Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea magistrale in Bioingegneria

Feature selection and classification for Metagenomics diagnosis of Inflammatory Bowel Diseases

Relatore:
Prof.ssa Barbara Di Camillo

Laureando:
Filippo Pietrobon

Correlatore:
Dott. Sebastian Daberdaku

Anno accademico 2019/2020

Contents

Chapter 1

Introduction

This thesis deals with the discriminative power of metagenomics in the field of Inflammatory Bowel Diseases. Currently, endoscopy represents the gold standard in the diagnosis of these conditions, but it is necessary to identify novel molecular biomarkers in order to use less invasive methods.

More specifically, this thesis is focused on the implementation and the performance assessment of two algorithms of feature selection, and on the classification on microbiome data.

1.1 The DNA

The Deoxyribonucleic acid (DNA) is a molecule composed of two chains that coil around each other to form a double helix carrying genetic instructions for the development, growth and reproduction of all organisms and many viruses. The two DNA strands are also known as polynucleotides as they are composed of simpler monomeric units called nucleotides, respectively: cytosine (C), guanine (G), adenine (A) and thymine (T). Each of which is composed also by a sugar, called deoxyribose, and a phosphate group. The nucleotides are joined together to one another in a chain by covalent bonds between the sugar of one nucleotide and the phosphate of the next. The nucleotides of the two chains are bound together according to base pairing rules (A with T and C with G) with hydrogen bonds to make the double-stranded DNA.

Both strands of the double-stranded DNA store the same biological information; this information is replicated as and when the two strands separate. A large part of the DNA (more than 98% for humans) is non-coding, meaning that these sections, probably, do not serve as patterns for protein sequences.

The two strands of DNA run in opposite directions to each other and are thus antiparallel. It is the sequence of these four bases along the backbone that encodes genetic information. RNA strands are created using DNA strands as a template in a process called transcription, where DNA bases are exchanged for their corresponding bases except in the case of thymine (T), which RNA substitutes for uracil (U). Under the genetic code, these RNA strands specify the sequence of amino acids within proteins in a process called translation.

Within eukaryotic cells, DNA is organized into long structures called chromosomes. Before cell division, these chromosomes are duplicated in the process of DNA replication, providing a complete set of chromosomes for each daughter cell. Chromatin proteins, such as histones, compact and organize DNA to better control which parts of the DNA are transcribed.

The set of chromosomes in a cell makes up its genome; the human genome has approximately 3 billion base pairs of DNAs arranged into 46 chromosomes. The information carried by DNA is held in the sequence of pieces of DNA called genes. Transmission of genetic information in genes is achieved via complementary base pairing. For example, in transcription, when a cell uses the information in a gene, the DNA sequence is copied into a complementary RNA. This RNA copy is then used to make a matching protein sequence in a process called translation.

To sum up we can say that genes, through the RNA, give instructions and proteins carry out these instructions to do a specific job.

Each type of protein is a specialist that only does one job, so if a cell needs to do something new, it must make a new protein to do this job. Similarly, if a cell needs to do something faster or slower than before, it makes more or less of the protein responsible of that job. Genes tell cells what to do by telling them which proteins to make and in what amounts. The structure of a gene consists of many elements of which the actual protein coding sequence is often a small part. These include DNA regions that are not transcribed as well as untranslated regions of the RNA.

Flanking the open reading frame, genes contain regulatory sequence that is required for their expression. First, genes require a promoter sequence. The promoter recognized and bound by transcription factors that recruit and help RNA polymerase bind to the region to initiate transcription. The recognition typically occurs as a consensus sequence like the TATA box. A gene can have more than one promoter, highly transcribed genes have "strong" promoter sequences that form strong associations with transcription factors, thereby initiating transcription at a high rate. Other genes have "weak" promoters that form weak associations with transcription factors and initiate transcription less frequently.

In all organisms, two steps are required to read the information encoded in a gene's DNA and produce the protein it specifies. First, the gene's DNA is transcribed to messenger RNA (mRNA). Second, that mRNA is translated to protein. RNA-coding genes must still go through the first step but are not translated into protein. The process of producing a biologically functional molecule of either RNA or protein is called gene expression, and the resulting molecule is called a gene product.

The nucleotide sequence of a gene's DNA specifies the amino acid sequence of a protein through the genetic code. Sets of three nucleotides, known as codons, each correspond to a specific amino acid.

Additionally, a "start codon", and three "stop codon" indicate the beginning and end of the protein coding region. There are 64 possible codons (four possible nucleotides at each of three positions, hence 4^3 possible codons) and only 20 standard amino acids; hence the code is redundant and multiple codons can specify the same amino acid. The correspondence between codons and amino acids is nearly universal among all known living organisms.

Transcription produces a single-stranded RNA molecule known as messenger RNA; whose nucleotide sequence is complementary to the DNA from which it was transcribed. The mRNA acts as an intermediate between the DNA gene and its final protein product. The gene's DNA is used as a template to generate a complementary mRNA. The mRNA matches the sequence of the gene's DNA coding strand because it is synthesized as the complement of the template strand.
Transcription is performed by an enzyme called an RNA polymerase, which reads the template strand in the $3'$ to $5'$ direction and synthesizes the RNA from $5'$ to $3'$. To initiate transcription, the polymerase first recognizes and binds a promoter region of the gene. Thus, a major mechanism of gene regulation is the blocking or sequestering the promoter region, either by tight binding by repressor molecules that physically block the polymerase, or by organizing the DNA so that the promoter region is not accessible. In eukaryotes, transcription occurs in the nucleus, where the cell's DNA is stored. The RNA molecule produced by the polymerase is known as the primary transcript and undergoes post-transcriptional modifications before being exported to the cytoplasm for translation. One of the modifications performed is the splicing of introns which are sequences in the transcribed region that do not encode protein.

Translation is the process by which a mature mRNA molecule is used as a template for synthesizing a new protein. Translation is carried out by ribosomes, large complexes of RNA and protein responsible for carrying out the chemical reactions to add new amino acids to a growing polypeptide chain by the formation of peptide bonds. The genetic code is read three nucleotides at a time, in units called codons, via interactions with specialized RNA molecules called transfer RNA (tRNA). Each tRNA has three unpaired bases known as the anticodon that are complementary to the codon it reads on the mRNA. The tRNA is also covalently attached to the amino acid specified by the complementary codon. When the tRNA binds to its complementary codon in an mRNA strand, the ribosome attaches its amino acid cargo to the new polypeptide chain. During and after synthesis, most new proteins must fold to their active three-dimensional structure before they can carry out their cellular functions.

Genes are regulated so that they are expressed only when the product is needed, since expression draws on limited resources. A cell regulates its gene expression depending on its external environment, its internal environment, and its specific. Gene expression can be regulated in many ways: from transcriptional initiation, to RNA processing, to post-translational modifications of the protein and through biological pathways.

A biological pathway is a series of actions among molecules in a cell that leads to a certain product or a change in the cell. It can trigger the assembly of new molecules, such as a fat or protein, turn genes on and off, or spur a cell to move.

These biological pathways control a person's response to the world. For example, some pathways subtly affect how the body processes drugs, while others play a major role in how a fertilized egg develops into a baby. Other pathways maintain balance while a person is walking, control how and when the pupil in the eye opens or closes in response to light and affect the skin's reaction to changing temperature.

There are many types of biological pathways. Among the most well-known are pathways involved in metabolism, in the regula-

tion of genes and in the transmission of signals.
Metabolic pathways make possible the chemical reactions that occur in our bodies. An example of a metabolic pathway is the process by which cells break down food into energy molecules that can be stored for later use.
Gene-regulation pathways turn genes on and off.
Signal transduction pathways move a signal from a cell's exterior to its interior. Different cells are able to receive specific signals through structures on their surface called receptors. After interacting with these receptors, the signal travels into the cell, where its message is transmitted by specialized proteins that trigger a specific reaction in the cell. For example, a chemical signal from outside the cell might direct the cell to produce a particular protein inside the cell. In turn, that protein may be a signal that prompts the cell to move.

1.2 Shotgun Sequencing

DNA sequencing is the process of determining the nucleotide order of a given DNA fragment.
In shotgun sequencing, DNA is broken up randomly into numerous small segments, which are sequenced using the chain termination method to obtain reads. Multiple overlapping reads for the target DNA are obtained by performing several rounds of this fragmentation and sequencing. Computer programs then use the overlapping ends of different reads to assemble them into a continuous sequence. One important thing to take into account in the sequencing process is the Sequencing Depth, this term refers to the number of unique reads that include a given nucleotide in the reconstructed sequence. Deeper is the sequencing higher is the number of unique reads of each region of a sequence [1].
Nowadays shotgun techniques are still applied, but they have improved a lot over the years. Indeed, the shotgun strategy currently has new sequencing technologies: the next-generation sequencing ones.
These technologies produce shorter reads but many hundreds of thousands (even millions) of reads in a relatively short time. This results in high coverage, but the assembly process is much more computationally intensive. These technologies are vastly superior to Sanger sequencing due to the high volume of data and the relatively short time it takes to sequence a whole genome [2].
In what follows, some examples of next-generation sequencing methods are described.

Illumina (Solexa) sequencing

The Illumina sequencing technology follows three basic steps: amplifying, sequencing, and analyzing. The process begins with purified DNA. The DNA gets chopped up into smaller pieces and given adapters, indices, and other kinds of molecular modifications that

act as reference points during amplification, sequencing, and analysis [4].
The modified DNA is loaded onto a specialized chip where amplification and sequencing will take place. Along the bottom of the chip are hundreds of thousands of oligonucleotides (short, synthetic pieces of DNA). They are anchored to the chip and able to grab DNA fragments that have complementary sequences. Once the fragments have attached, a phase called cluster generation begins. This step makes about a thousand copies of each fragment of DNA. Next, primers and modified nucleotides enter the chip.
These nucleotides have reversible 3' blockers that force the polymerase to add on only one nucleotide at a time as well as fluorescent tags. After each round of synthesis, a camera takes a picture of the chip.
A computer determines what base was added by the wavelength of the fluorescent tag and records it for every spot on the chip. After each round, non-incorporated molecules are washed away. A chemical deblocking step is then used in the removal of the $3'$ terminal blocking group and the dye in a single step. The process continues until the full DNA molecule is sequenced. With this technology, thousands of places throughout the genome are sequenced at once via massive parallel sequencing.

Ion Torrent semiconductor sequencing

This method of sequencing is based on the detection of hydrogen ions, which are released during the polymerization of DNA, as opposed to the optical methods used in other sequencing systems [5].
A microwell containing a template DNA strand to be sequenced is flooded with a single type of nucleotide. If the introduced nucleotide is complementary to the leading template nucleotide it is

incorporated into the growing complementary strand.
This causes the release of a hydrogen ion that triggers a hypersensitive ion sensor, which indicates that a reaction has occurred. If homopolymer repeats are present in the template sequence, multiple nucleotides will be incorporated in a single cycle. This leads to a corresponding number of released hydrogens and a proportionally higher electronic signal.
The major benefits of ion semiconductor sequencing are rapid sequencing speed and low upfront and operating costs. This has been enabled by the avoidance of modified nucleotides and optical measurements.

Even more sophisticated sequencing methods has been implemented in the last years, the so called Third-generation sequencing methods. An example of these method is:

Nanopore DNA sequencing

Nanopore sequencing is a third-generation approach used in the sequencing of polynucleotides in the form of DNA or RNA. Using nanopore sequencing, a single molecule of DNA or RNA can be sequenced without the need for PCR amplification or chemical labeling of the sample [3].
Nanopore sequencing uses electrophoresis to transport an unknown sample through an orifice of 10^{-9} meters in diameter.
A nanopore system always contains an electrolytic solution when a constant electric field is applied, an electric current can be observed in the system. The magnitude of the electric current density across a nanopore surface depends on the nanopore's dimensions and the composition of DNA or RNA that is occupying the nanopore.
Sequencing is made possible because, when close enough to nanopores, samples cause characteristic changes in electric current density across nanopore surfaces.

1.3 Metagenomics

Metagenomics is the study of microbes in their natural living environment, which involves the complex microbial communities in which they usually exist. Metagenomics examines the genomic composition of an entire organism, including each of the microbes that exist within it. It is an important concept for the microbes and the host to be thought of as interdependent and observed as a community, rather than considered to be separate entities.
In the past, these microbes were only cultivated *in vitro*. On the other hand, lately, thanks to the spreading of metagenomics, the scientific community was able to deepen its research and observe not-known-microbes.
Therefore, this opens up a new playing field in the metagenomics research. If we were able to deepen our understanding of the microbial communities that regularly interact with humans (such as those that reside inside the gastrointestinal tract) we could shed some light on the interplay between microbes and human health. Microbes may encode metabolic pathways, which are essential for the survival of humans while being absent in the human genome [7].
Indeed, changes in the symbiotic microbial levels can be associated with several health conditions such as inflammatory bowel disease (IBD), cardiovascular disease, eczema or cancer [6].
The microbes responsible for these changes are not likely to be the type of microbes that directly cause illness or infection. Instead, the more likely explanation is that the microbes and the human body ordinarily work together to successfully digest food, remove toxins from the body and prevent infection from taking over the body. If there is a change in the function of the microbe that is so closely involved with human health, this may translate to the presentation of a health condition in the affected human.

Chapter 2

The Dataset

The dataset is composed by abundance values of biological pathways of three kind of subjects. One of these in healthy state and two affected by Inflammatory Bowel Diseases, namely: Ulcerative Colitis and Crohn's Disease [8, 9].

2.1 The Diseases

The umbrella term "Inflammatory bowel diseases" (IBD) includes a spectrum of chronic inflammatory disorders which recurrently affect the gastro-intestinal tract. Ulcerative colitis (UC) and Crohn's disease (CD) are the two main clinically defined manifestations of IBD, each with distinctive clinical and pathological features.

Ulcerative Colitis

Ulcerative Colitis (UC) is a long-term condition that results in inflammation and ulcers of the colon and rectum. The primary symptoms of active disease are abdominal pain and diarrhea mixed with blood. Weight loss, fever, and anemia may also occur. Often, symptoms come on slowly and can range from mild to severe. Symptoms typically occur intermittently with periods of no symptoms between flares. Complications may include inflammation of eye, joints, or liver and colon cancer. The cause is, nowadays, unknown but some new theories involve immune system dysfunction, genetics, changes in the normal gut bacteria and environmental factors.

Crohn's Disease

Crohn's Disease is an inflammatory disease that may affect any segment of the gastrointestinal tract from the mouth to the rectum. Symptoms are very similar with the Ulcerative Colitis and can include fever, diarrhea, abdominal pain and weight loss. Other complications outside the gastrointestinal tract may include anemia, skin rashes, arthritis, inflammation of the eye, and tiredness. Bowel obstruction may occur as a complication of chronic inflammation, and those with this disease are at greater risk of bowel cancer.
The cause of Crohn's disease it is believed to be due to a combination of environmental, immune, and bacterial factors in genetically susceptible individuals. It results in a chronic inflammatory disorder, in which the body's immune system attacks the gastrointestinal tract.

These two Inflammatory Bowel Diseases may appear very similar, but they can be distinguished by two main differences:

- Location: Ulcerative Colitis affects only the large intestine while in Crohn's Disease, inflammation can appear anywhere in the digestive tract, from mouth to the anus.
- Continuous Inflammation: People with Crohn's Disease often have healthy areas in between inflamed spots. But with Ulcerative Colitis, there are no healthy areas in between inflamed spots.

Currently, endoscopy constitutes the gold standard for the diagnosis and the monitoring of IBD. Its diagnosis is usually confirmed by biopsies on colonoscopy and complemented with measurement of molecular biomarkers including fecal calprotectin, serum C-reactive protein (CRP), and serum antibody markers including autoantibodies and microbial and peptide antibodies. However, their low sensitivity and high variability limit the clinical efficacy. Thus, there is a need to identify novel molecular biomarkers that could be assessed with less invasive methods and could benefit IBD clinical management and treatment.

Concerning the worldwide prevalence and incidence of these diseases, a systematic review of population-based studies from 1990 to 2016 provides some interesting data. Europe and North America show the highest prevalence values worldwide, e.g., 505 and 286 UC subjects per 100000 in Norway and USA, respectively and 322 and 319 CD subjects per 100000 in Germany and Canada, respectively. We can say that IBD has become a global disease with accelerating incidence in newly industrialized countries.

