Universitat Oberta de Catalunya

uoc.edu

Supervised methods to classify body composition in HIV-infected patients

Daniel Royo Solé
Máster en Bioinformática y Bioestadística UOC-UB
Área 2-Subárea 2-Análisis de datos

Nuria Perez Alvarez, **Esteban Vegas Lozano**
Carles Ventura Royo

06/2019

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Supervised methods to classify body composition in HIV-Infected patients*
Nombre del autor:	*Daniel Royo Solé*
Nombre del consultor/a:	*Nuria Perez Alvarez, Esteban Vegas Lozano*
Nombre del PRA:	*Carles Ventura Royo*
Fecha de entrega (mm/aaaa):	06/2019
Titulación::	*Máster en Bioinformática y Bioestadística UOC-UB*
Área del Trabajo Final:	*Área 2-Subárea 2-Análisis de datos*
Idioma del trabajo:	Inglés
Palabras clave	*Machine learning, VIH, classification methods*

Resumen del Trabajo:

Desde que se diagnosticó por primera vez a principios de la década del 1980, la investigación sobre el SIDA ha mejorado exponencialmente. Hoy en día, en campos como la bioestadística, donde la biología se encuentra con la estadística avanzada y las matemáticas, las líneas de estudio han cambiado y existe otro enfoque para resolver problemas de investigación científica.

Debido a las mejoras que la investigación ha proporcionado a la calidad de vida de las personas seropositivas, su esperanza de vida es aproximadamente normal, pero desarrollan enfermedades relacionadas con el envejecimiento: sus huesos se vuelven más frágiles, sus músculos se debilitan y su grasa puede estar anormalmente distribuida.

El objetivo de este proyecto es encontrar métodos de clasificación supervisados por aprendizaje automático para detectar enfermedades del envejecimiento en personas infectadas por el VIH.

Así se calcula un modelo de red neuronal significativo (test de "No information rate" p-valor<0.05) utilizando R y, en particular, el paquete mlr, para clasificar con éxito una enfermedad de masa magra asociada con el proceso de envejecimiento, tomando como variables explicativas los datos de masa ósea y masa grasa. La topología y distribución de pesos de la red de este modelo proporciona información sobre las variables, que puede resultar de interés clínico.

Dada la elevada precisión y los parámetros de rendimiento del método de clasificación, se valida la suposición de que se puede predecir una enfermedad de envejecimiento magro por las variables de tejido óseo y graso y, por consiguiente, se logran los objetivos de este proyecto.

Abstract:

Since it was first diagnosed in the early 1980s, the research on AIDS has exponentially improved along the years. Nowadays, in fields like biostatistics where biology meets advanced statistics and mathematics, the study lines have turned and another approach can be made to handle scientific problems.

Due to the successful improvements that research has provided to the quality of life of HIV-positive individuals, their life expectancy is approximately normal, but they develop ageing related diseases: their bones turn to be more fragile, their muscles get weaker, and their fat mass may be abnormally distributed.

The aim of this project is to find machine learning supervised classification methods to detect ageing diseases in HIV-infected individuals, and discuss if these methods provide valuable information regarding the variables of the dataset and how they relate to each other.

Thus, a significative ("No information rate" test p-value < 0.05) Neural Network model is computed using R and in particular the package mlr, to successfully classify a lean mass disease associated with the ageing process, using bone and fat data samples as explicative variables. The topology and weight distribution of this model's network provides information about the most relevant variables, which may be of clinical interest.

Given the elevated accuracy and positive performance parameters of said classification method, it is safe to say that the assumption that a lean mass ageing disease could be predicted by the bone and fat tissue variables is validated, and consequently the goals of this project are achieved.

Contents

List of Figures

List of Tables

Introduction

In the last quarter century technological development has suffered a really fructiferous exponential bump that has delivered really important research material. In addition to the purest theoretical results that research has provided in these years, a new working manner has become more popular: open source materials and multidisciplinary study teams have spread the range that scientific development can reach. Thus fields like material science, quantic computation or biostatistics, that mix techniques, theoretical and practical, from diverse scientific fields have been home to new scientific encounters and are becoming more popular each day.

In particular, biostatistics, the scientific field where biology, mathematics, statistics and computation meet, has been granted with the latest cutting-edge technological advances that of course lead to new investigation lines and fields that had yet to be explored.

One of these brand new topics that has aroused interest amongst the bioscientific community is Machine Learning, and its application in biological problems. From genomic analysis to illness detection, machine learning methods are nowadays used and developed as an efficient numerical and experimental method that helps build conjectures or prove results in a way that had never been done before.

This project is related to the *"Fundació de lluita contra la SIDA"* (FLSIDA), where research teams have already developed analyses around the subject, and the data used in this work has been provided by them. This dataset, provided by FLSIDA consists of different body measurements (DEXAS) from HIV-positive individuals; machine learning techniques will be used to determine if these inviduals are positive or not for certain body composition illnesses. The challenging part comes from the aim to predict a certain tissue-related illness without relying on the data related to this specific tissue. For instance, to try to predict if an individual is lipodystrophic, which means that their fat distribution is abnormal, by analyzing their bone and lean mass data samples.

