. *Generalized Additive Models: An Introduction With R*, volume 66. 01 2006.

[15] Simon N Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(1):3–36, 09 2010.

[16] Simon Wood and Fabian Scheipl. *gamm4: Generalized Additive Mixed Models using 'mgcv' and 'lme4'*, 2020. R package version 0.2-6.

[17] Giampiero Marra and Simon N. Wood. Practical variable selection for generalized additive models. *Computational Statistics Data Analysis*, 55(7):2372–2387, 2011.

[18] Andreas Groll and Gerhard Tutz. Regularization for generalized additive mixed models by likelihood-based boosting. *Methods of information in medicine*, 51(2):168–177, 2012.

[19] William H. Greene. Accounting for excess zeros and sample selection in poisson and negative binomial regression models. *NYU Working Paper No. EC-94-10*, 3 1994.

[20] UCLA: Statistical Consulting Group. Zero-inflated negative binomial regression — r data analysis examples.

[21] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[22] Miron B Kursa and Witold R Rudnicki. Feature selection with the boruta package. *Journal of statistical software*, 36:1–13, 2010.

[23] Ramon Diaz-Uriarte and Sara Alvarez de Andrés. Variable selection from random forests: application to gene expression data. *arXiv preprint q-bio/0503025*, 2005.

[24] Frauke Degenhardt, Stephan Seifert, and Silke Szymczak. Evaluation of variable selection methods for random forests and omics data sets. *Briefings in bioinformatics*, 20(2):492–503, 2019.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[27] Achim Zeileis, Christian Kleiber, and Simon Jackman. Regression models for count data in R. *Journal of Statistical Software*, 27(8), 2008.

[28] Anestis Touloumis. Simulating correlated binary and multinomial responses under marginal model specification: The simcormultres package. *The R Journal*, 8(2):79–91, 2016. R package version 1.9.0.

[29] Ying Taur, Robert R. Jenq, Miguel-Angel Perales, Eric R. Littmann, Sejal Morjaria, Lilan Ling, Daniel No, Asia Gobourne, Agnes Viale, Parastoo B. Dahi, Doris M. Ponce, Juliet N. Barker, Sergio Giralt, Marcel van den Brink, and Eric G. Pamer. The effects of intestinal tract bacterial diversity on mortality following allogeneic hematopoietic stem cell transplantation. *Blood*, 124(7):1174–1182, 08 2014.

[30] Marina Burgos da Silva, Doris M. Ponce, Anqi Dai, Sean M. Devlin, Antonio L. C. Gomes, Gillian Moore, John Slingerland, Roni Shouval, Gabriel K. Armijo, Susan DeWolf, Teng Fei, Annelie Clurman, Emily Fontana, Luigi A. Amoretti, Roberta J. Wright, Hana Andrlova, Oriana Miltiadous, Miguel-Angel Perales, Ying Taur, Jonathan U. Peled, and Marcel R. M. van den Brink. Preservation of the fecal microbiome is associated with reduced severity of graft-versus-host disease. *Blood*, 140(22):2385–2397, 12 2022.

[31] Emma E Ilett, Mette Jørgensen, Marc Noguera-Julian, Jens Christian Nørgaard, Gedske Daugaard, Marie Helleberg, Roger Paredes, Daniel D Murray, Jens Lundgren, Cameron MacPherson, Joanne Reekie, and Henrik Sengeløv. Associations of the gut microbiome and clinical factors with acute GVHD in allogeneic HSCT recipients. *Blood Adv.*, 4(22):5797–5809, November 2020.

[32] Jianchang Hu and Silke Szymczak. A review on longitudinal data analysis with random forest. *Briefings in Bioinformatics*, 24(2):bbad002, 2023.

[33] Maksymilian Wojtas and Ke Chen. Feature importance ranking for deep learning. *arXiv preprint arXiv:2010.08973*, 2020. Accepted by NeurIPS 2020.