2.2 Structure

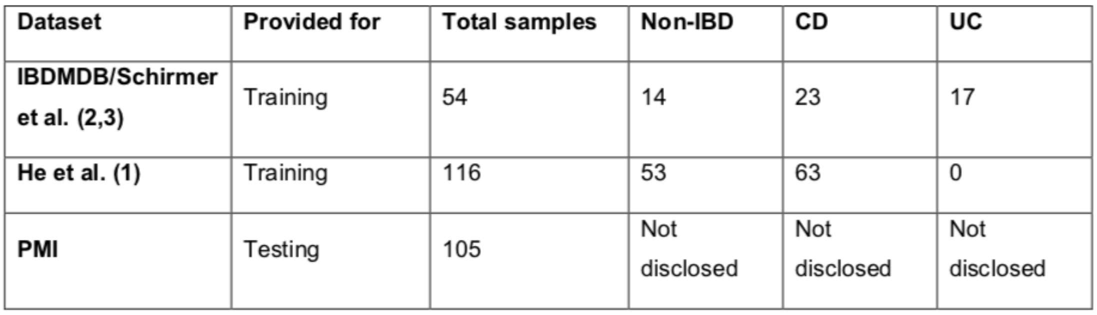
The dataset is composed by metagenomics data from two published human studies as training datasets, and from one internal human study as test dataset (called PMI).
The two published human studies used as training set are respectively:

IBDMDB/Schirmer et al. dataset: This first training dataset includes paired-end whole genome sequencing reads from a publicly available longitudinal study conducted in North America as a part of the second edition of the human microbiome project, namely the integrative Human Microbiome Project [9]. It contains a total of 1338 paired fastq files including raw data obtained from fecal samples of adults and children collected at multiple time points. Following quality checking, and after selecting one sample per adult subject, the dataset provided for training includes a total of 54 samples from 23 CD, 17 UC and 14 non-IBD subjects.

He et al. dataset: The second training dataset comprises paired-end whole genome sequencing reads from a publicly- available cross-sectional study conducted in China. It contains a total of 123 paired fastq files including raw data obtained from fecal samples of adults. Following quality checking, the dataset provided for training included a total of 116 samples from 63 CD and 53 non-IBD subjects [8].

Regarding the quality check mentioned before the sample selection was based on the sample metadata. One common criterion for all samples selected for this project was the age of the subjects to whom the sample belong as adult subjects (age>=18). In addition, for the IBDMDB/Schirmer dataset, only the earliest time-point per subject has been kept, and for the He et al. dataset has been included only the subjects that are not marked as "host contamined" in the original research metadata.

PMI: The internal dataset used as test set consists of 105 paired-end whole genome sequencing of fecal samples from CD, UC and non-IBD. No other information has been published.



Dataset	Provided for	Total samples	Non-IBD	CD	UC
IBDMDB/Schirmer et al. (2,3)	Training	54	14	23	17
He et al. (1)	Training	116	53	63	0
PMI	Testing	105	Not disclosed	Not disclosed	Not disclosed

Figure 2.1: Summary of Challenge's datasets

The data matrices used for the data analysis and classification was

the Pathway abundances matrices. These matrices, for all three datasets, were generated using the Biobakery's pipeline starting from the raw reads, using default settings and reference databases, except the 16S database that was generated using a text search for ”16S” in the NCBI nucleotide database, selecting all sequences which belong to ”Fungi”, ”Protists”, ”Bacteria”, ”Archaea” and ”Viruses”, with a range of length between 700 and 2000 base pairs and storing those sequences into a fasta file.
More specifically, the HUMAnN2 component of the Biobakery pipeline [14] computed pathway abundances for each sample by associating reads with MetaCyc reaction pathways, stratified where possible by species. So, in the pathway abundance matrix there are two levels of pathways:

- The highest level known as Community pathways
- The deepest level known as Species pathways

Pathway abundance files generated for each sample using the Biobakery pipeline were joined into a single matrix with the sample identification numbers as column names and the unique pathways identification number as row names, as illustrated. When pathway abundance was missing for a sample, the pathway abundance value was set to 0.

organization.png

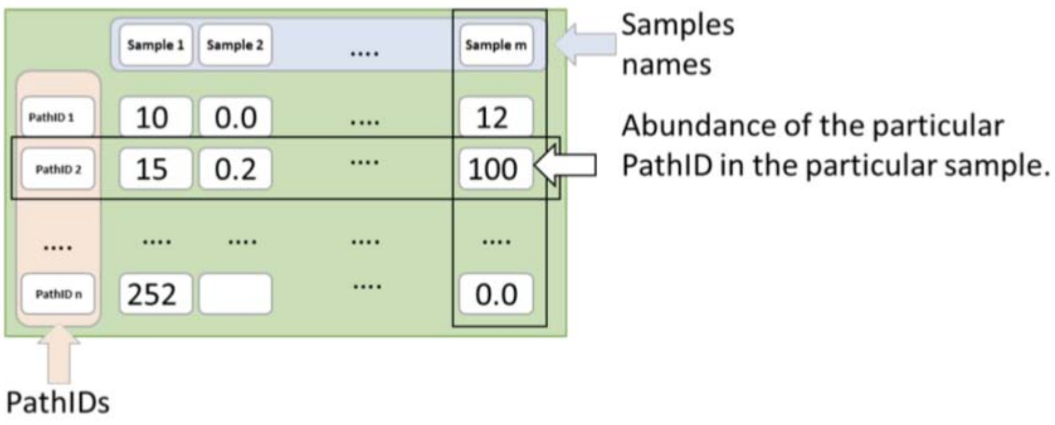


Figure 2.2: Dataset organization

In addition to the pathway matrices, they also provide a "PathID description" file that contains the full pathway information associated with each PathID as shown in the next figure.

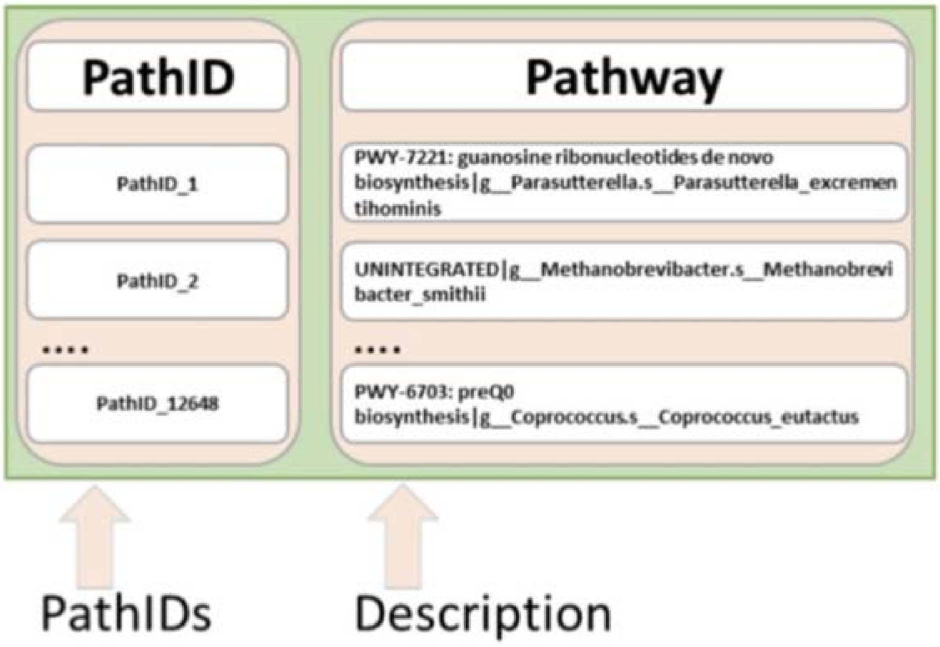

PathID
Pathway
PathID_1
PWY-7221: guanosine ribonucleotides de novo biosynthesis|g__Parasutterella.s__Parasutterella_excrementihominis
PathID_2
UNINTEGRATED|g__Methanobrevibacter.s__Methanobrevibacter_smithii
....
....
PathID_12648
PWY-6703: preQ0 biosynthesis|g__Coprococcus.s__Coprococcus_eutactus
PathIDs
Description

Figure 2.3: PathIDs matrix organization

Finally, Class labels associated with each selected sample from subjects diagnosed with UC, CD and non-IBD are provided for both training datasets by the Challenge organizers.

2.3 IMPROVER Challenge

This thesis is based on the International Challenge called SBV Improver project funded by PMI Research and Development. This year's challenge is called *MEDIC: Metagenomics Diagnosis for Inflammatory Bowel Disease Challenge.*
The aim of the challenge is to investigate the diagnostic potential of metagenomics data to classify patients with Inflammatory Bowel Disease (IBD) and non-IBD subjects. Furthermore, within the IBD category, the participants have to attempt to classify

Ulcerative Colitis (UC) and Crohn's Disease (CD) subjects.

In literature many studies demonstrate that IBD comprises complex genetic disorders, with multiple contributing genes. However, not all subjects carrying mutations in those identified genes develop IBD. Indeed, other components, such as diet and microbiota, seem to play a role in the etiology of the disease.
The human microbiome composed of various microorganisms colonize different body sites, such as the gut, mouth, genitals, skin, and airways, and vary in compositions. The microbiome is recognized to play a positive role in host supporting the maintenance of homeostasis, by contributing in the metabolism of nutrients, detoxification, helping immunity, preventing the propagation of pathogenic microbes for examples. A balanced interaction of microbes with the host plays an important part in preserving health. The dynamics and function of the microbiota can be influenced by many host-related and environmental factors, such as age, gender, diet, and drugs. Dysbiosis, a disruption of this balance, is associated with skin and neurological disorders as well as many diseases such as immune-related diseases, metabolic diseases, inflammatory bowel disease.

The link between pathogenesis of IBD and the intestinal microbiota has been established in animal models of colitis showing that germ-free conditions prevent inflammation and in human studies, showing that probiotics or surgical diversion of the fecal stream help the management of IBD and improve inflammation. Evidence also points out that microbiome dysbiosis may cause an inappropriate immune response that results in alteration of the intestinal epithelium barrier integrity. An increase of epithelial permeability allows further infiltration of microbial organisms that, in turn, provoke further immune responses.

The characterization of the microbiome relies on 16S or shotgun sequencing of metagenomes from fecal or intestine biopsy samples. Recent studies investigated microbiome changes in CD and/or UC

compared to non-IBD using metagenomics sequencing data, and reported differences in composition and abundances between subjects suffering from IBD compared with non-IBD subjects.
The pathway abundances matrix constitutes the starting point for machine learning and identification of discriminative metagenomics features and model predictive of IBD status. This Challenge aims to explore this new avenue.
More in the specific what the organizers wants is to build up four binary classifiers to discriminate between healthy states in these ways:

- IBD vs non-IBD
- UC vs non-IBD
- CD vs non-IBD
- UC vs CD

Moreover, we also had to provide a confidence value in the interval [0,1], with 1 being the highest confidence that the sample belongs to "class 1".

To do this I can briefly sum up the pipeline in 3 different steps:

- Data Preprocessing and Visualization
  In this step, I examine the nature of the data in the pathway matrix to better understand its abundance unit of measure and choose the better normalization technique. After this step, I apply 3 different approaches of data visualization to see how good the classes divide between each other.

- Feature Selection
  The large number of PathID in the pathway abundance matrix (around 12 thousands) made impossible to make a classifier using all of them so, in this step, I use different algorithms, known in literature, to perform a feature selection on the entire matrix to search for the most informative one.

- Classification
  Knowing the most informative features chosen by the different methods I trained an SVM classifier for every binary classification required for the Challenge.

In the next chapters I'll describe these 3 steps more in depth.

Chapter 3

Data Preprocessing and Visualization

3.1 From RAW data to OTU table: HumanN2 tool

A first important step, before the data analysis of the matrix, is to understand how they moved from RAW data format to OTU tables in order to have the knowledge of how this data are structured.
The tool used to do so is HumanN2 [16]: this tool determines the presence, absence and abundance of metabolic pathways in a microbial community from metagenomic sequencing data.
In the following we can see the computational pipeline implemented in HumanN2, through a process of seven steps:

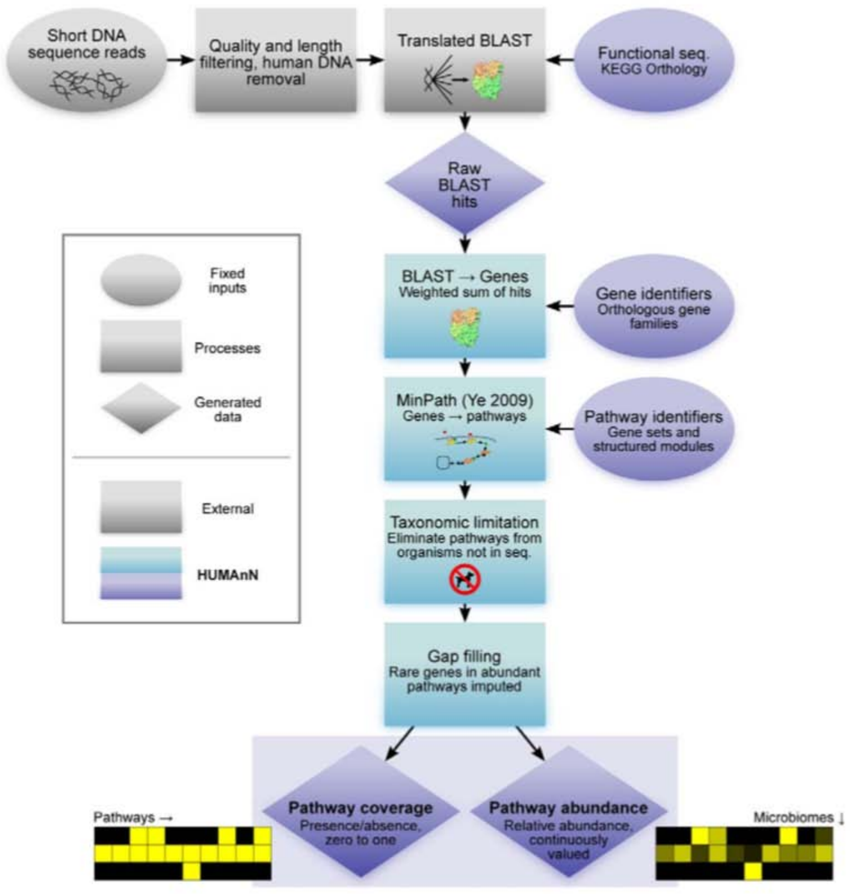

Short DNA sequence reads
Quality and length filtering, human DNA removal
Translated BLAST
Functional seq. KEGG Orthology
Raw BLAST hits
BLAST → Genes Weighted sum of hits
Gene identifiers Orthologous gene families
MinPath (Ye 2009) Genes → pathways
Pathway identifiers Gene sets and structured modules
Taxonomic limitation Eliminate pathways from organisms not in seq.
Gap filling Rare genes in abundant pathways imputed
Pathway coverage Presence/absence, zero to one
Pathway abundance Relative abundance, continuously valued
Pathways →
Microbiomes ↓
Fixed inputs
Processes
Generated data
External
HUMAnN

Figure 3.1: PathIDs matrix organization

1. Short reads are sequenced from a community sample, quality and length filtered, and screened for residual host (human) DNA. This process is carried out by the organizers of the Challenge, externally to HumanN2.

2. Reads are searched against a characterized protein sequence database. HumanN2 can operate using results from several standard or accelerated BLAST implementation and from different orthologous protein family catalogs; the organizers employed MetaCyc and KEGG Orthology.

3. For each metagenomic sample, HumanN2 recovers the abundance of individual orthologous gene families by counting its reads' BLAST hits in a weighted manner, normalized by each gene family's average sequence length.
4. Genes are assigned to pathways using MinPath, a maximum parsimony approach to explaining observed genes with available pathways [17].
5. Pathways unlikely to be present based on the BLAST hits' approximate organismal profile are removed in a taxonomic limitation step, which also allows normalization for genes' average copy number.
6. A biological smoothing or gap filling step is performed, preventing small numbers of apparently absent genes in an otherwise abundant pathway from diminishing its presence due to noise.
7. Finally, HumanN2 assigns each pathway a coverage (presence/absence) score in each sample based on the detection of all its constituent genes, as well as an abundance score indicating its relative abundance in the sample's metagenome.