Context and aim of the project

Since the early 1980s, AIDS has attracted international medical and political attention, which included media coverage, fundings, awareness and stigma prevention campaigns, and numerous scientific investigation lines. Even though the life expectancy of an infected non-treated individual is no longer than 11 years, due to the scientific results that have been developed since it was first detected, if a HIV-positive goes under combination antiretroviral therapy (cART), their life expectancy can get closer to a normal one. As nowadays treated HIV-positive individuals can live just as long as without the disease, they develop age-related illnesses that can make their body tissues age quicker than a non HIV-positive individual (Meir-Shafrir & Pollack, 2012).

HIV-positive people tend to age sooner: their bones may be more fragile, their muscles can be weaker, and their fat mass can turn out abnormally distributed. The detection of these ageing diseases is vital to guarantee the quality of life of HIV-infected patients, and early detection is definitely beneficial to their wellness.

The aim of this project is to find machine learning classification methods to detect ageing diseases in HIV-infected individuals, and discuss if these methods provide valuable information regarding the variables of the dataset and how they relate to each other.

Goals and desired achievements

HIV-positive individuals undergoing cART are proven to develop early ageing in some cases. Since early prevention of the ageing diseases is directly related to their wellness, the goal of this project is to find accurate classification algorithms that can produce predictions with patients that have not been diagnosed yet with said diseases.

In addition, if classification algorithms are found, once they are proven to be accurate and significative they can be analysed in order to know which of the variables of the provided dataset are the ones that have a greater effect on the disease detection.

The final goal of this project would be to deliver an accurate detection algorithm for at least one body tissue, and if the detection is good enough in terms of statistical significance, to provide information about the algorithm and the decisions it makes regarding the variables and their relationships.

Methods

All of this project is developed using `Rmarkdown` and the `R` software engine. The data analysis uses several third party R packages including `mlr`, `haven`, `gmodels` or `NeuralNetTools`. The dataset is provided in `.xls` format, and it is imported directly to R.

The data analysis section consists of an advanced statistical analysis of the dataset. Starting from a naive data visualization, it includes missing value detection and treatment, data verification, outlier detection and variable description and grouping. Before starting the predictive section, as the dataset contains a fair amount of explanatory variables, a feature relationship analysis is made, which consists of correlation results and displays and a principal component analysis using two dimensional plots. Once the feature engineering is done, the results are examined in order to perform dimension reduction to the dataset to avoid redundant features that can lead to poorly trained models.

Once the dimension reduction is computed, the discussion about which supervised classification method to used can begin. The three machine learning methods to compute the classification prediction are Random Forest, XGBoost and Neural Networks.

Planning

The total time to develop and write the full project is approximately eleven weeks, from March 19^{th} to June 5^{th}. To make a reasonable planning the tasks have to fit in four essential categories: Take off, Data analysis, Machine learning, and Additional tasks. For the take off, which includes understanding the background of the project and getting familiar with the bibliography, two weeks were scheduled; the data analysis section is one of the most essential parts of the project, without a nicely treated data no good prediction can be made, thus five weeks are set for this part; the machine learning section, which includes a brief discussion, model training, tuning and comparisons, can be also quite long, and even though it would be better to have more time to develop this subject, four weeks is the time set for this section, overlapping for one week with the previous section. For the additional tasks, that includes two meetings with the project advisors and writing down the body of the thesis no weeks have been counted, but presumely, about 2 to 3 weeks are kept just for the writing part, and the meetings are meant to take about 4 hours in two seeparate days along the development of the project.

Summary: Resulting products

As the main goal of this project was to obtain proper classification methods to predict certain diseases in HIV-infected individuals, the main product is the set of trained and tuned algorithms that achieved a maximum performance when predicting sarcopenia, lipodystrophy and osteoporosis.

Furthermore, since the method with the best performance when predicting the three diseases is neural networks, as computed in Appendix 3, one can analyze the information delivered by the Garson and Olden algorithms to provide a biological or clinic interpretation of how the predictive methods work, and which variables are more have the biggest decision weight in the network. Thus, as a complementary result, being able to interpret the "black box" of the neural network may provide valuable data about said diseases and how they are related to other body tissues.

