### Materials and Methods

Data Processing

The raw amplicon sequence variant tables were processed in a way to yield a machine learning format. A python 3 script was used to transform the tables, using the ‘pandas’ package, 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 files and response tables into one data frame.

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 microbial counts, from their corresponding ASV files. 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()” function.

Training Classifiers using eXtreme Gradient Boosting (XGB)

Binary logistic classifiers were trained using the package ‘xgboost’, version 1.6.0.1, on a dataset of 22 samples for each amplicon sequence variant type. For each taxa from kingdom down to species, a classification model was trained for 600 training epochs 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 as the booster type, to ensure harnessing the ensemble power of decision trees as weak learners.

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).

Since the ranking of feature importances for a single XGB trained model were shown to vary across repeated training, the model was resampled many times to increase confidence. Furthermore, all the 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).

Increasing SHAP confidence via a probabilistic framework

A probabilistic model was developed, as a way to strengthen the selection and rankings of SHAP values per microorganism count. This was done by repeating the following procedure of data sampling into training and testing subsets, training on the larger sample, and using the smaller for testing for 1, 2, 20, 1000 and 5000 iterations. 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. To confirm the non-normal distributions, the Shapiro-Wilk normality test was applied to all distribution of relevant size (where N > 3) as well as on the final median distribution. This opened the possibility to test each median SHAP value against its population median, to uncover all SHAP that were to extreme given the distribution. This was done using a one sample (two-tailed) Wilcoxon signed-rank test.

Avoiding Model Over-Fitting by Implementing A k-fold Cross Validation

To prevent the final predictive classifier from over-fitting, a k-fold cross-validation technique was implemented, where the value k was choosen to be 3. The dataset was shuffled using the ‘sample()’ method found in native R and split into 3 equally sized folds with a “plus one” in the last fold. Two folds were taken as the training set, where the k-1 fold was used for testing. This was repeated until all folds were used for testing at most once. Metrics such as accuracy, precision, recall, F-score, Youden’s J index and AUC score were calculated. To limit the variability of the results, the 3-fold cross-validation was repeated at least 6 times. 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 mesurment of the cross-validation.

Final Logistic Regression Model

A binary logistic regression model was trained on the selected biomarkers, using base R functions. All dependent variables were normalized using a log-transform approach. 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’. Same package and functions were used for cross-validation metrics as well.