3.2 Gene and Pathway abundance concepts and calculus

Before choosing the normalizing method to apply to the data matrix we need to understand more in depth how HumanN2 calculate genes, and consequently, pathways abundance [15]. To do so, we will focus on the last 5 steps:

Orthologous gene family Abundances

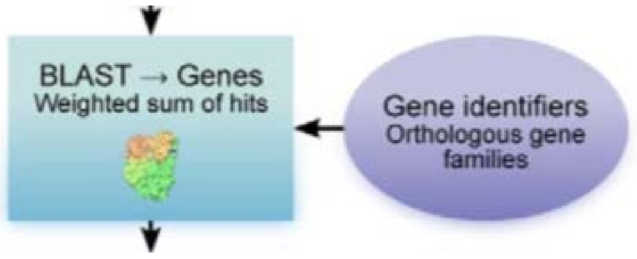


Figure 3.2: gene abundance

In this step HumanN summarizes the BLAST results, as the number of reads that matched each protein family, weighted by the quality of the matches. To do so, they used KEGG Orthology gene families (KOs), a catalog of organisms-independent identifiers corresponding to groups of gene sequences carrying out comparable biochemical functions. KO i consists of a set of one or more specific gene sequences $G_i = \{g_{i,1}, g_{i,2}, ...\}$ from individual organisms annotated in KEGG.
Orthologous family abundance w_i were calculated independently within each metagenome for KO i and read j as:

$$w_i = |G_i| \sum_{g \in G_i} \frac{1}{|g|} \sum_j \frac{1 - p_{g,j}}{\sum g' 1 - p_{g',j} (3.1)}$$

Where $|g|$ is the nucleotide length of gene sequence g in KO i. $|G_i|$ is the number of such sequences, and p_{ij} is the p-value of the BLAST hit of read j to sequence g. That is, the relative abundance of KO i in a metagenome is the number of reads j that map to a gene sequence in the family, weighted by the inverse p-value of each mapping and normalized by the average length of all gene sequences in the orthologous family.

Assign gene families to pathways

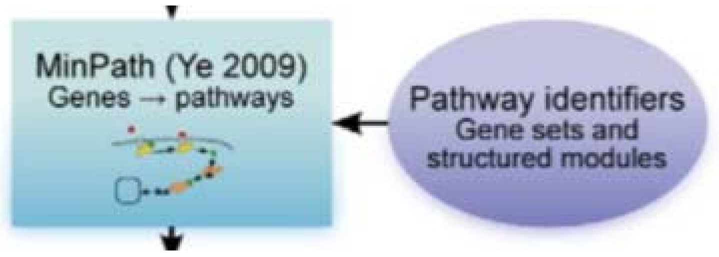


Figure 3.3: Pathways assigning

In this step KOs are consolidated into one or more pathways using MinPath. MinPath defines each pathway as an unstructured gene set and selects the fewest pathways that can explain the genes observed within each community.
More specifically, HumanN associates each KO family i with a vector of relative abundances $w = [w_{i1}, w_{i2}, ...]$ in each metagenome. KOs were then assigned to zero or more pathways using MinPath v1.2. KOs assigned to two or more pathways are effectively duplicated and their abundance included in each; this results in two independent vectors of abundance tuples of the form (KO, pathway ID) for each metagenome.
$wp = [w_{j1,p1}, w_{i1,p2}, w_{i2,p1}, w_{i2,p2}, ...]$, where $w_{i,p} = w_j$ for all pathways p.

Filtering Pathways by taxonomic limitation

Taxonomic limitation
Eliminate pathways from organisms not in seq.

Figure 3.4: Pathway filtering step

HumanN2 employs an additional pathway filter step to be useful in removing false positive pathways selected by MinPath.
Specifically, by retaining a very approximate organismal abundance profile of gene families hit during the initial BLAST process, HumanN is able to remove pathways in gross disagreement with observed taxa in an unsupervised manner.
Specifically, taxonomic limitation is performed by removing only (KO, ID) tuples for which the same KO was assigned to multiple pathways. For each sample, approximate abundances for each organism o in KEGG were calculated as a sum over all weighted, normalized BLAST hits to sequences from that organism:

$$w_o = |G_o| \sum_{g \in G_o} \frac{1}{|g|} \sum_j \frac{1 - p_{g,j}}{\sum g' 1 - p_{g',j}(3.2)}$$

Each pathway was then assigned an approximate expected relative abundance by summing w_o values over all organisms' genomes in which it was annotated.
Finally, any (KO, ID) pair with two or more IDs and corresponding to a pathway with observed relative abundance below the average expected abundance for that ID was removed. That is, for $\delta_{o,p} = 1$ if pathway p was annotated to organism o in KEGG and 0 otherwise.

Smoothing by gap filling

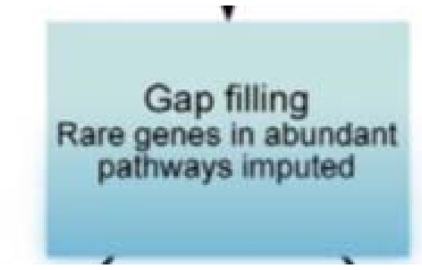


Figure 3.5: Gap filling step

Taxonomic limitation was used by HumanN to reduce false positive pathways, they found a small degree of replacement or gap filling of certain missing genes to likewise reduce false negatives. A small number of low abundance genes within otherwise abundant pathways often occurred due to noise or poor BLAST hits. Biological gap filling was added to increase the effective contribution of unobserved members of otherwise abundant pathways.
Within each retained pathway ID, KOs with relative abundance 1.5 inter-quartile ranges below the pathway median were boosted to an effective abundance equal to median. That is, for all pathways p such that there existed some $w_{i,p} > 0$, let $\tilde{w_{i,p}}$ be the lower inner fence of $w_{i,p}$ over all $i \in p$, and each $w_{i,p}$ for $i \in p$ was set to $max(w_{i,p}, \tilde{w_{i,p}})$.

Smoothing by gap filling

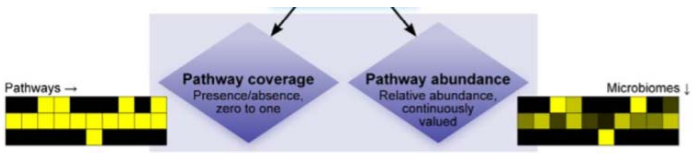

Pathways →
Pathway coverage
Presence/absence, zero to one
Pathway abundance
Relative abundance, continuously valued
Microbiomes ↓

Figure 3.6: Coverage and abundance step

The final outputs for each sample were thus coverage (presence/absence) and abundance values for KEGG pathways.
Coverage is calculated to indicate the likelihood that all genes needed to operate the pathway are present; Abundance is calculated as the average copy number of the pathway's operational subset.
Given the vector w_p, coverage for each pathway p in a sample was calculated as the fraction of KOs in the pathway that were confidently present, specifically with abundance greater than the overall sample median. That is:

$$cov_p = \frac{2}{|p|} \sum_{i \in p} \partial\{w_{i,p} > \tilde{w_{i,p}}\} \tag{3.3}$$

Pathway abundance was calculated as the average of the upper half of its individual gene abundances, that is:

$$abd_p = \frac{2}{|p|} \sum_{i \in [p/2]} w_{i,p} \tag{3.4}$$

for [p/2] the most abundant half of $w_{i,p}$

3.3 CPM normalization

Normalization is a necessary step before data analysis to make accurate comparisons between samples. Normalization is usually

applied when the data matrix deals with gene expression and is the process of scaling raw count values to remove some ”uninteresting” factors. In this way the expression levels are more comparable between and/or within samples.
The main factors often considered during normalization are:

- Sequencing depth:
  Accounting for sequencing depth is necessary for comparison of gene expression between samples. In the example below, each gene appears to have doubled in expression in sample A relative to sample B, however this is a consequence of sample A having double the sequencing depth.

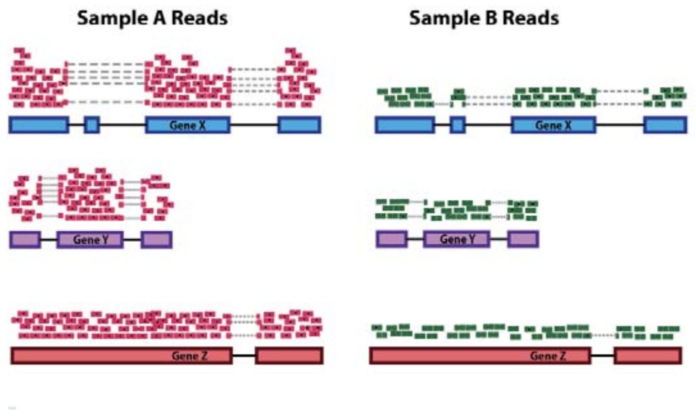


Figure 3.7: sequencing depth problem

- Gene length:
  Accounting for gene length is necessary for comparing expression between different genes within the same sample. In the example, gene X and gene Y have similar levels of expression, but the number of reads mapped to gene X would be many more than the number mapped to gene Y because gene X is longer[18].

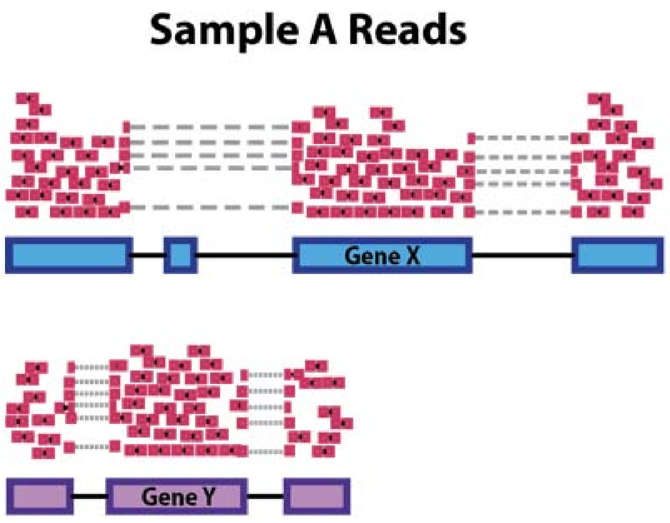


Figure 3.8: gene length problem

In the Challenge case we don not deal with gene expression directly, but from a data type derived from a gene expression.
Because of the pathway abundance calculation aforementioned, if we want to compare a specific pathway abundance between two sample, we have to deal with a size factor derived from the sequencing depth in the gene abundance calculus.
A simple but effective normalization methods to overcome this problem is the CPM normalization.

In the following a brief 3 step description of this technique:

- Sum the entire abundance values of a specific sample
- Divide every abundance value of that sample by the sum of the abundances and multiply by 10^6
- Apply the first two steps for every sample

To apply this method, I used an R script in which is applied the *cpm* function by *edgeR v3.14.0*.

From now on, if not differently specified, every time Iâll mention the pathway abundance matrices, Iâll consider their normalized versions.

In the next sections Iâll examine 3 different ordination/visualization methods to see how the IBD, UC and CD classes divides in a 2-dimensional space.
These techniques can be seen as multivariate methods which summarize a multidimensional dataset in such a way that, when it is projected onto a low dimensional space, any intrinsic pattern the data may possess becomes apparent upon visual inspection.

3.4 PCA: concept and plot comparisons

PCA uses a rotation of the original axes to derive new axes, which maximize the variance in the dataset. Computationally, PCA is an eigen analysis and the most important consequences are:

- There is a unique solution to the eigen analysis
- The axes (also called principal components or PC) are orthogonal to each other, and thus independent
- Each PC is associated with an eigenvalue. The sum of the eigenvalues will equal the sum of the variance of all variables in the dataset. The eigenvalues represent the variance extracted by each PC and are expressed as a percentage of the sum of all eigenvalues. The relative eigenvalues thus tell how much variation that PC is able to ”explain”
- Axes are ranked by their eigenvalues. Thus, the first axis has the highest eigenvalues and thus explains the most variance, the second axis has the second highest eigenvalues, etc...

To implement PCA in R there are many packages but, in this case, the high dimension of the data matrix made necessary to implement the method in a more computationally efficient way.

```
PCA_SVD <- function(sample) {
  Z<-t(t(sample)-apply(sample,2,mean))
  SVdec<-svd(Z) # singular value decomposition
  d<-SVdec$d  # diagonal of d
  varperc<-rep(0,length(d))
  for (i in (1:length(d))) varperc[i]<-sum((d^2)[1:i])/sum(d^2)
  # compute % of variance explained
  PCs_SVD<-SVdec$v # Principal Components
  Y_SVD<-sample%*%PCs_SVD # projection of the N samples
  # in the new coords along the PCs
  return(Y_SVD)
}
```

To see how good the classes divide in a two-dimensional space and if the normalization process leads to some improvement in the two-class division, I made some PCA plots comparisons between the data before and after normalization. In the following we can see some examples of the plots I did.

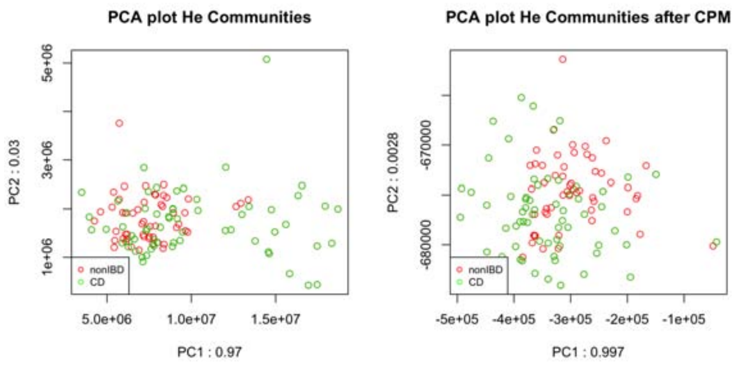


Figure 3.9: comparison of pathways at community level in He et al. dataset pre and post CPM normalization

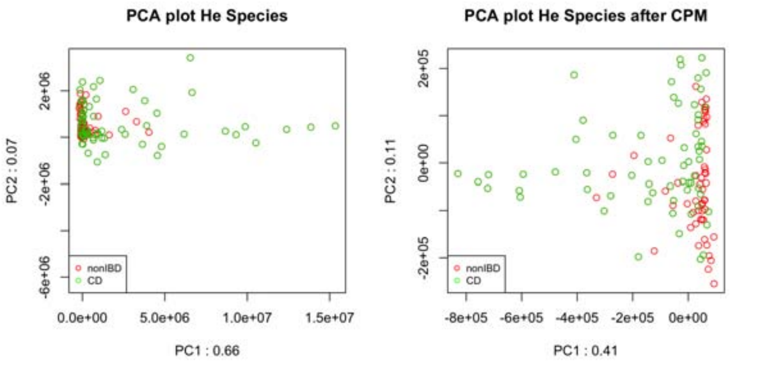


Figure 3.10: comparison of pathways at species level in He et al. dataset pre and post CPM normalization

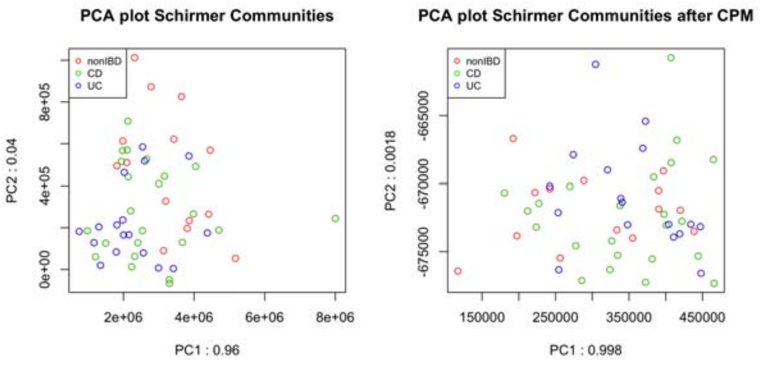


Figure 3.11: comparison of pathways at community level in Schirmer et al. dataset pre and post CPM normalization

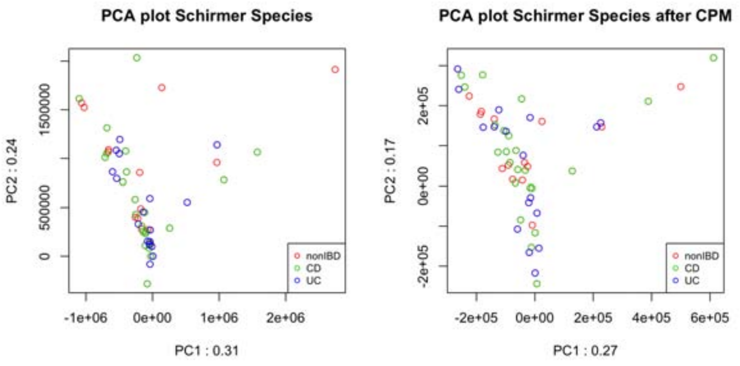


Figure 3.12: comparison of pathways at species level in Schirmer et al. dataset pre and post CPM normalization

3.5 PCoA: concept and plot comparison

Principal coordinate analysis attempts to represent the distances between samples in a low-dimensional, Euclidean space. In particular, it maximizes the linear correlation between the distances in the distance matrix, and the distances in a space of low dimension (typically 2 axes are selected).
The first step of a PCoA is the construction of a dissimilarity matrix. While PCA is based on Euclidean distances, PCoA can handle dissimilarity matrices calculated from quantitative, semi-quantitative, qualitative, and mixed variables. As always, the choice of dissimilarity measure is critical and must be suitable to the data in question. In this case, since we are dealing with abundance data, Bray-Curtis distance is often recommended in literature.
When the distance metric is Euclidean, PCoA is equivalent to Principal Components Analysis.