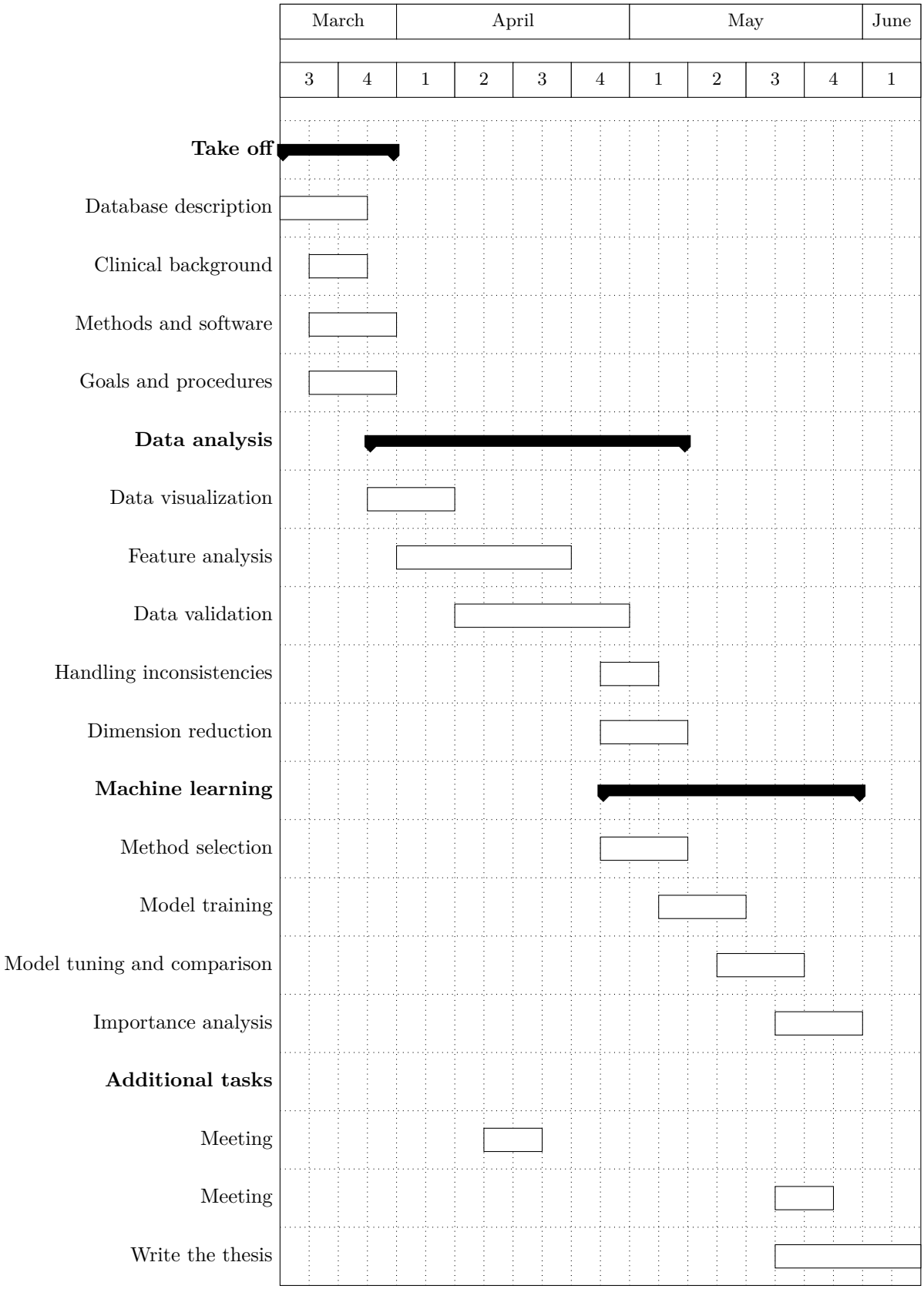


Figure 1: Gantt Chart

Summary: Analysis and procedures

This dissertation is split in two parts: Exploratory data analysis and Machine Learning.

The exploratory data analysis section starts with a chapter about data exploration, where the author gets familiar with the provided dataset and visualizes its variables, and also validates that the dataset is consistent. This chapter is followed by the process called as data clean-up, that starts handling missing and outlier values; once the missing values are fixed and the outliers are detected, the author proceeds studying the relationship and correlations between features, which leads to a dimension reduction process, where the goal is to be able to build a similar dataset with combinations or subsets of the initial variables without loss of information. This data analysis part ends when the author performs a principal component analysis that confirms that a dimension reduction is both feasible and safe.

After reducing the dimension of the dataset the main goal is to determine which supervised classification method would fit the data better, test them and define a candidate model to predict the ageing diseases.

This process starts with the definition of Random Forests, XGBoost and Neural Networks. Each method undergoes a repeated stratified cross-validation that provides the first performance parameters, which turn to be quite decent. Then, since said methods were applied with the default parameters that the package provides, to improve the performance a really time consuming cross-validated tuning process is done, providing the optimal parameters for each method and body tissue.

Once the three best models are set, to end the author provides information about the insights of the neural networks, that relate how relevant are certain features in order to predict certain diseases.

Data exploration

The dataset that has been provided is a DEXAS table with data of 1480 individuals. A DEXA (Dual-Energy X-Ray Absorptiometry) (Ayyappan, Niveditha, & Breur, 2017) is a radiological test that measures bone, lean mass and fat mass distributions and quantities when passing X-rays with two different degrees of energy through the body.

Thus, prior to beginning exploring the data, this dataset has to be carefully examined and precisely cured in order to ease further analyses. This chapter starts with basic data description procedures, followed by a description and classification of the variables. This first section of the chapter follows the regular and typical data exploration that is made in most machine learning scenarios. Thus, the second part of the chapter consists of gathering all the cured data and make, prove or discard assumptions about it in order to make further computations optimal.

Variable exploration

Once one receives a dataset, the first thing to do is to list all the feature names and try to guess what they describe, so the first step towards exploring the provided dataset is to display the variable names that define the table. In addition to know the feature names, it is also useful to know its type, and check if they are numeric, boolean or characters.

```
raw_data <- as.data.frame(read_sav(paste(getwd(),params$file, sep="")))
n <- nrow(raw_data)
dim(raw_data)
```

```
## [1] 1480   82
```

```
table(sapply(raw_data, typeof))
```

```
##
## character    double
##         1        81
```

It looks like the dataset consists of 1480 rows and 82 columns, and only one of them is of type *character*, while the other ones are of type *double* or numeric. This `character` variable seems to be the one regarding the gender of the individuals, and can be left out since there is another categorical variable that specifies the gender in a numerical way, even though some feature checking will have to be made in order to prove that the two variables are equivalent.

Once one gets to see the column names and types of the dataset, it is handy to interpret the variables and try to give them biological meaning, by splitting the columns in subsets. An organic way to classify the explanatory variables is as follows:

- `n.general` Includes all the variables that measure general aspects of the person.
- `n.fat` Includes all the variables that contain measurements of fat distribution and proportion.
- `n.bone` Includes all the variables that contain measurements of bone density and mass.
- `n.lean` Includes all the variables that contain measurements of lean mass distribution and proportion.

