### Materials and Methods

Data Processing

The raw ASV format tables were processed in a way to yield a machine learning format. A python 3 script was used to transform the tables, using ‘pandas’ package of version 1.4.3. Tables were transposed, duplicates removed, rows re-indexed and ordered based on sample names accordingly. The final data frame was created by merging processed ASV and response tables into one.

Abundance Analysis

Microbial abundance was analysed using R version 4.2.1. To calculate shanon entropy, function ‘entropy()’ from the package ‘agrmt’ version 1.42.8 was used. To get the simpsons index, function named ‘simpson()’ was used from the package ‘abdiv’ version 0.2.0. Hill indexes were calculated using the function ‘Hill()’ from the package ‘rasterdiv’ of version 0.2.5.2. Gamma diversity index was computed using the ‘gamma\_div()’ function from the package hilldiv, version 1.5.1. Lastly for any data transformations, the package ‘dplyr’ version 1.0.10 was utilized.

Shannon entropy and Simpson’s D index were calculated on the “growth vs. reduction” dichotomized relative “ASV” counts. Both values were derived by column-wise calculations, where the mean across all these values was used for comparison between dichotomies.

Hill’s Index was calculated using the following parameters: “window = 3”, “alpha = 1”, “rasterOut=TRUE”, “np = 1”, “na.tolerance=1.0”, “cluster.type = "SOCK"”, for both dichotomies. The actual index was derived by first calculating the mean for each species in both cohorts, column-wise. After that the whole mean for each of the “growth” or “reduction” dichotomy was calculated, and use in non-parametric comparison.

Gamma index was calculated on the exact same datasets as Shannon entropy and Simpson’s Index, with the addition of the parameter “q = 0”, required for the “gamma\_div()” funciton.

Training Classifiers using eXtreme Gradient Boosting (XGB)

For boosting, the package ‘xgboost’, version 1.6.0.1 was used. A base classification model was trained for 600 training iterations using the following hyperparametes; learning rate was set to 0.001, gamma pruning rate to 0, maximum tree depth to 9, column sample by tree to 1, subsample to 1 and minimal child weight to 1. A tree booster of type ‘gbtree’ was used for gradient boosting.

To look into the ‘black-box’ of the classifier, SHAP values were calculated using the same package, and plotted for global importance (i.e. bar plot of how much each feature contributed regardless of prediction direction ranked in descending order) and local importance (i.e. swarm plot of each features importance to predicting either ‘reduction’ or ‘growth’ ranked in descending order).

Important features identified in the high dimensional space, were further analysed for any statistical significance using the Willcox Rank Sum Test (also called the Mann Whitney U Test).

Adding Selection Robustness via a Probabilistic Model

A probabilistic model was developed, as a way to strengthen the selection and rankings of SHAP values per microorganism count. The dataset (N=22) was trained using gradient boosting and cross-validated using k-fold cross-validation. Dataset was reshuffled in a 80%-20% train-test split and 5000 was selected as the number for ‘k’. SHAP values were collected for all folds and histograms were plotted to explore distribution of each individual feature. A median SHAP value was calculated for each SHAP value and a final distribution of these medians were plotted using a histogram plot. The Shapiro-Wilk normality test was applied to all distribution of relevant size (where N > 3) and the final median distribution.

After the identification of the 2 most important bacteria,

Avoiding Model Over-fitting by using k-fold Cross Validation

To prevent the final predictive classifier from over-fitting, a k-fold cross-validation technique was implemented, with a train-test split of 80-20%, respectively. The dataset (N=22) was reshuffled using the ‘sample()’ method found in native R and normalized using the ‘log1p’ approach, also called the ‘log (1 + x)’, due to the dataset consisting of some zero values. A portion of 0.8 shuffled indices extracted from the original dataset was used to create the training dataset, while the rest of 0.2 was used to get the testing dataset. An additional constraint was added, that prevented the creation of samples with factors of only 1-level (e.g. sampling a group of only negative/positive labels). This was repeated for k-folds, where ‘k’ was selected to be 1000. For every 1000 splits, a logistic regression classifier was trained on the training dataset and validated on the test datasets. For each test dataset, a ROC-AUC curve was generated, and metrics such as accuracy, precision, recall, F-score, Youden’s J index and AUC score were calculated. Each metric was collated in a single table, together with 95% and 99% confidence intervals for the AUC score. The mean of each metric mentioned above was used as the final performance of the cross-validation.

Final Univariate Logistic Regression Biomarker Model

A binary logistic regression model was trained on the selected microorganisms, using base R functions. The model was trained using the logarithm of the ratio between bacteria a and b. Prediction probabilities were converted back to discrete probabilities using a 0.3 threshold, which was selected experimentally. A graph called area under the receiver operating curve (ROC-AUC) was created to visualise the true vs. false positive rate of the model and an AUC value was computed to measure model performance. This was done using the package ‘pROC’, version 1.18.0. Moreover, metrics such as accuracy, precision, recall, F1 and Youden’s J index were also calculated, for each model trained on different biomarkers in question. All the above mentioned metrics except the AUC score were calculated using ‘confusionMatrix()’ , ‘precision()’, ‘recall()’ and ‘F\_meas()’ functions from the ‘caret’ package, version ‘6.0-93’. Exactly the same method for metric calculation was done for cross-validation metrics as well.