To apply this method in R I used two packages: *vegan v2.4-2* and *ape v5.3*.

```
PCoA <- function(sample) {
```

```
dist=vegdist(sample, method = ''bray'') # compute the dissimilarity
# matrix
PCOA=pcoa(dist) # compute PCoA
PC<- PCOA$vectors[,c(1,2)] # take only the first two components
return(PC)
}
```

In the following graphs we can see some comparisons using this technique pre and post normalization.

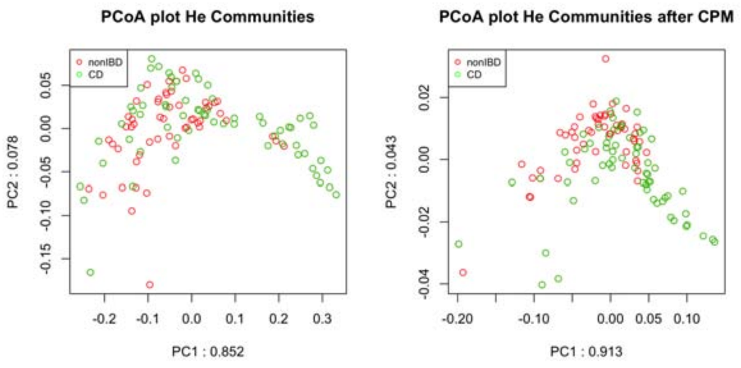


Figure 3.13: comparison of pathways at community level in He et al. dataset pre and post CPM normalization

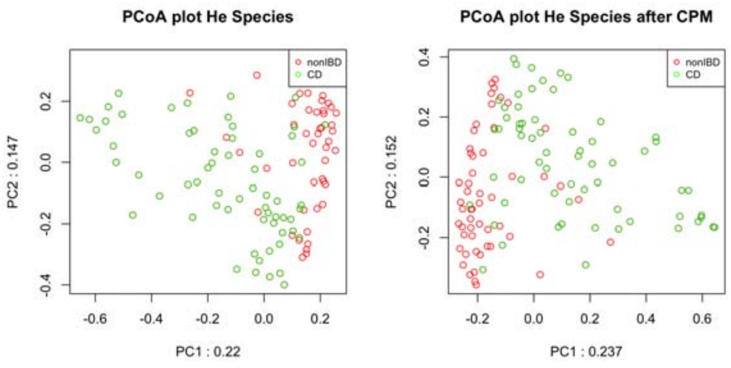


Figure 3.14: comparison of pathways at species level in He et al. dataset pre and post CPM normalization

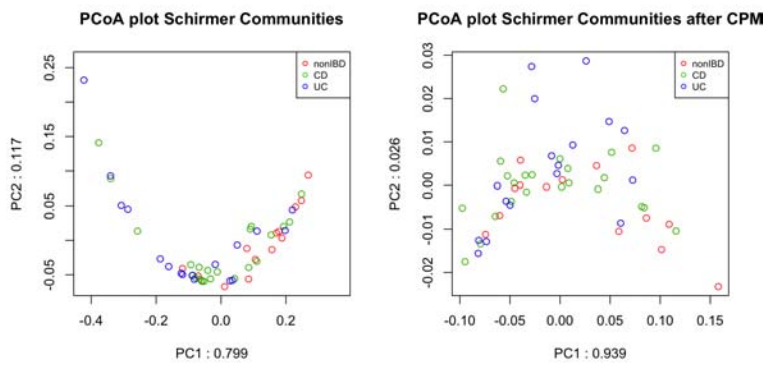


Figure 3.15: comparison of pathways at community level in Schirmer et al. dataset pre and post CPM normalization

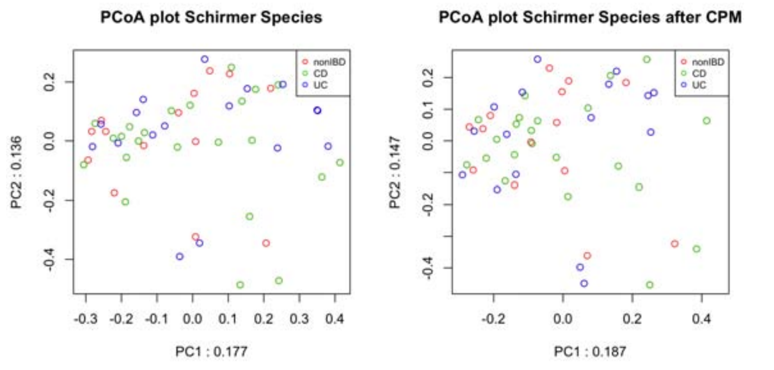


Figure 3.16: comparison of pathways at species level in Schirmer et al. dataset pre and post CPM normalization

3.6 NMDS: concept and plot comparison

NMDS (Non-metric Multidimensional Scaling) attempts to represent the pairwise dissimilarity between objects in a low-dimensional space. Any dissimilarity coefficient or distance measure may be used to build the distance matrix used as input.
NMDS is a rank-based approach, this means that the original distance data is substituted with ranks. While information about the

magnitude of distances is lost, rank-based methods are generally more robust to data which do not have an identifiable distribution.
NMDS is an iterative algorithm and its routines often begin by random placement of data objects in ordination space. Then, the algorithm starts to refine this placement by an iterative process. Its attempt is to find the ordination in which ordinated object distances closely match the order of the object's dissimilarities in the original distance matrix. The resulting stress value reflects how the ordination summarizes the observed distances among the samples.
NMDS is not an eigen analysis and this has three important consequences:

- There is no unique ordination result
- The axes of the ordination are not ordered according to the variance they explain
- The number of dimensions of the low-dimensional space must be specified before running the analysis
- There is no unique solution

The end solution depends on the random placement of the objects in the first step. Running the NMDS algorithm multiple times to ensure that the ordination is stable is necessary, as any one run may get trapped in local optima which are not representative of true distances.

To implement this method in R I used the same packages used for PCoA but this time, instead of using the *pcoa()* function I used *metaMDS()* function that automatically runs NMDS multiple time to ensure not to trap into a local optima.

```
NMDS <- function(sample) {
  set.seed(2)
  dist=vegdist(sample, method = ''bray'') # compute the dissimilarity
  # matrix
  NMDS1=metaMDS(dist, k = 2, trymax = 100, trace = F) # compute NMDS
  # 100 times
  PC=NMDS1$points
  return(PC)
}
```

In the following graphs we can see some comparisons using this technique pre and post normalization.

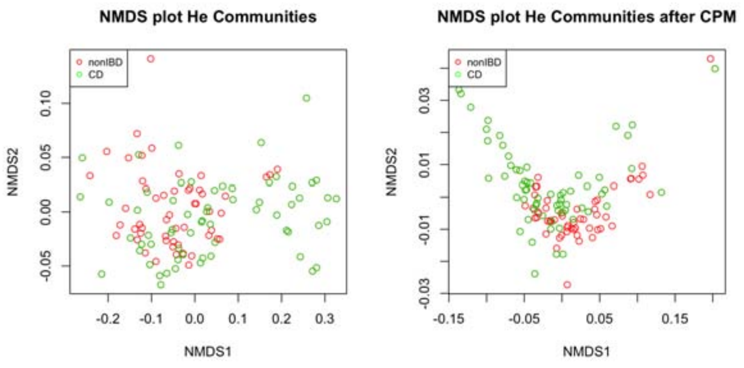


Figure 3.17: comparison of pathways at community level in He et al. dataset pre and post CPM normalization

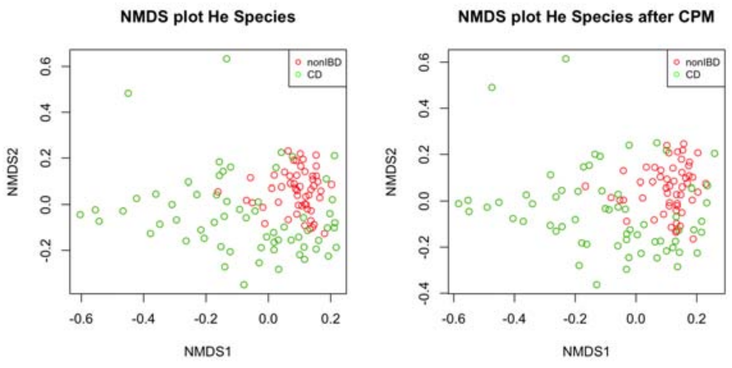


Figure 3.18: comparison of pathways at species level in He et al. dataset pre and post CPM normalization

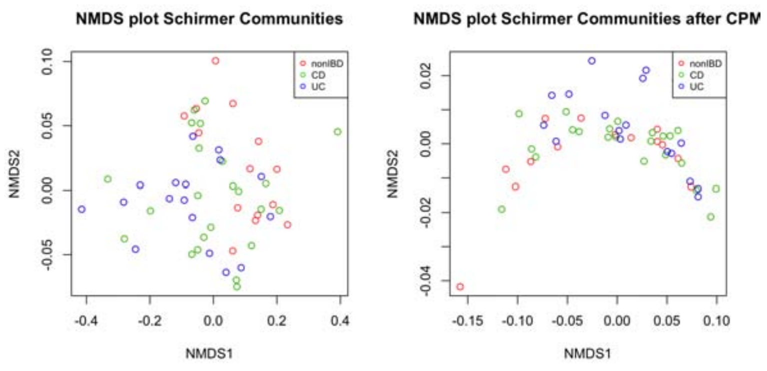


Figure 3.19: comparison of pathways at community level in Schirmer et al. dataset pre and post CPM normalization

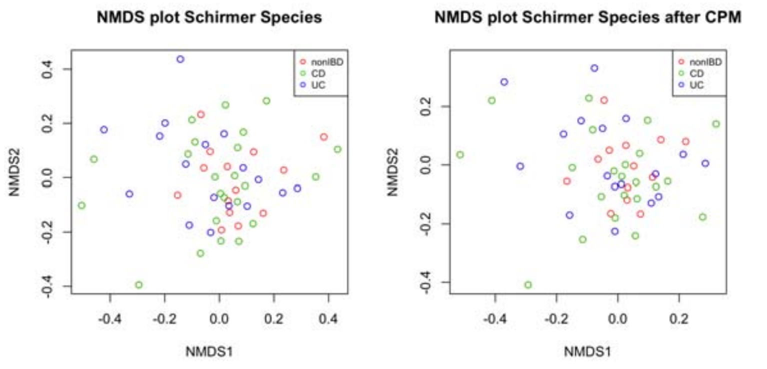


Figure 3.20: comparison of pathways at species level in Schirmer et al. dataset pre and post CPM normalization

The past graphs can give us a first impression on how good the classifier we will build could be.
A good separation between classes means that it will be easy to build a classifier to discriminate them, in contrary, if we donât see a clear separation means that it could be harder to classify them with high precision. It’s important to underline that these graphs show the problem only in 2-dimensional space and so the classes may appear harder to separate than in a higher dimensional space.

From the Schirmer et al. dataset graphs we can see that the separation between classes is not clear at all, something better we can see in the He et al. Species where we can see a better separation.

What we can try to do now is to merge the two dataset and do the PCoA or NMDS taking only the subjects requested in the 4 comparisons, namely:

- IBD vs non-IBD
- UC vs non-IBD
- CD vs non-IBD
- UC vs CD

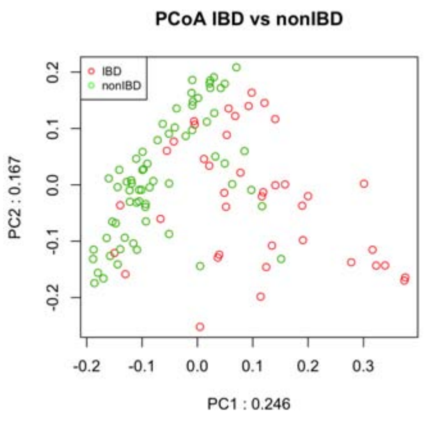


Figure 3.21: PCoA plot of the IBD vs non-IBD subjects on the entire dataset

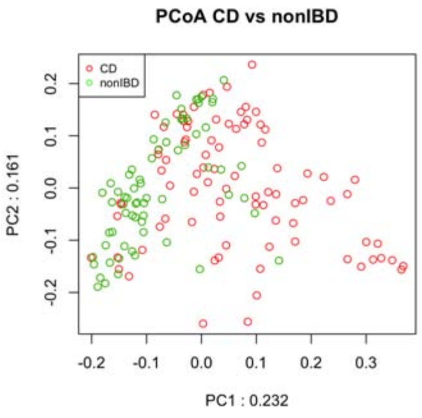


Figure 3.22: PCoA plot of the CD vs non-IBD subjects on the entire dataset

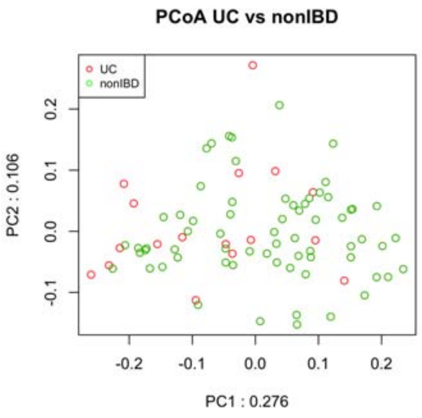


Figure 3.23: PCoA plot of the UC vs non-IBD subjects on the entire dataset

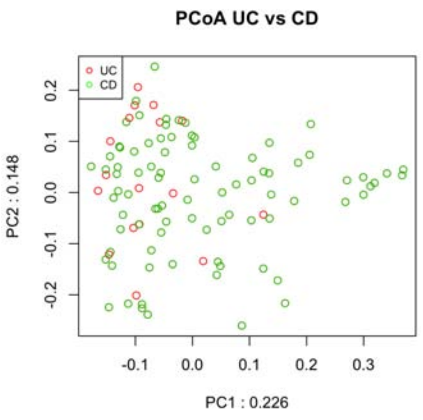


Figure 3.24: PCoA plot of the UC vs CD subjects on the entire dataset

In this way what we can see is a clearer separation in the case of IBD vs non-IBD, a slightly better separation in CD vs non-IBD case and not improved results in the last two comparisons (UC vs non-IBD and UC vs CD) probably because these health states are less different in a biological point of view.

Chapter 4

Classification Method

After a first screening of the data the next step is choosing the classification method.
There are many types of methods, from the simpler and faster Naive Bayes classifier to the more complex and more computationally onerous like Random Forest. The one we have chosen is a good tradeoff between these two characteristics, the Support Vector Machine (SVM).

4.1 Introduction to SVM classifier

A Support Vector Machine is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.
In two-dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

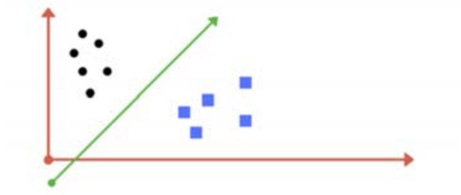


Figure 4.1: svm's hyperplane

The algorithm, implemented by Vapnik et al., minimizes the upper bound of the generalization error by maximizing the margin between the separating hyperplane and the data, abiding to the structure risk minimization principle for model selection [10].

A usual binary classification problem usually involves separating data into training and test sets. The samples of the training set are the pairs (x_i, y_i), where x_i is a vector representing the features of the given sample and $y_i \in \{1, 1\}$ is the corresponding class label. The goal of SVM is to produce a model based on the training data which predicts the class labels of the test data given only the feature vectors of the test data.

$$\begin{aligned} \min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \\ subjecttoy_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ \xi_i \geq 0, i = 1, ..., l \end{aligned} \tag{4.1}$$

where $\phi(x_i)$ maps x_i into a higher-dimensional (and potentially even an infinite-dimensional) space, and $C > 0$ is the penalty parameter of the error term. In practice the dual formulation of this problem is solved instead, due to high dimensionality of the vector variable w:

$$\begin{aligned}\min_{\alpha} \frac{1}{2}\alpha^T y_i y_j \phi(x_i)^T \phi(x_j)\alpha - e^T\alpha \\ subjecttoy^T\alpha \\ 0 \leq \alpha_i \leq C, i = 1, ..., l\end{aligned} \tag{4.2}$$

Where e is rhe vector of all ones.
After solving the dual problem, the optimal w is given by:

$$w = \sum_{i=1}^{l} y_i \alpha_i \phi(x_i) \tag{4.3}$$

and by setting $K(x_i, x_j) = \phi(x_i)^T \phi(x_j) K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ the decision function is give by:

$$\begin{aligned} f(x) &= sgn(w^T \phi(x_i) + b) \\ &= sgn(\sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b) \end{aligned} \tag{4.4}$$

Lastly, is important to notice that there is no need to compute the mapped feature vectors $\phi(x)$ explicitly. Instead, only the dot products between mapped feature vectors are calculated $K(x_i, x_j) = \phi(x_i)^T \phi(x_j) K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$.
$K(x_i, x_j)$ is also known as *Kernel function*.
The function of Kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types, for example: linear, nonlinear, polynomial, radial, and sigmoid. The role of the Kernel is to project the data into a higher dimension space in order to find the hyperplane which classifies the data without increasing the computational cost much.