General variables

The provided dataset lists, as explained, data taken from the DEXAS test. This does not mean that it does not include any more features about the individuals, since the dataset also contains variables that explain the gender (binary and text), weight (Kg), height (cm) and age (absolute and grouped) of the patients, as well as the BMI (body mass index), defined as the body mass divided by the square of the body height, which is universally expressed in units of kg/m^2.

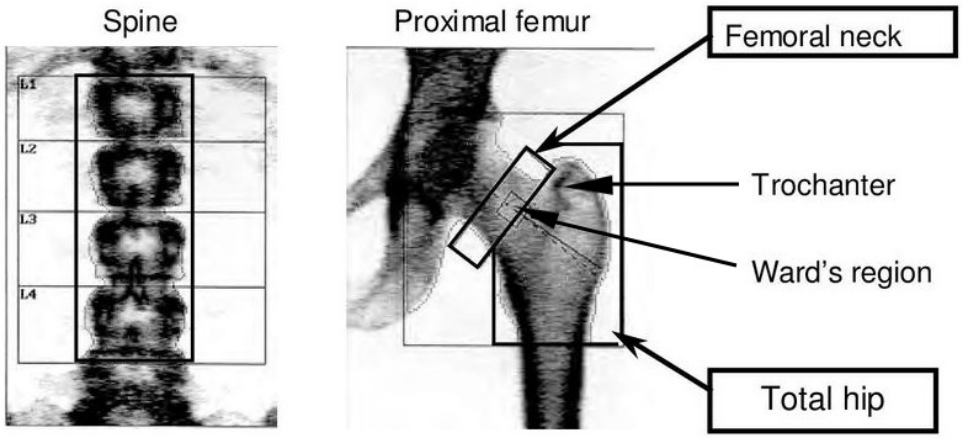


Figure 2: Parts of the lower spine and the femur.

$$general = \{gender, gender\_num, Height, Weight, Age, Age\_cat, BMI\}$$

As specified before, one could think of removing redundant variables such as `gender`, that is also a character array, and wether `Age` or `Age_cat`, since as can be seen in Table 1 it looks like these columns are equivalent.

Table 1: Some samples of the age and gender related columns

Age	Age_cat	gender	gender_num
40	0	F	2
35	0	F	2
33	0	F	2
28	0	F	2
36	0	F	2
45	0	F	2

$$general = \{gender\_num, Height, Weight, Age, BMI\}$$

Bone variables

The DEXAS test leaves a quite sizeable group of bone-related variables. To explain them, it is handy to pack them in subsets, since their biological meaning is related in most cases.

As displayed in Figure 2 (Cosman et al., 2014), (Elkoushy, Jundi, Lee, & Andonian, 2014), all the data beginning with L_i, where i is an integer is referred to the vertebrae; in this case, the vertebrae data that is provided refers to the first lower four. As for the remaining preffixes, as shown as well in the image, `NeckF` stands for the neck area of the femur, `Wards` stands for the Wards triangle, and `Troch` is related to the Trochanter section of the femur. Combinations of vertebrae such as `L1L4` are the result of dividing the value of the data taken from the first by the value of the data taken from the last. And of course, the preffix `Total` lists the mean data of the whole bone tests of that individual, and the preffix `TotalF` gathers the mean of the samples related to the femur.

The suffixes, T, and Z, list the T-score and Z-score of the specific part of the bone the preffix sets. The Z score provides a comparison with healthy subjects of the same age, while the T score provides a comparison with healthy young adults (20-29 years). The intent of the T score for adults is not to provide an age-matched comparison, but to estimate the loss of bone relative to peak density.

For adults, clinical evaluations of bone status are based on BMD, defined as the ratio of bone mineral content (BMC) to the two-dimensional projected image of bone area (BA). The size and geometry of the adult skeleton remains relatively stable over many years; thus, BA remains relatively constant for the individual. BMD has a lower variance than BMC; therefore, it has evolved as the bone parameter to assess abnormal conditions. (Ellis et al., 2001)

Thus, the features contained in the dataset that make reference to bone samples are the following:

`TotalBMD, L1BMD, L2BMD, L3BMD, L4BMD, L1L4BMD, L2L4BMD, NeckFBMD, WardsBMD, TrochBMD, TotalFBMD, L1T, L2T, L3T, L4T, L1L4T, L2L4T, NeckFT, WardsT, TrochT, TotalFT, L1Z, L2Z, L3Z, L4Z, L1L4Z, L2L4Z, NeckFZ, WardsZ, TrochZ, TotalFZ`.

All of these variables are in fact numeric and non-binary.

Lean variables

This set of variables lists the ones that are strictly related to the lean mass distribution in the body. To describe them, it is handy to split them in two parts: the ones referring to specific body parts, and the ones that are lean mass related parameters.

The first subset, the one defined as the group of lean mass related variables of specific body parts, consists of the following features: `RALg, LALg, BothALg, RLLg, LLLg, BothLLg`.

The names of these features are built as follows: The first block is wether `L,R, Both`, that stands for left, right and the sum of the both sides. The next block is one of the following values: `A, L` that stand for arm and leg respectively. This is followed by a capital `L`, for "lean", and finally a `g` that explains that the measurement is made in grams.

The second subset, that gathered lean mass related parameters, contains `FFMI, Apendicularleanmas, TLg, TotalLg`.

`TLg` and `TotalLg` indicate the value of lean mass in grams in the trunk and in the whole body respectively.

The nutritional status of a patient is typically determined from the use of body mass index (BMI), however there are numerous cases where BMI may not accurately reflect the actual composition of the patient. Fat free mass index (FFMI) might offer a better representation since it includes an actual estimate of body composition in the equation (FFMI=FFM (kg/m^2)). (Loenneke et al., 2012)

Finally, the appendicular lean mass is a frequently used indicator regarding the amount of lean mass that can be found in the limbs of an individual, including both arms and legs.

Fat variables

Just as the lean related variables, there are as well two defined subsets for the fat set. The first one, that groups variables that define fat proportion in specific parts of the body follow the same name rules as the lean set: `RAFp, RAFg, LAFp, LAFg, BothAFp, RLFp, RLFg, LLFp, LLFg, BothLFp`. In this case, the last character can be a `g` (for grams) or a `p`. The ones ending with a `p` include the specific data expressed as a percentage.

There is also a subset that groups relationships between variables, such as: `FTrunkgFLegsg, FtrunkpFlimbsp, FtrunkgFtotalg, FLegsgFtotalg, FlimbsgFtotalg`, just as, for example, the `L1L4` variable that was defined in the bone set.

`TFp, TFg, TotalFp, TotalFg` like in the lean set, list data for both the trunk and the whole body, and finally the two last variables `FMR`, and `FMI` are DEXA indicators for the fat tissue. Fat mass ratio (FMR) is

defined as the ratio between the percent of the trunk fat mass and the percent of the lower-limb fat mass (Freitas et al., 2010). FMI (fat mass index) was first introduced in a study involving nutritional assessment (VanItallie, 1990) and is calculated by taking the body fat mass component from BIA and dividing by height squared (Peltz, Aguirre, Sanderson, & Fadden, 2010).

```
n.general <- c('Age', 'Height', 'Weight', 'BMI')
n.fat <- c('RAFp', 'RAFg', 'LAFp', 'LAFg', 'BothAFp',
           'RLFp', 'RLFg', 'LLFp', 'LLFg', 'BothLFp',
           'TFp', 'TFg', 'TotalFp', 'TotalFg',
           'FMR', 'FTrunkgFLegsg', 'FtrunkpFlimbsp', 'FtrunkgFtotalg',
           'FLegsgFtotalg', 'FlimbsgFtotalg', 'FMI'
          )
n.bone <- c('TotalBMD', 'L1BMD', 'L1T', 'L1Z', 'L2BMD',
             'L2T', 'L2Z', 'L3BMD', 'L3T', 'L3Z', 'L4BMD', 'L4T',
             'L4Z', 'L1L4BMD', 'L1L4T', 'L1L4Z', 'L2L4BMD', 'L2L4T',
             'L2L4Z', 'NeckFBMD', 'NeckFT', 'NeckFZ', 'WardsBMD',
             'WardsT', 'WardsZ', 'TrochBMD', 'TrochT', 'TrochZ',
             'TotalFBMD', 'TotalFT', 'TotalFZ')
n.lean <- c('RALg','LALg','BothALg','RLLg','LLLg','BothLLg','TLg',
             'TotalLg','FFMI','Apendicularleanmas')
```

Data verification

The first step towards verifying the dataset, and measuring the quality of the data is to look for empty cases, and if some, empty variables.

```
##                  ID              gender          gender_num
##                   0                   0                   0
##             TotalBMD              Height                RAFp
##                  12                   2                   0
##                RAFg                RALg                LAFp
##                   0                   0                   0
##                LAFg                LALg              BothAFp
##                   0                   0                   0
##              BothAFg              BothALg                RLFp
##                   0                   0                   0
##                RLFg                RLLg                LLFp
##                   0                   0                   0
##                LLFg                LLLg              BothLFp
##                   0                   0                   0
##              BothLFg              BothLLg                 TFp
##                   0                   0                   0
##                 TFg                  TLg             TotalFp
##                   0                   0                   0
##              TotalFg             TotalLg               L1BMD
##                   0                   0                   1
##                 L1T                 L1Z               L2BMD
##                   1                   1                   1
##                 L2T                 L2Z               L3BMD
##                   2                   1                   1
##                 L3T                 L3Z               L4BMD
##                   2                   2                   2
##                 L4T                 L4Z             L1L4BMD
##                   2                   3                   1
```

```
##                L1L4T                L1L4Z              L2L4BMD
##                    1                    1                    2
##                L2L4T                L2L4Z             NeckFBMD
##                    2                    2                    0
##              NeckFT               NeckFZ             WardsBMD
##                    0                    0                    1
##              WardsT               WardsZ             TrochBMD
##                    1                    1                    1
##              TrochT               TrochZ            TotalFBMD
##                    1                    2                    0
##             TotalFT              TotalFZ               Weight
##                    0                    0                    0
##                 BMI                  FMI                 FFMI
##                    2                    2                    2
##   Apendicularleanmas                  FMR        FTrunkgFLegsg
##                    2                    0                    0
## Indexdistributionfat      FtrunkpFlimbsp       FtrunkgFtotalg
##                    0                    0                    0
##        FLegsgFtotalg      FlimbsgFtotalg            LLegFgBMI
##                    0                    0                    2
##           LLegFpBMI                  Age              Age_cat
##                    2                    0                    0
##        Lipodistrophy           Sarcopenia           LipoSarcop
##                    0                    2                    2
##             BMI_cat            phenotype            minTscore
##                    2                    2                 1480
##          Tscore_3cat
##                 1480
```

Even though the issues around missing data are well-documented, it is common practice (Pampaka, Hutcheson, & Williams, 2016) to ignore missing data and employ analytical techniques that simply delete all cases that have some missing data on any of the variables considered in the analysis. It is proven that as long as the missing data rows do not define a big portion of the dataset, even though machine learning techniques that apply data imputation are popular and effective, the odds of encountering loss of information can be severe and may not even be obvious to the analyst.