During this work I used to different types of Kernels:

- For feature selection purpose I used a simpler and slightly faster Linear Kernel: $K(x_i, x_j) = x_i^T x_j K(x_i, x_j) = x_i^T x_j$

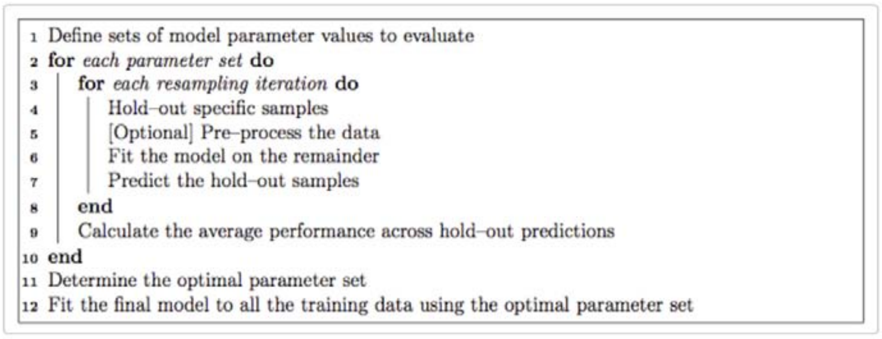
- For build up the main classifier I used a more sophisticated one, the Radial Basis Function (RBF) Kernel: $K(x_i, x_j) = exp(-\gamma||x_i - x_j||^2)$, $\gamma > 0$

4.2 SVM Implementation

In order to implement this type of classifiers in R I used two packages: *e1071* the one in which is implemented the SVM classifier and *caret* which contains functions to streamline the model training process.
One of the primary tools used in this package is the train function which can be used to:

- Evaluate, using resampling, the effect of model tuning parameters on performance
- Choose the optimal model across these parameters
- Estimate model performance from training set

These processes can be summarized by the following figure.



```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

Figure 4.2: caret's workflow

The tuning parameters can be chosen automatically or can be passed to the *train* function by the *tuneGrid* argument.

SVM classifier can be tuned mainly by two parameters: C and Gamma (or Sigma).

- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger margin separating hyperplane, even if that hyperplane misclassifies more points.
- The Gamma (or Sigma) parameter defines how far the influence of a single training example reaches. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.

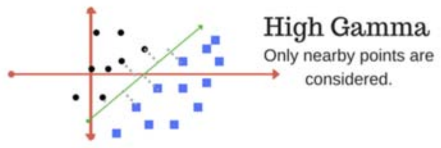


Figure 4.3: Gamma's tuning effect

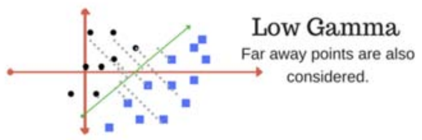


Figure 4.4: Gamma's tuning effect

In order to modify the resampling method, a "trainControl" function is used. Moreover, with the option "method" is it possible to control the type of resampling. For this purpose, it was necessary

to use "repeatedcv" to specify repeated K-fold cross-validation.
Finally, to choose different measures of performance, additional arguments are given to *trainControl*. The "summary Function" argument is used to pass in a function that takes the observed and predicted values and estimate some measure of performance. Two such functions are already included in the package: *default-Summary* and *twoClassSummary*. I used the latter that compute measures specific for to two-class problems, such as the area under the ROC curve, the sensitivity and specificity.
Since the ROC curve is based on the predicted class probabilities (which are not computed automatically), another option is required. The *classProbs = TRUE* option is used to include these calculations.
Lastly, the function will pick the tuning parameters associated with the best results. Since I am using custom performance measures, the criterion that should be optimized must also be specified. In the call to train, I used *metric = "ROC"* to do this.

```
# trainControl object setup for 10 fold CV
control <- trainControl(method = ''repeatedcv'',
                        number = 10,
                        repeats= 10,
                        classProbs = TRUE,
                        summaryFunction = twoClassSummary,
                        search = ''grid'',
                        allowParallel = TRUE,
                        preProcOptions = c(''center'', ''scale''),
                        savePredictions=''final'')

C = 2^(-5:15) # grid of C values
sigma = 2^(-15:3) # grid of sigma values
grid <- expand.grid(C = C, sigma = sigma)

tic()
best.svm.model <- train(Label~.,
                        data=whole.dataset[, c(best.features, ''Label'')]
                        method = ''svmRadial'',
                        preProcess = c(''center'', ''scale''),
                        trControl=control,
                        tuneGrid=grid,
                        maximize=TRUE,
                        metric = ''ROC'')
toc()
```

Chapter 5

Feature Selection Methods

One of the main obstacles in the classification of this dataset in the huge number of PathIDs that the dataset contains (around 12k). In this situation is impossible to apply any type of classifiers and a process of feature selection is necessary [12]. In literature there are many types of feature selection algorithms; in this case I used/implemented two of them: a modified version of Recursive Feature Elimination (RFE) [13] and Randomised Logistic Regression (RLR) [11].

5.1 Recursive Feature Elimination

The most known Recursive Feature Elimination is a technique that begins by building a model on the entire set of the features and computing an importance score for each feature. The least important predictor is then removed, the model is re-build, and the importance scores are computed again. This procedure continues until it reaches a pre-selected number of features.

There are many ways to compute importance scores and the one implemented is the one described in [13].

The parameters that the algorithm takes as input are the following:

- *feat.select.sample*: the samples to use for feature selection

- *class.labels:* the class label for each subject
- *positive.label:* the positive class label
- *negative.label:* the negative class label
- *bootstrap.iterations:* number of bootstrapping iterations
- *percentage.to.remove:* percentage of features to remove after each iteration
- *feature.number.threshold:* when to start removing just one features per iteration
- *parallel.bootstrap:* if the user wants to parallelize the bootstrap iterations

The idea is to define the importance of a feature for a SVM in terms of its contribution to a cost function $J(\alpha)$. At each step of the RFE procedure, an SVM is trained on the given dataset, J is computed and the feature less contributing to J is discarded. In the case of linear SVM, the variation due to the elimination of the i-th feature is $\partial J(i) = w_i^2$. Where w is calculated multiplying *coefs* (the weights of each support vector) by SV (the support vectors).
The elimination of the less informative features takes now a different direction from the one described in the paper.
At the beginning of the algorithm, and for each *bootstrap.iterations*, the *feat.select.sample* are partitioned in a training set and in a test set and from now on the feature selection begins. As input we can find *feature.number.threshold* and *percentage.to.remove* parameters.
These two parameters work in this way:

- If the number of remaining features (after the computations of their score) are greater than *feature.number.threshold* the algorithm removes the *percentage.to.remove*e less informative features

- If the number of remaining features (after the computations of their score) are less than *feature.number.threshold* the algorithm removes just the last informative one

The algorithm proceeds until the last feature has been removed. Proceeding through these steps, the algorithm measures the performance of the features kept in each iteration using the test set trough an MCC metric.

The Matthews correlation coefficient (MCC) is a measure of the quality of a binary classification introduced by Brian W. Matthews in 1975. The MCC is a correlation coefficient between the observed and predicted binary classifications; it returns a value between â1 and 1. A coefficient of 1 represents a perfect prediction, 0 no better than random prediction and â1 indicates total disagreement between prediction and observation. The MCC can be easily calculated from a confusion matrix through this formula:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5.1}$$

At the end, the algorithm takes the mean of the feature importance and of the MCC values through all the bootstrap iterations and selects the number of features that maximizes the average MCC.

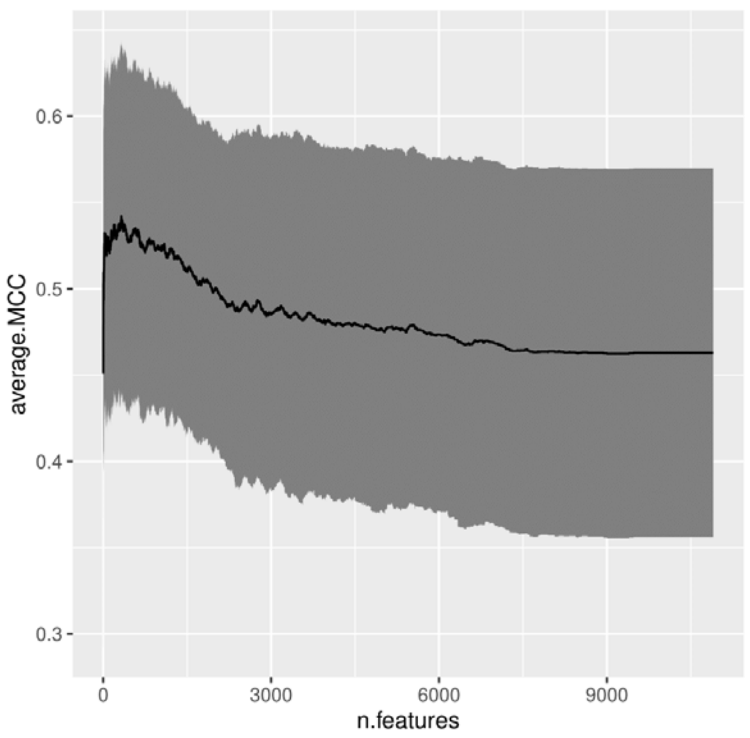


Figure 5.1: MCC's values through the number of features selected

The code is also parallelized using *doMC* R package that permits to choose the number of cores to use for parallel execution with *foreach* package.

The algorithm then returns a list containing the following objects:

- *average.MCC:* the average MCC values of the bootstrap iterations
- *median.MCC:* the median of the MCC values of the bootstrap iterations
- *stdev.MCC:* the standard deviation of the MCC values of the bootstrap iterations
- *optimal.number.of.features :* the optimal number of features

calculated as the index in which *average.MCC* reaches its maximum

- *best.features:* the features names corresponding with the optimal number of features
- *sorted.average.feature.rank:* a sorted list of feature by their importance

5.2 Randomized Logistic Regression

Randomised Logistic Regression works by resampling the train data and computing a Logistic Regression on each resampling. In short, we can say that the features selected more often are good features.

Also known as stability selection, this algorithm is an example of an embedded feature selection method that tends to work well in high-dimensional, sparse problems like the one we are facing with. The rationale behind the algorithm is to bootstrap many times the dataset and use a base structure algorithm to find out which features are important in every sampled version of the data. A feature to be selected as important for the classification of the subjects has to be selected in a high number of bootstrap iterations of the algorithm. On the contrary, a feature not related to the target variable has not to be selected in any iteration with the consequences that this tends to filter out features weakly related to the classification problem.

The parameter that the algorithm takes as input are the following:

- *feat.select.sample:* a matrix or data frame containing the samples to be used for feature selection
- *metadata:* the class labels of the subjects

- *positive.label:* the label of the positive class
- *negative.label:* the labels of the negative class
- *n.bootstrap.iterations:* the number of bootstrapping iterations
- *lambda.values:* list of regularization parameters
- *alpha:* scale value for the regularization parameters used in the internal randomization step (default 0.2)
- *p.w:* the probability for a feature to be penalized with a (*lambda/alpha*) regularization factor (default 0.5)
- *sample.fraction:* fraction of the overall samples to be re-sampled at each bootstrap iteration (default 0.5)
- *regularization.type:* "1": for LASSO regression; "0": for ridge regression; a value between 0 and 1 for elastic net regression (default 1)
- *epsilon:* threshold used to determine if the Logistic Regression model coefficients are not-null (defaults to $1e^{-}3$)
- *selection.threshold:* the threshold value used to determine the selected features
- *parallel:* if TRUE, the iterations are performed in parallel

The procedure used to list the feature importance is the Generalized Linear Model implemented in the *glmnet* R package.
The function has been used as follows:

```
lr.model <- glmnet(x = x.train,
y = y.train,
lambda = lambda,
penalty.factor = w,
alpha = regularisation.type,
family = ''binomial'',
standardize = TRUE)
```

Where:

- *x.train :* the dataset
- *y.train :* the class labels passed as factor in the form (0,1)
- *lambda:* a regularization parameter that identifies the regularization strength
- *w:* a number that multiplies lambda to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model
- *regularization.type:* same as mentioned before
- *family:* the type of distribution that the regression follows
- *standardize:* a flag for the variable standardization

The algorithm then takes the parameter *coef* from the output of the *lr.model.* This parameter contains the coefficients of the linear model used as feature weights.
In a first nested loop the algorithm calculates the feature weights for every bootstrap iterations with a fixed *lambda*, and on a second external loop iterates on different *lambda.values* passed as initial input of the algorithm. During these steps the algorithm computes the mean of the feature importance for every bootstrap iterations and then, for every *lambda.values*, returns a vector of feature selection probabilities; if a feature's selection probability is greater than *selection.threshold* is then selected.
The algorithm returns a list containing the following items:

- *feature.selection.probabilities:* the probability that each feature is selected by the algorithm
- *stability.scores:* the stability scores for each feature
- *selected.features:* the list of features with *feature.selection.probabilities* greater than the *selection.threshold*

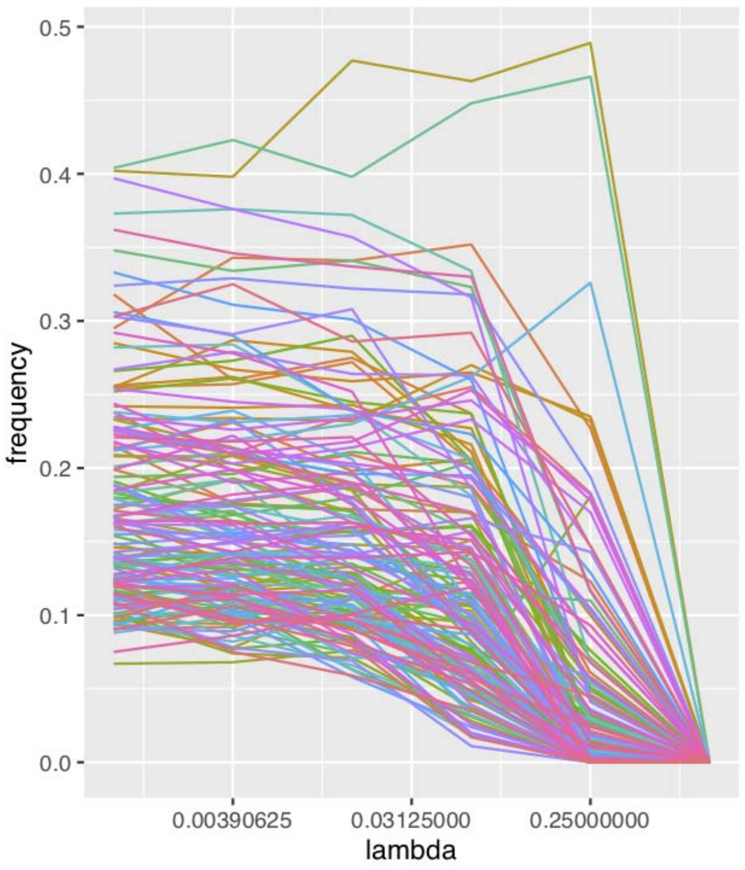


Figure 5.2: selection frequency of the selected features through the *lambda.values*

5.3 Training and Test set partition

Once we completed our feature selection and classification pipeline, another important aspect is how to test the classification performance.

For testing it we decided to split the provided training set provided in another Training and Test set in order to apply the feature selection and classification on the new training set and use the classifier in a label known dataset (the new Test set) in order to measure the performance of the pipeline.

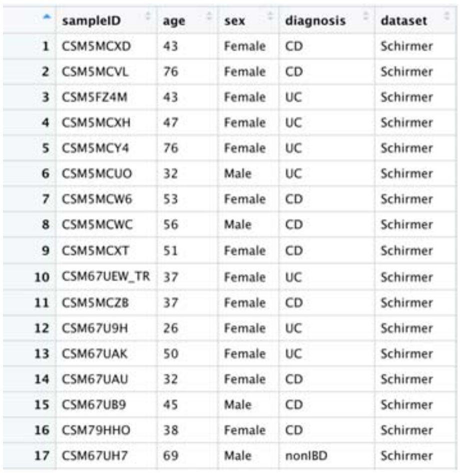
To decide in which measure split the original Training set we had to deal with two competing concerns: with less training data, the parameter estimates have greater variance. With less testing data, the performance statistic will have greater variance.
Knowing that, we went for an 80/20 division keeping the same proportion between the two health states considered in the classification task.
In the following an R script that exemplifies the process of division:

```
## Train-test split
## Here we split the dataset into training and test sets.
## We split *IBD* and *nonIBD* patients with the same proportions.
set.seed(42)
### 80%-20% split
train.IBD.class.idx <- sample(IBD.class.idx,
size = floor(0.80 * length(IBD.class.idx)))
train.nonIBD.class.idx <- sample(nonIBD.class.idx,
size = floor(0.80 * length(nonIBD.class.idx)))
train.idx <- c(train.nonIBD.class.idx,train.IBD.class.idx)

training.set <- whole.dataset[train.idx,]
training.class.labels <- class.labels[train.idx,]
training.class.labels$group=droplevels(training.class.labels$group)

test.set <- whole.dataset[-train.idx, ]
test.class.labels <- class.labels[-train.idx, ]
test.class.labels$group=droplevels(test.class.labels$group)
```

For the first tests, we divided test and training as mentioned, using the only information we had on the patients. Looking for the two original datasets (Schrimer et al. and He et al.) on the web, we found that there were many more metadata on them such as age and sex.



	sampleID	age	sex	diagnosis	dataset
1	CSM5MCXD	43	Female	CD	Schirmer
2	CSM5MCVL	76	Female	CD	Schirmer
3	CSM5FZ4M	43	Female	UC	Schirmer
4	CSM5MCXH	47	Female	UC	Schirmer
5	CSM5MCY4	76	Female	UC	Schirmer
6	CSM5MCUO	32	Male	UC	Schirmer
7	CSM5MCW6	53	Female	CD	Schirmer
8	CSM5MCWC	56	Male	CD	Schirmer
9	CSM5MCXT	51	Female	CD	Schirmer
10	CSM67UEW_TR	37	Female	UC	Schirmer
11	CSM5MCZB	37	Female	CD	Schirmer
12	CSM67U9H	26	Female	UC	Schirmer
13	CSM67UAK	50	Female	UC	Schirmer
14	CSM67UAU	32	Female	CD	Schirmer
15	CSM67UB9	45	Male	CD	Schirmer
16	CSM79HHO	38	Female	CD	Schirmer
17	CSM67UH7	69	Male	nonIBD	Schirmer

Figure 5.3: example of the metadata table

In the literature it is known that our microbiome changes during our life. According to [19], age is a key factor in the diversification of our microbiome. For this reason, it was necessary to take these metadata into account for keeping the training and test set as balanced as possible.

In order to do so, we use a function called stratified which takes as input:

- the aforementioned metadata data.frame
- the column names of the metadata to take into account
- the percentage of the data in the first split
- a flag parameter to choose if you want to keep the proportion also in the second split

Which returns a list containing the IDs of the patients in the first and in the second split.