Prior to handling missing data, the missing features have to be built. Since the only columns that are completely empty are `minTscore` and `Tscore_3cat`, and can be built using data from other columns, in order to treat missing data, it is convenient to first compute the missing colums.

Medical conditions and missing data

It is easy to notice that there are two empty columns, regarding the variables `minTscore` and `Tscore_3cat`. Since these variables can be obtained through simple calculations including the remaining variables, one would proceed filling in the empty columns as follows:

$$\text{minTscore} = \min(\text{L1T, L2T, L3T, L4T, NeckFT, WardsT, TrochT})$$

$$Tscore\_3cat = \begin{cases} 0 & if\ minTscore \geq -1 \\ 1 & if\ -1 > minTscore \geq -2.5 \\ 2 & if\ -2.5 > minTscore \end{cases}$$

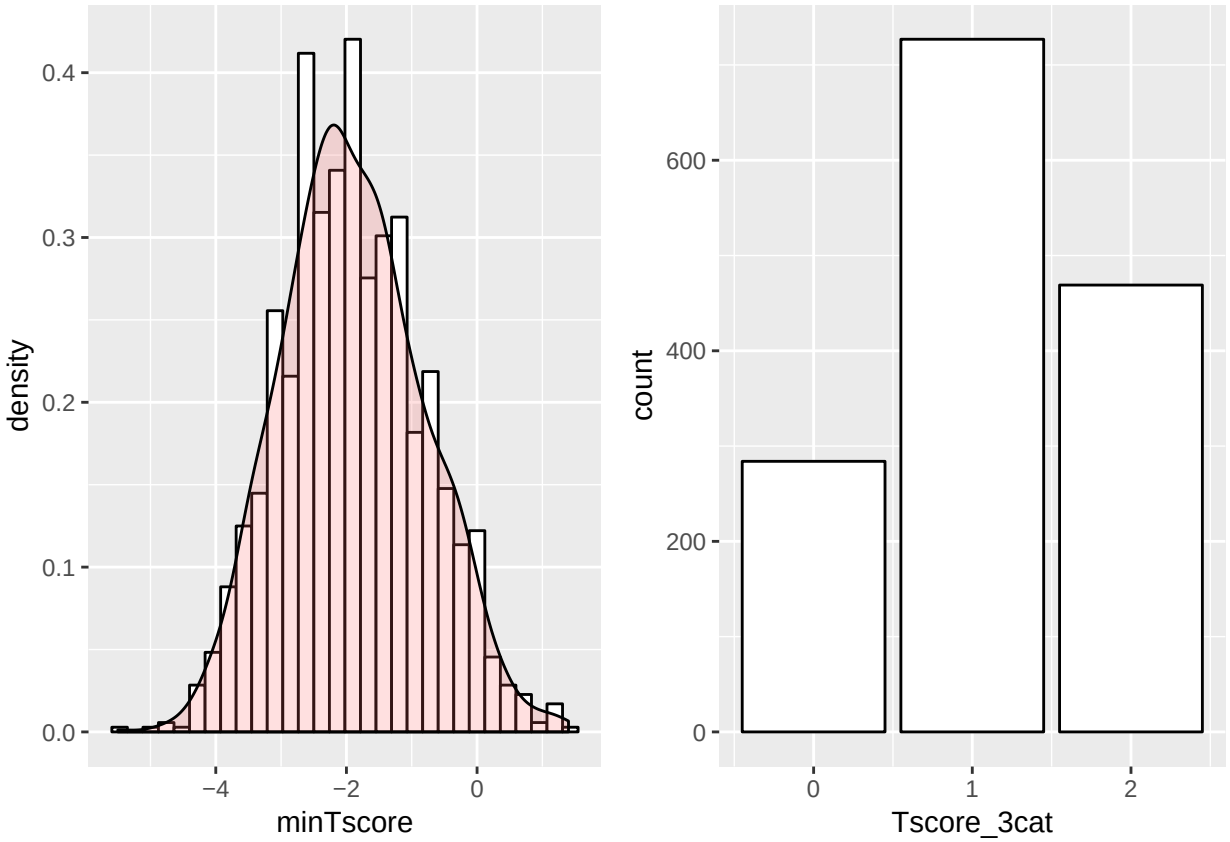


Figure 3: Distribution of minTscore and Tscore_3cat

Table 2: New columns and related variables

L1T	L2T	L3T	L4T	NeckFT	WardsT	TrochT	minTscore	Tscore_3cat
-1.3	-1.7	-2.0	-2.3	-1.0	-1.4	-1.0	-2.3	1
-0.6	-0.4	-0.1	-0.7	0.6	0.9	0.0	-0.7	0
0.1	0.7	1.1	0.1	0.4	0.3	-0.1	-0.1	0
-2.3	-1.7	-1.0	-0.8	-0.9	-0.7	-1.4	-2.3	1
-1.4	-0.8	-0.5	-0.9	-0.5	-0.8	-1.0	-1.4	1
-2.8	-2.4	-1.6	-2.6	-1.6	-1.7	-1.4	-2.8	2

As displayed in Table 2, `minTscore` and `Tscore_3cat` are now non-empty and their value is correct, as can be seen comparing the bone variables and the resulting columns.