```
stratified.split <- stratified(metadata, group = c(''age'',
''sex'', ''diagnosis'', ''dataset''), size = 0.8, bothSets=TRUE)
training.set.metadata <- stratified.split$SET1
test.set.metadata <- stratified.split$SET2
```

From now on, every result showed was obtained with this stratified partition.

Chapter 6

Results and Comparisons

6.1 Pipeline performance on Test set

The results and the performance obtained by the two feature selection methods are shown in this section. The main comparison metric used is the ROC's Area Under the Curve (AUROC).

ROC is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0 and 1.
The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how the model is at predicting the positive class when the actual outcome is positive.
The false positive rate is calculated as the number of false positives divided by the sum of the number of false positives and the number of true negatives.

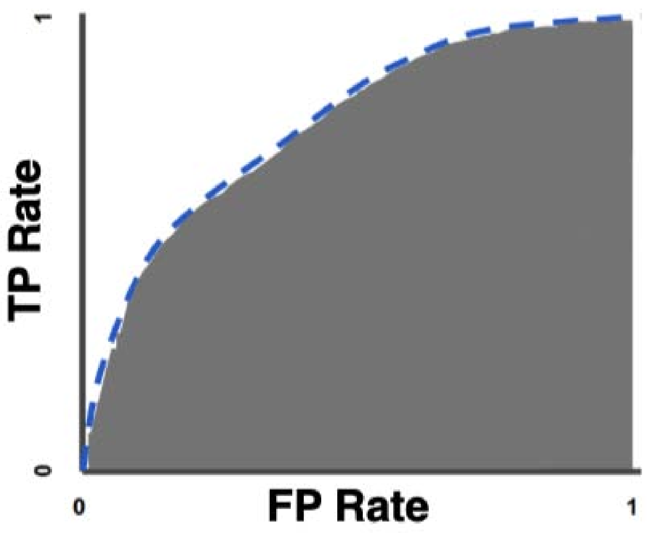


Figure 6.1: AUROC example

The AUROC provides a measure of performance across all possible classification thresholds. Higher is the value of the AUC higher is the ability of the classifier to discriminate between two classes. For example, a classifier with AUROC of 0.5 has an ability to equal to a random classification model.

The pipeline has been applied in 3 different partitions of the entire Pathways matrix: Firstly, keeping only Pathways at Community level. Secondly, keeping only Pathways at Species level. Lastly, using the entire Pathway matrix.
During these steps the pipeline has been applied at the different data matrixes keeping only the subjects for the specific classification task, that is: IBD vs non-IBD, CD vs non-IBD, UC vs non-IBD and UC vs CD. Both using RFE and RLR feature selection algorithms.

The tuning parameters set for the two feature selection algorithms are the following:
RFE:

- *bootstrap.iterations*=100
- *percentage.to.remove*=0.05
- *feature.number.threshold*=100

RLR:

- *n.bootstrap.iterations*=1000
- *sample.fraction*=0.5
- *selection.threshold*=0.1
- *lambda.values*=2ŝeq(-10,0,2)

Kept for every matrix and every classification task.

In total, for every classification task has been build 6 different classifiers. In the following the results achieved using the different partitions and methods.

IBD vs non-IBD classification performance

Community Matrix
RFE(number of feature selected: 158)

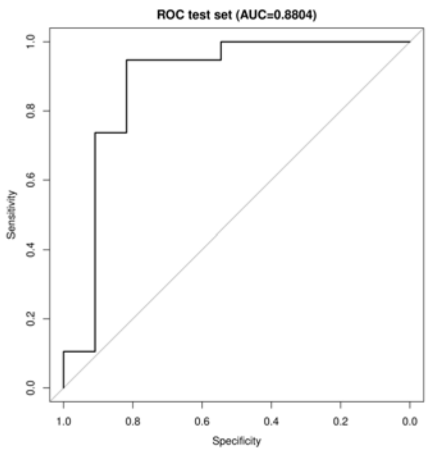


Figure 6.2: AUROC

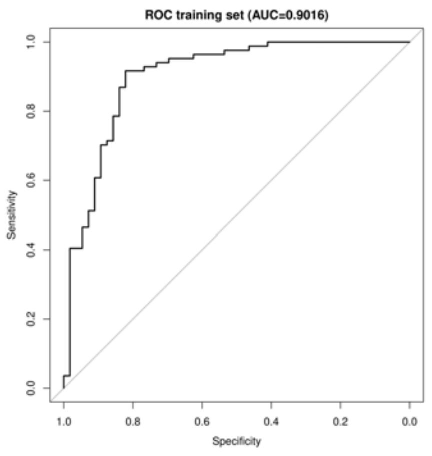


Figure 6.3: AUROC

RLR(number of feature selected: 153)

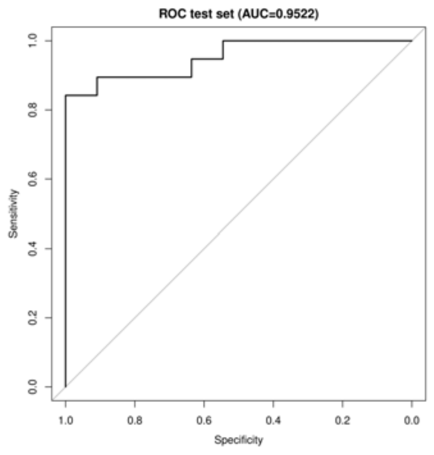


Figure 6.4: AUROC

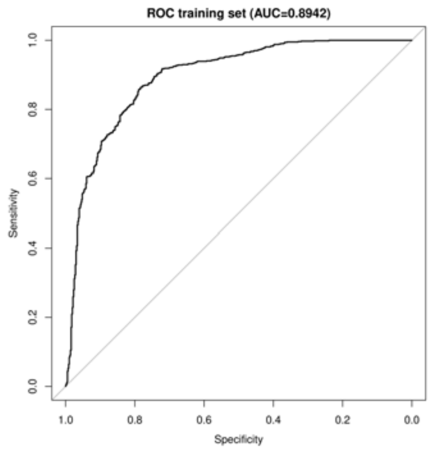


Figure 6.5: AUROC

<u>Species Matrix</u>

RFE(number of feature selected: 616)

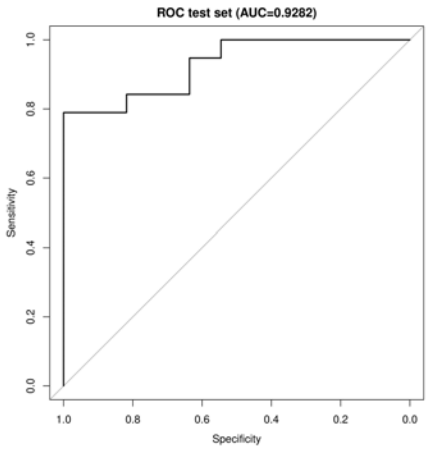


Figure 6.6: AUROC

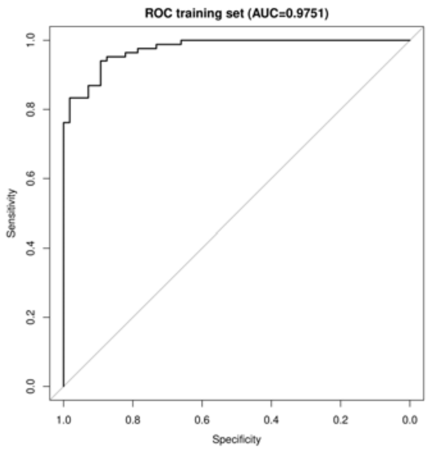


Figure 6.7: AUROC

RLR(number of feature selected: 58)

Figure 6.8: AUROC

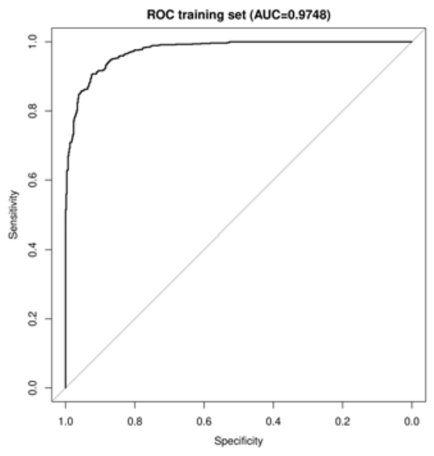


Figure 6.9: AUROC

Entire Matrix

RFE(number of feature selected: 249)

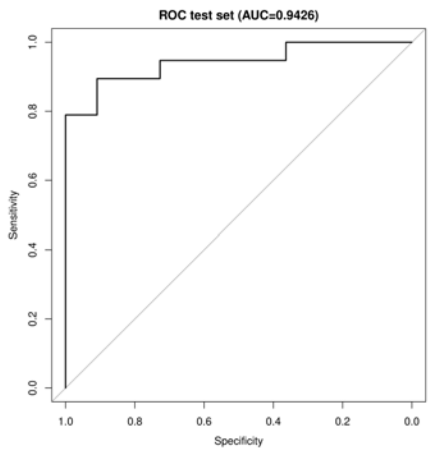


Figure 6.10: AUROC

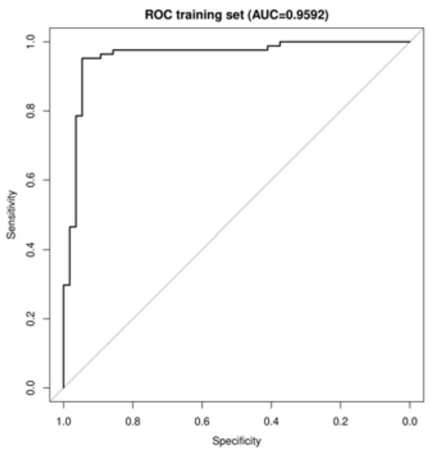


Figure 6.11: AUROC

RLR(number of feature selected: 62)

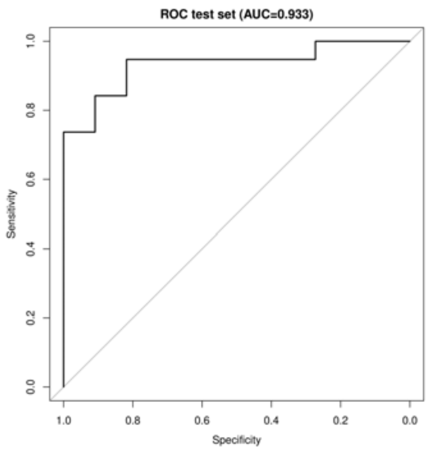


Figure 6.12: AUROC

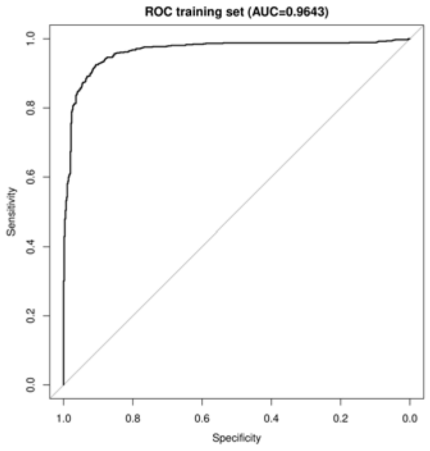


Figure 6.13: AUROC

CD vs non-IBD classification performance

Community Matrix

RFE(number of feature selected: 93)

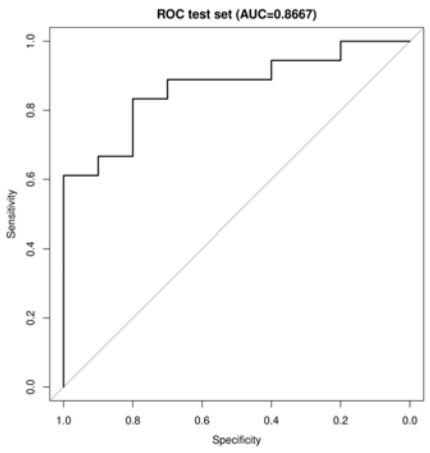


Figure 6.14: AUROC

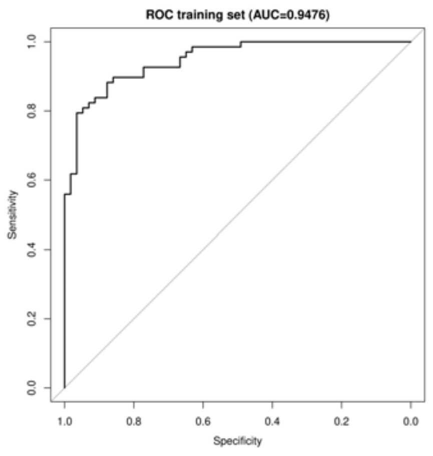


Figure 6.15: AUROC

RLR(number of feature selected: 122)

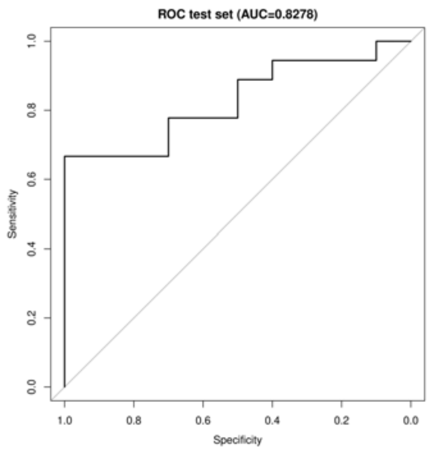


Figure 6.16: AUROC

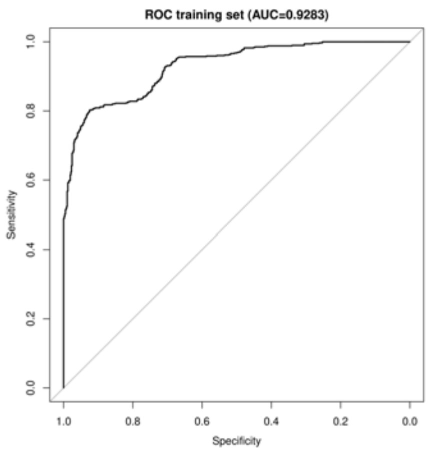


Figure 6.17: AUROC

Species Matrix

RFE(number of feature selected: 408)

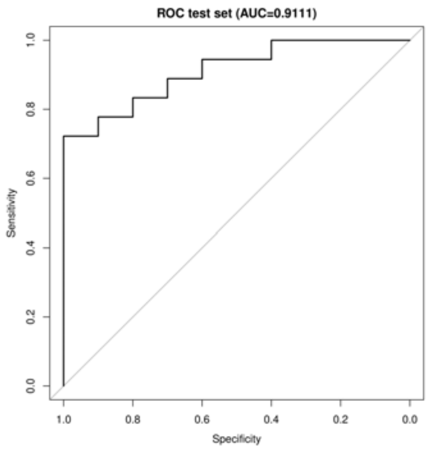


Figure 6.18: AUROC

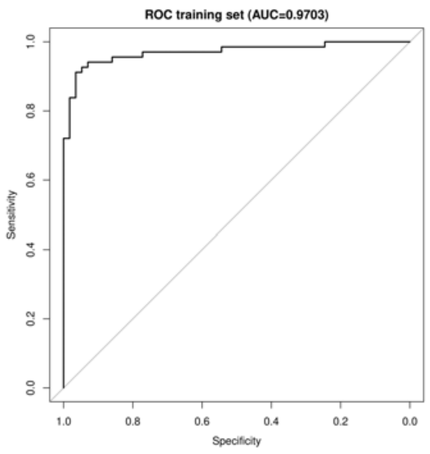


Figure 6.19: AUROC

RLR(number of feature selected: 50)

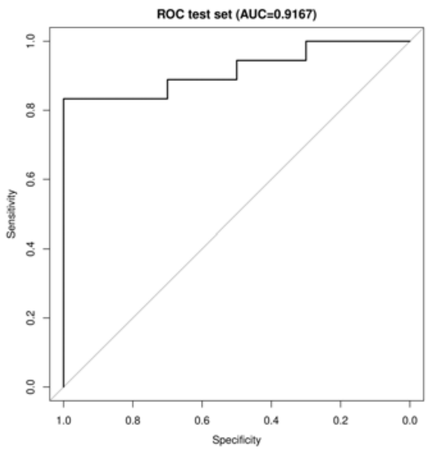


Figure 6.20: AUROC

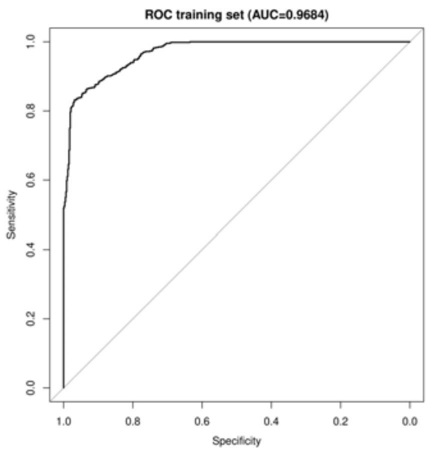


Figure 6.21: AUROC

<u>Entire Matrix</u>
RFE(number of feature selected: 607)

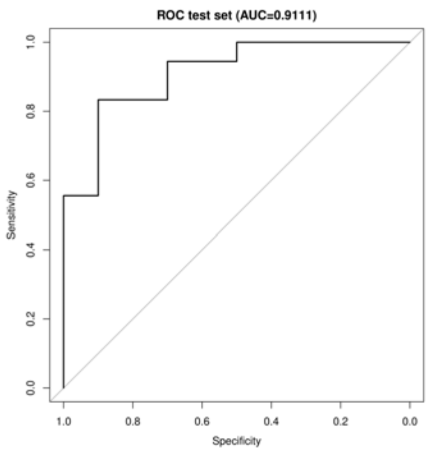


Figure 6.22: AUROC

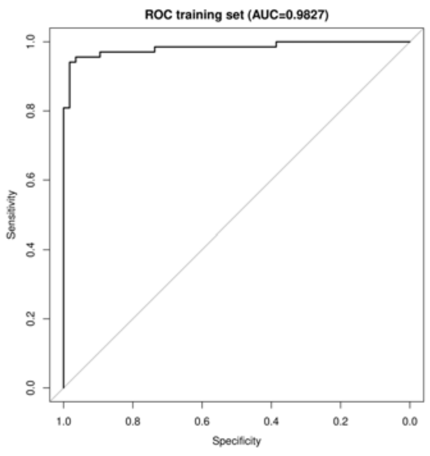


Figure 6.23: AUROC

RLR(number of feature selected: 55)

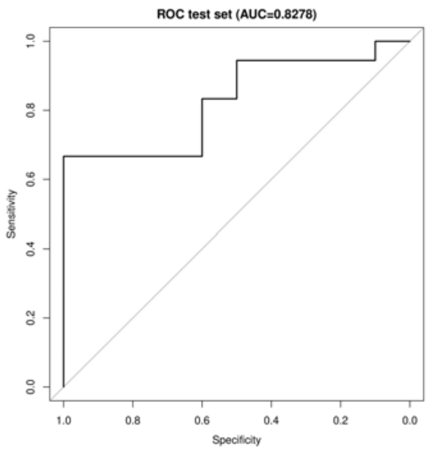


Figure 6.24: AUROC

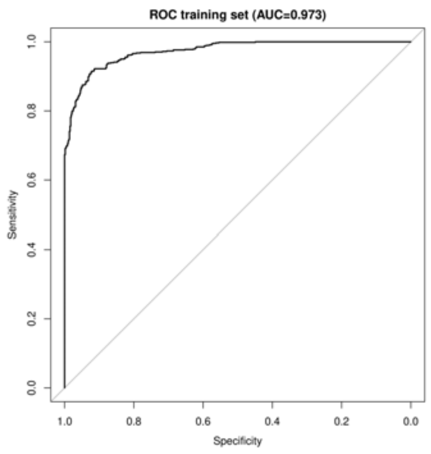


Figure 6.25: AUROC

UC vs non-IBD classification performance

Community Matrix

RFE(number of feature selected: 71)

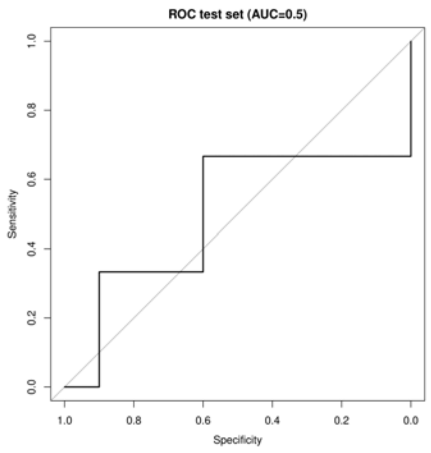


Figure 6.26: AUROC

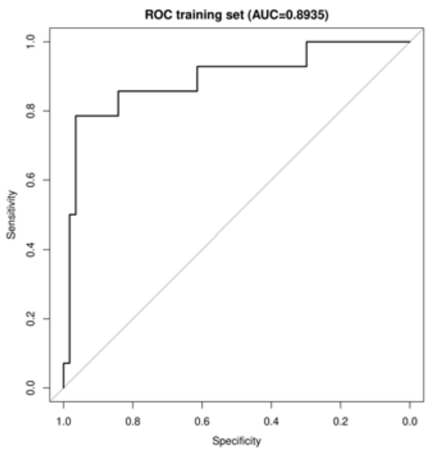


Figure 6.27: AUROC

RLR(number of feature selected: 45)

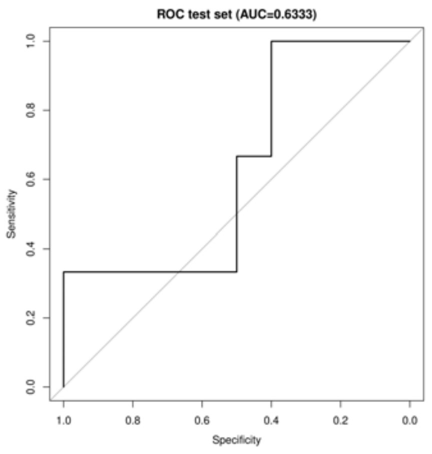


Figure 6.28: AUROC

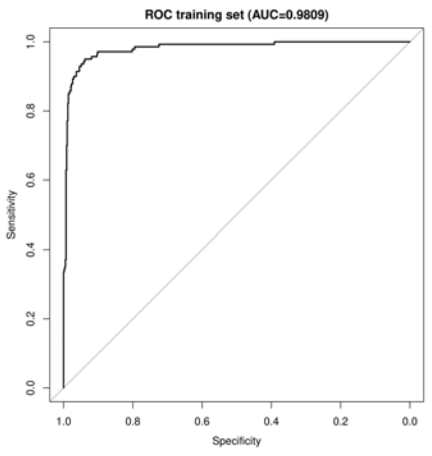


Figure 6.29: AUROC

<u>Species Matrix</u>

RFE(number of feature selected: 24)

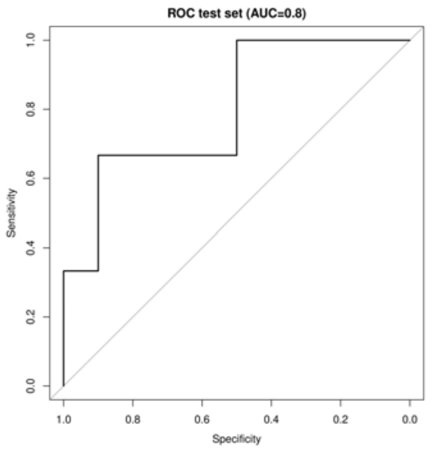


Figure 6.30: AUROC

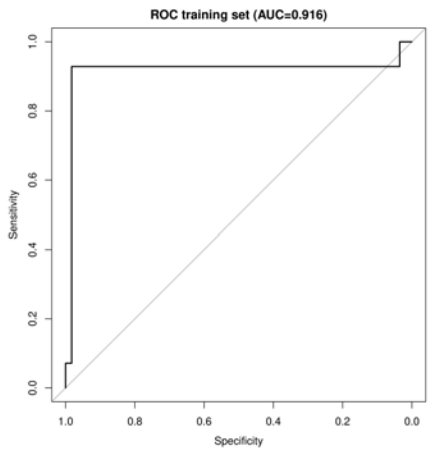


Figure 6.31: AUROC

RLR(number of feature selected: 37)

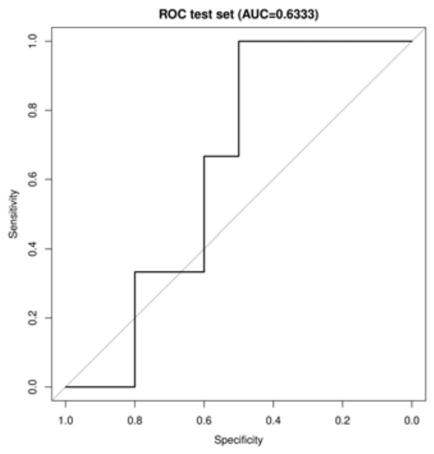


Figure 6.32: AUROC

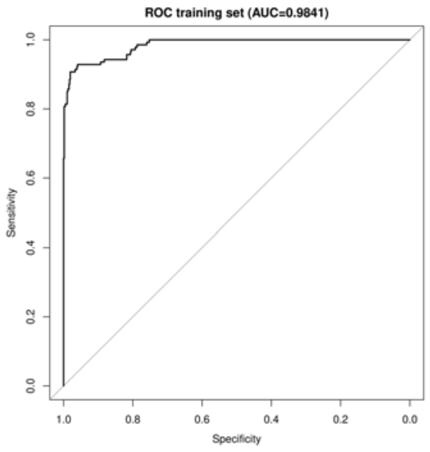


Figure 6.33: AUROC

<u>Entire Matrix</u>
RFE(number of feature selected: 381)

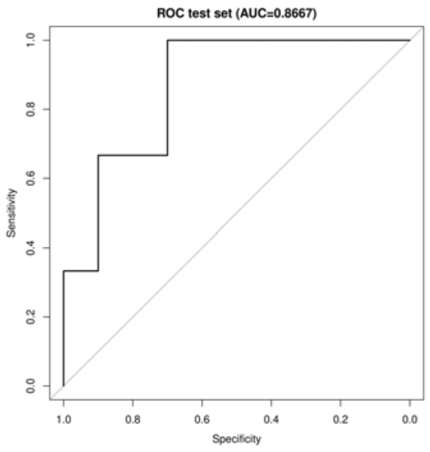


Figure 6.34: AUROC

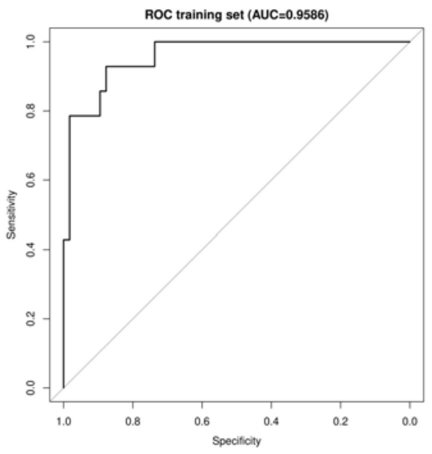


Figure 6.35: AUROC

RLR(number of feature selected: 30)

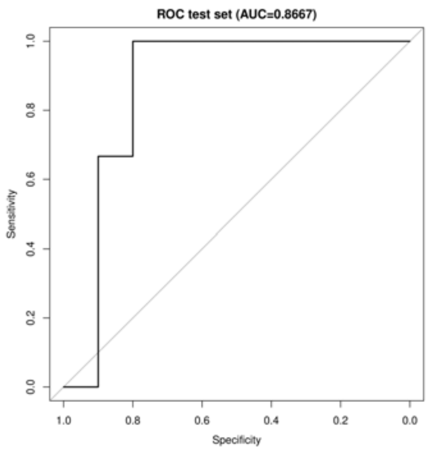


Figure 6.36: AUROC

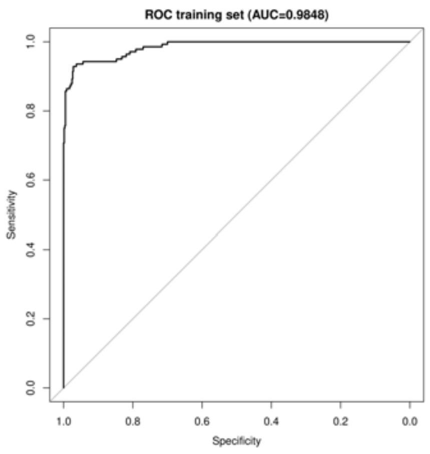


Figure 6.37: AUROC

UC vs CD classification performance

Community Matrix
RFE(number of feature selected: 190)

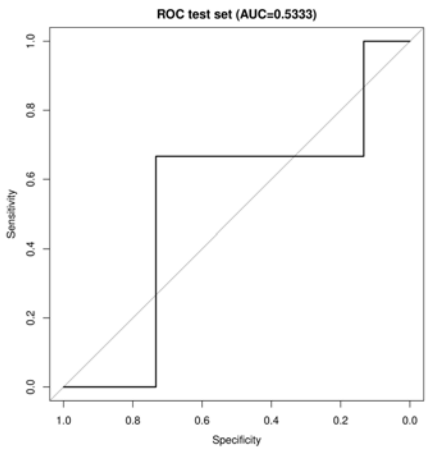


Figure 6.38: AUROC

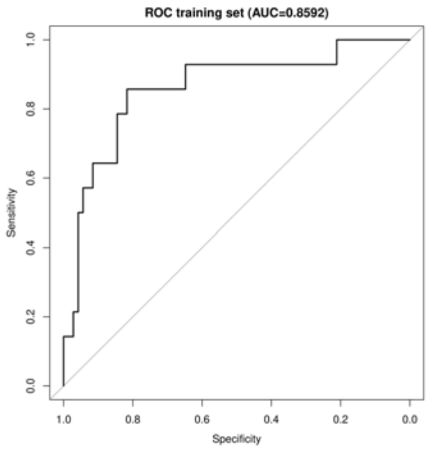


Figure 6.39: AUROC

RLR(number of feature selected: 71)

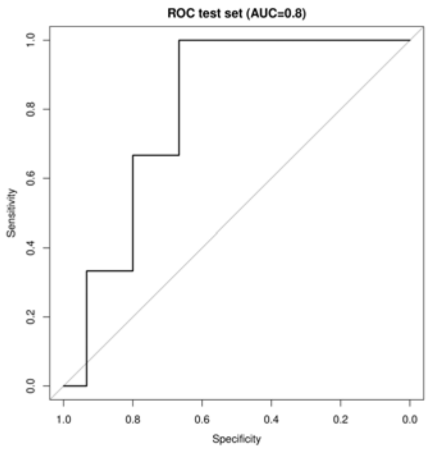


Figure 6.40: AUROC

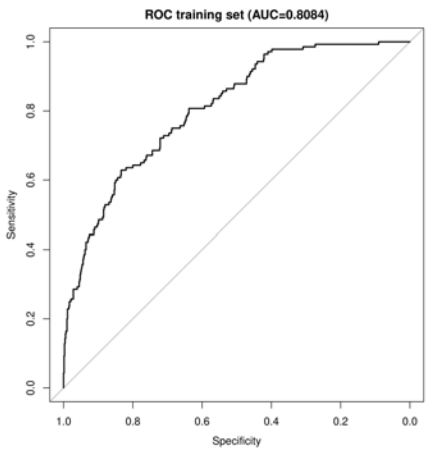


Figure 6.41: AUROC

Species Matrix

RFE(number of feature selected: 336)

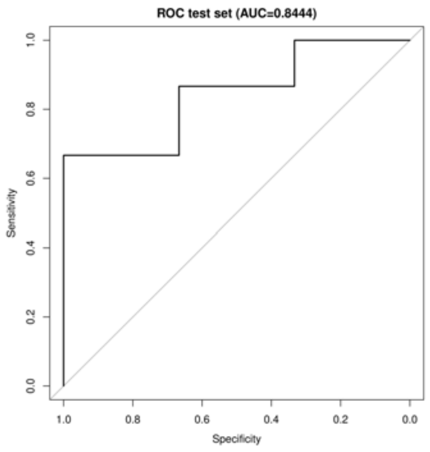


Figure 6.42: AUROC

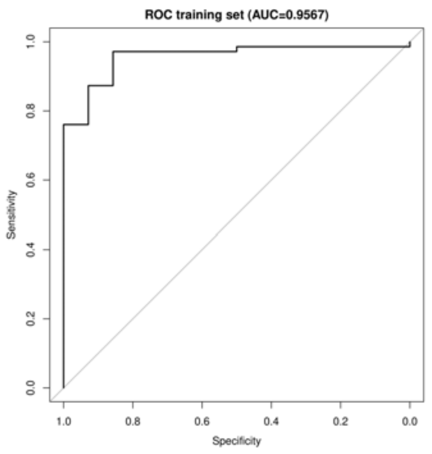


Figure 6.43: AUROC

RLR(number of feature selected: 28)

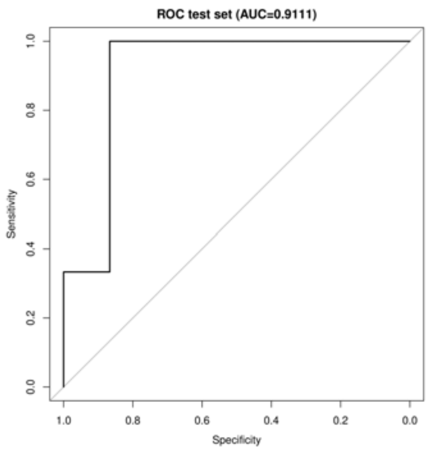


Figure 6.44: AUROC

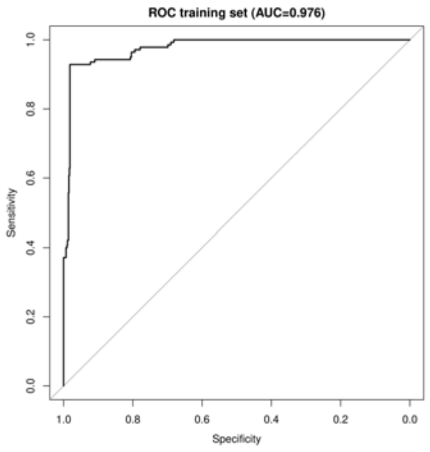


Figure 6.45: AUROC

Entire Matrix
RFE(number of feature selected: 201)

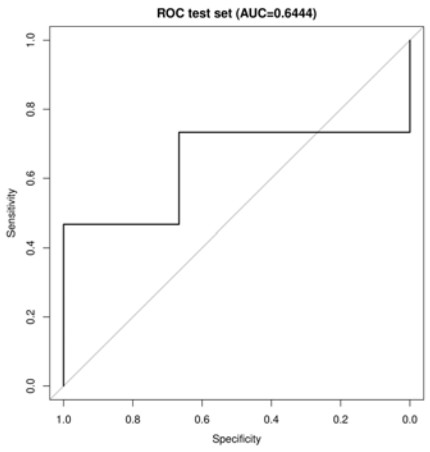


Figure 6.46: AUROC

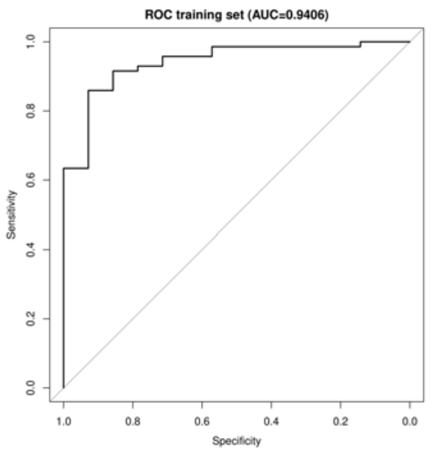


Figure 6.47: AUROC

RLR(number of feature selected: 30)

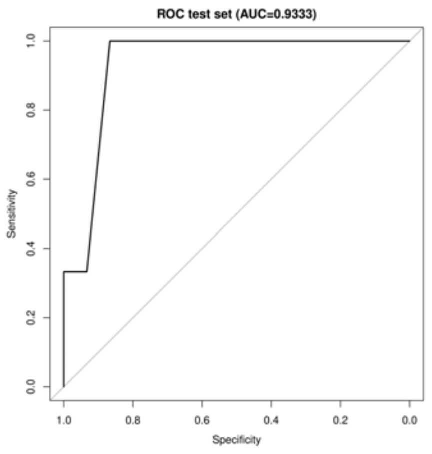


Figure 6.48: AUROC

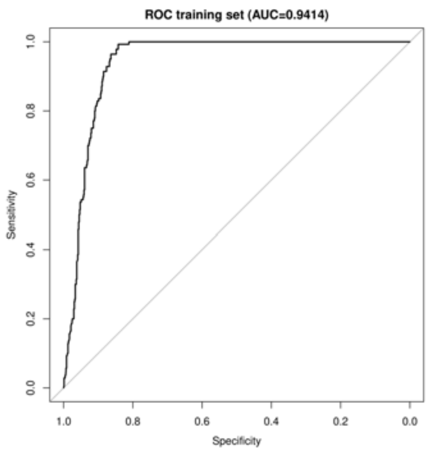


Figure 6.49: AUROC

Analyzing the previous graphs, we can make some final considerations on the problem given by the challenge organizers.

On IBD vs non-IBD classification

The AUROC shows that this task is the easiest to deal with, every type of pipeline used gave high AUC values reaching the highest value with the RFE algorithm on the entire matrix (AUC$\cong$0.96). One important thing to notice is that, using just Community Matrix, which contains only 448 features over twelve thousand (3.7% of the total) it's possible to reach AUC$\approx$ 0.95.

On CD vs non-IBD classification

In this case, on every matrix except the Community one (similar results), the RLR algorithm outperforms RFE, reaching his maximum AUROC value on the Species Matrix (AUC$\cong$ 0.91).

On UC vs non-IBD classification

The performance of the classifiers shows that this task is the hardest one, both for RFE and RLR algorithm. What we can see are very high AUC's on the training set with every algorithm and in every matrix but, when we try to classify the test set, we see very poor performance probably caused by overfitting. The best performance belongs to the RLR algorithm on the entire matrix with an AUROC$\cong$ 0.86.

On UC vs CD classification

For this last classification the performances of the two methods are almost similar and quite good in the case of Community and Species matrix showing results between 0.8 and 0.9 of AUC. On the entire matrix, instead, RLR outperforms RFE both in training and in test set reaching an AUROC on test set of 0.93.

In order to make a double check on the pipeline performance it is necessary to test it with a LOOCV procedure.
Indeed, this methodology uses a single observation from the original matrix as the validation data, and the remaining observations as the training data. Since this methodology is very computational demanding, it has been only applied in one case: on the entire Pathways matrix with RLR feature selection method (because it is the less computational demanding).
The results are the following:

IBD vs non-IBD classification performance

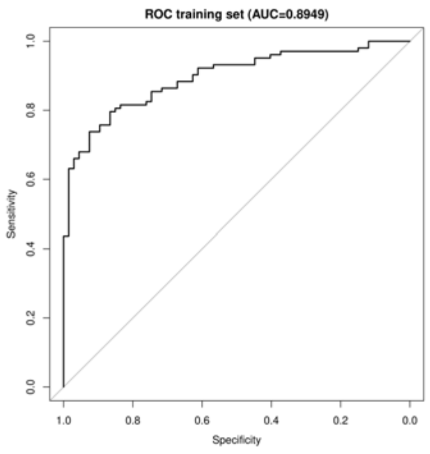


Figure 6.50: AUROC

CD vs non-IBD classification performance

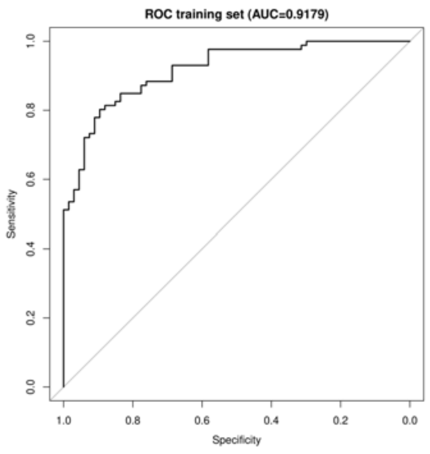


Figure 6.51: AUROC

UC vs non-IBD classification performance

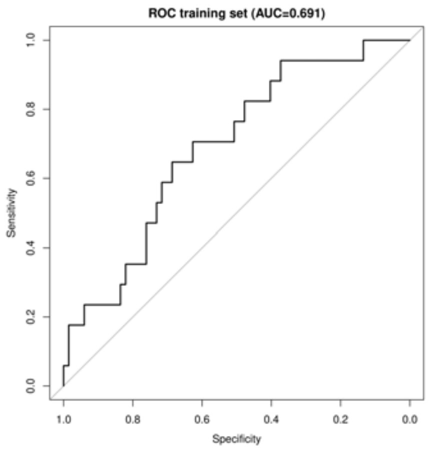


Figure 6.52: AUROC

UC vs CD classification performance

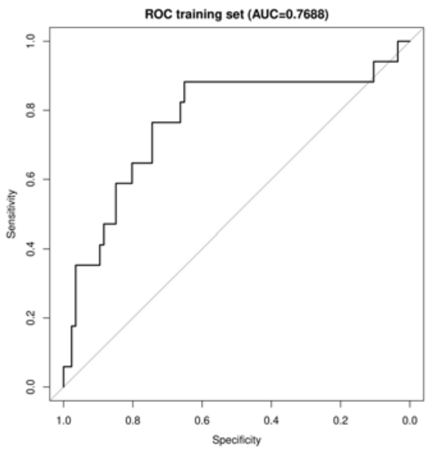


Figure 6.53: AUROC

It is possible to see that, on average, the AUROC's results are lower than the ones obtained with an 80/20 partition. Consequently, this can underline the existence of an inter-subject vari-

ability that lowers the discriminative power of the classifier.
One of the sources of this variability may be the country in which the two datasets have been collected. For Schirmer et al. it was North America, while, for He et al. it was China. Since we are dealing with gut microbiome, the geographical location can make a great difference, because different meal habits influence a lot this type of data.

With these pipelines we underline that the diagnosis of Inflammatory Bowel Diseases through metagenomics marker is viable. In the future, more efforts in the collection of the data have to be done, since classifying subjects with very far origins from the dataset ones is harder and less accurate.

6.2 Challenge submissions

Since the challenge organizers permits multiple submissions and only the best one will be kept for the scoring, we decided to submit different versions of the RLR pipeline varying the input matrix and the *selection.threshold parameter*.
For the RFE pipeline, instead, since the tuning parameters does not change the results much, we varied only the input matrix.
In the following a sum up of the different submissions.

RLR submissions:

- Input matrix: Pathways matrix, *selection.threshold*=0.1
- Input matrix: Pathways matrix, *selection.threshold*=0.15
- Input matrix: Pathways matrix, *selection.threshold*=0.2
- Input matrix: Pathways matrix, *selection.threshold*=0.25
- Input matrix: Pathways matrix, *selection.threshold*=0.3
- Input matrix: Pathways matrix, *selection.threshold*=0.35
- Input matrix: Pathways matrix, *selection.threshold*=0.4
- Input matrix: Entire matrix, *selection.threshold*=0.1
- Input matrix: Entire matrix, *selection.threshold*=0.15
- Input matrix: Entire matrix, *selection.threshold*=0.2
- Input matrix: Entire matrix, *selection.threshold*=0.25
- Input matrix: Entire matrix, *selection.threshold*=0.3
- Input matrix: Entire matrix, *selection.threshold*=0.35
- Input matrix: Entire matrix, *selection.threshold*=0.4

RFE submissions:

- Input matrix: Pathways matrix
- Input matrix: Entire matrix

Bibliography

[1] Staden R, *A strategy of DNA sequencing employing computer programs*, 1979

[2] Julian Pierre, Jordan Taylor, Amit Upadhyay, Bhanu Rekepalli, *Next Generation Sequencing Pipeline Development and Data Analysis*, 2010

[3] Thomas P. Niedringhaus, Denitsa Milanova, Matthew B. Kerby, Michael P. Snyder, Annelise E. Barron, *Landscape of Next-Generation Sequencing Technologies*, 2011

[4] Quail MA, Smith M, Coupland P, Otto TD, Harris SR, Connor TR, *A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers*, 2012

[5] Pennisi E, *Semiconductors inspire new sequencing technologies*, 2010

[6] Ilseung Cho, Martin J. Blaser, *The Human Microbiome: at the interface of health and disease*, 2012

[7] Jun Sun, Eugene B.Chang, *Exploring gut microbes in human health and disease: Pushing the envelope*, 2014

[8] He, Q., Gao, Y., Jie, Z., Yu, X., et al., *Two distinct metacommunities characterize the gut microbiota in Crohn's disease patients*, 2017

[9] Schirmer, M., Franzosa, E. A., Lloyd-Price, J., McIver, et al. *Dynamics of metatranscription in the inflammatory bowel disease gut microbiome*, 2018

[10] Vladimir N. Vapnik, Corinna Cortes, *Support-vector networks*

[11] Meinshausen N., Buhlmann P., *Stability selection*, 2010

[12] Yi-Hui Zhou and Paul Gallins, *A Review and Tutorial of Machine Learning Methods for Microbiome Host Trait Prediction*, 2019.

[13] C. Furlanello, M. Serafini, S. Merler, G. Jurman, *Gene Selection and. Classification by Entropy-based Recursive Feature Elimination*, 2003.

[14] Lauren J. McIver, Galeb Abu-Ali, Eric A. Franzosa, Randall Schwager, Xochitl C. Morgan, Levi Waldron, Nicola Segata and Curtis Huttenhower, *BioBakery: a metaâomic analysis environment*, 2017.

[15] Sahar Abubucker, Nicola Segata, Johannes Goll, Alyxandria M. Schubert, et al., *Metabolic Reconstruction for Metagenomic Data and Its Application to the Human Microbiome*, 2012.

[16] Eric A. Franzosa, Lauren J.McIver, Gholamali Rahnavard, Luke R.Thompson, et al. *Species-level functional profiling of metagenomes and metatranscriptomes*, 2018.

[17] Yuzhen Ye, Thomas G. Doak, *A Parsimony Approach to Biological Pathway Reconstruction/Inference for Genomes and Metagenomes*, 2009.

[18] Barbara Di Camillo, Tiziana Sanavia, Matteo Martini, Giuseppe Jurman, et al. *Effect of Size and Heterogeneity of Samples on Biomarker Discovery: Synthetic and Real Data Assessment*, 2012.

[19] Francisco Daniel, Davila Aleman, et al. *Microbiome evolution during host aging*, 2019.

[20] Meyer, P., Alexopoulos, L., Bonk, T. et al., *Verification of systems biology research in the age of collaborative competition*, 2011

[21] Meyer, P., Hoeng, J., Rice, J. J., et al., *Industrial methodology for process verification in research (IMPROVER): toward systems biology verification*, 2012

Acknowledgement

I would like to thank the sbv IMPROVER Team (www.sbvimprover.com) and Philip Morris International for organizing the “Metagenomics Diagnosis for Inflammatory Bowel Disease Challenge” (MEDIC) and for providing the datasets utilised in this thesis [20, 21].