Once these new variables are built, it is adequate to make a quick variable description to see how they are distributed in Figure 6.

Now that the originally null columns are filled, two more columns can be built. Osteoporosis and osteopenia are defined through the minimum T-score in a patient: *"As defined by the World Health Organization (WHO), osteoporosis is present when BMD is 2.5 SD or more below the average value for young healthy women (a T-score of < -2.5 SD). A second, higher threshold describes "low bone mass" or osteopenia as a T-score that lies between -1 and -2.5 SD. "Severe" or "established" osteoporosis denotes osteoporosis that has been defined in the presence of one or more documented fragility fractures."* (Sözen, Özişik, & Başaran (2017))

```
raw_data[raw_data$Tscore_3cat==2, 'Osteoporosis'] <- 1
raw_data[raw_data$Tscore_3cat!=2, 'Osteoporosis'] <- 0

raw_data[raw_data$Tscore_3cat!=1, 'Osteopenia'] <- 0
raw_data[raw_data$Tscore_3cat==1, 'Osteopenia'] <- 1
```

As a consequence of building four new features in the dataset no more empty columns remain, thus it is appropriate to handle the remaining isolated empty values. Since the empty values belong to a tiny portion of the dataset, one can think about wether to remove them or fill them with mean values. Missing data engineering is definitely not a trivial problem, and even though imputation methods (which are way more complex than just applying means or medians) can be applied (Pampaka et al., 2016), for the sake of simplicity and without losing major information, the rows that contain empty values will be erased, because there are rows that contain a lot of empty results and filling them in with mean values would be redundant.

```
raw_data <- raw_data[complete.cases(raw_data),]
```

After this first data clean up, 22 rows have been removed, which means that a 1.486% of the number of rows in the original dataset has been discarded, which is a fairly acceptable quantity of rows to be removed, resulting in a datset of 1458 observations.

Medical conditions and data validation

Just as the `Tscore_3cat` variable, that defines both osteoporosis and osteopenia, has been computed, the two other illness-related binary variables have to be validated in order to check that the data that has been provided is consistent and reliable.

The medical condition known as lipodystrophy is described as the inabillity to normally produce and maintain fat tissue, and consequently providing abnormal fat mass distributions, or as described in (Savage, Petersen, & Shulman (2007)), *"The lipodystrophic syndromes encompass a rare group of conditions characterized by partial or complete absence of adipose tissue. The disorders may be genetic or acquired and are further classified according to the anatomic distribution of the lipodystrophy."*. The medical definition based on numbers is stated by using the Fat Mass Ratio, and it works different in terms of gender, since the critical value is of 1.24 for men and 0.95 for women, for a HIV-positive population. (Grenha et al., 2018)

This same exploration should be done with the lean-related medical condition. The variable regarding a medical condition related to the lean mass is `Sarcopenia`, but since there is no data of the strength given by the muscles in the dataset, it would be incorrect to talk about Sarcopenia. Instead, even though the variable name will be kept as `Sarcopenia`, what will be measured is the *low quantity or quality muscle mass*, defined as Appendicular Lean Mass Index/height^2. The critical values may differ depending on what biography is relied on. The article Echeverría et al. (2018) suggests 7.26 and 5.5 as the critical values for male and female respectively. Since these values are not strictly fixed and can vary depending on the status of the measured population, the variable `Lipodistrophy` will be taken as correct and will establish the critical values.

So, as proven in Figure 4, one can take 7.225 kg/m^2 for men and 5.67 kg/m^2 for women as the critical values of apendicular lean mass for the measured lean disease in the dataset, which are in fact the minimum values in the healthy set for each gender.

As a quick description in Figure 5, one can easily see that the lean-related medical condition is much less common than Lipodystrophy, but still a good amount of individuals have the condition, so a proper analysis can be done with this data.

To summarize, the mosaic plot in Figure 6 contains the proportion of normal and non-normal individuals in the sets of Sarcopenia, Lipodystrophy and Osteoporosis. As displayed in Figure 5 it is clear that the amount of lipodystrophy-positive individuals is greater than the normal ones, and by looking at the mosaic plot in Figure 6, one can see that the biggest subset is the one with lipodystrophy positive, osteoporosis

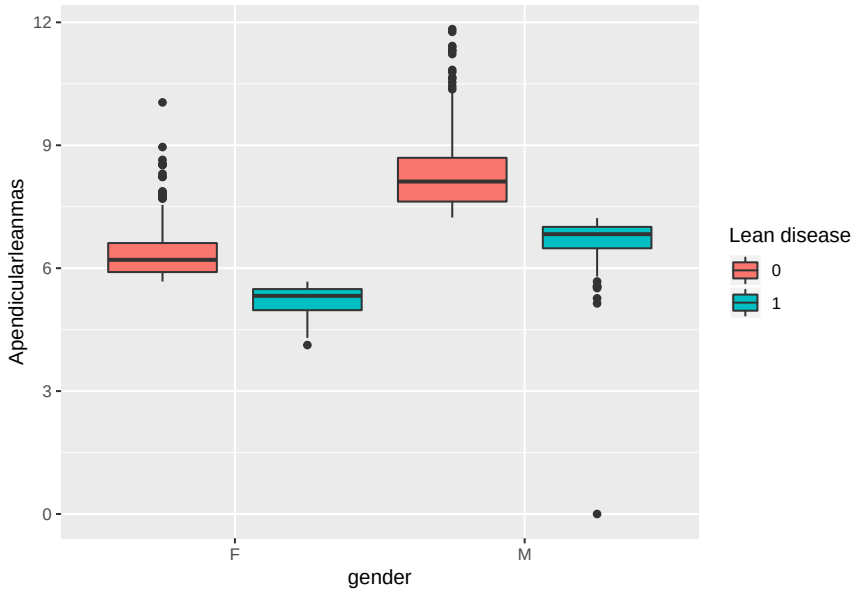


Figure 4: Distribution of the lean disease for both male and female

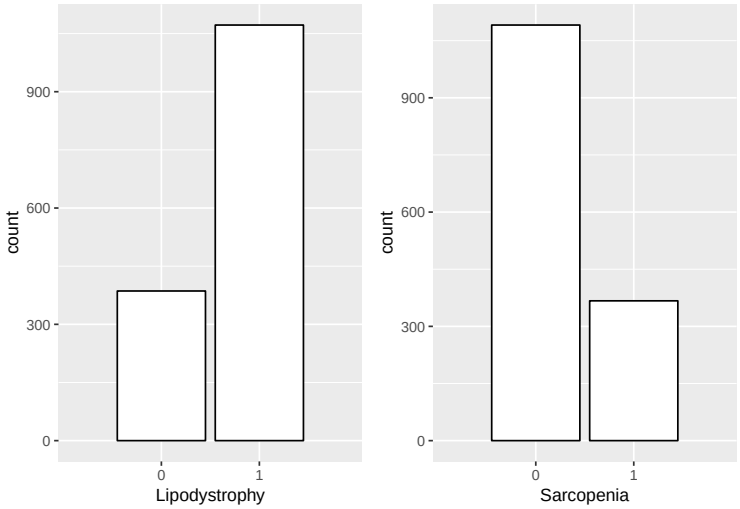


Figure 5: Distribution of Lipodystrophy and the lean-related illness within the individuals

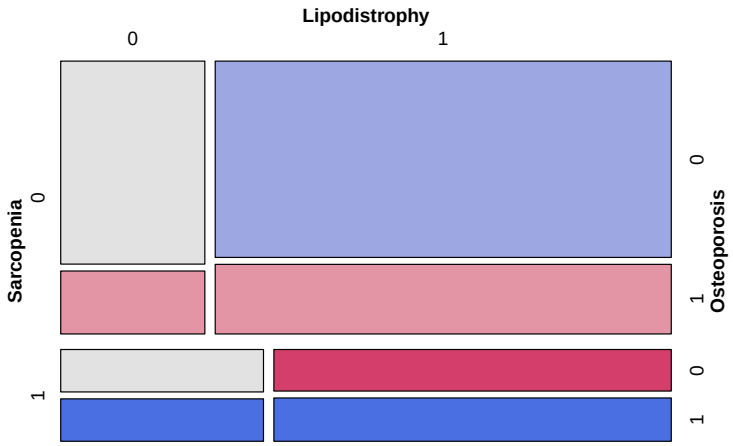


Figure 6: Combined distribution of Sarcopenia, Osteoporosis and Lipodystrophy

negative, lean-related illness negative individuals. On the other hand, the smallest subset is the one containing lipodystrophy negative, osteoporosis positive, lean-related illness positive individuals.

Descriptive variable visualization

One of the most naive steps towards getting familiar with the dataset is to start analyzing the distribution of the continuous and discrete variables, and checking the outlier values of the continuous ones. If outlier values are found, one should send them to the clinical team in order to decide whether it is nice to ignore them or let the clinical team fix the data. To check the outliers one can just calculate them numerically, even though a graphical visualization is a quicker solution to determine if there are or are not outliers for every variable. In Appendix 1 there is the graphical representation of every continuous and discrete feature in the dataframe, which helps detecting outliers rapidly.

In order to study the distribution of the features two subsets have to be defined, one for the continuous variables, and another for the discrete ones.

```
cont.vars <- c('TotalBMD', 'Height',
               'RAFp', 'RAFg', 'RALg', 'LAFp', 'LAFg',
               'LALg', 'BothAFp', 'BothAFg', 'BothALg', 'RLFp',
               'RLFg', 'RLLg', 'LLFp', 'LLFg', 'LLLg',
               'BothLFp', 'BothLFg', 'BothLLg', 'TFp', 'TFg',
               'TLg', 'TotalFp', 'TotalFg', 'TotalLg','L1BMD',
               'L1T', 'L1Z','L2BMD', 'L2T', 'L2Z',
               'L3BMD', 'L3T', 'L3Z','L4BMD', 'L4T' ,
               'L4Z', 'L1L4BMD','L1L4T','L1L4Z', 'L2L4BMD',
               'L2L4T','L2L4Z','NeckFBMD', 'NeckFT', 'NeckFZ',
               'WardsBMD', 'WardsT', 'WardsZ','TrochBMD', 'TrochT' ,
               'TrochZ', 'TotalFBMD', 'TotalFT', 'TotalFZ', 'Weight',
               'BMI', 'FMI', 'FFMI', 'Apendicularleanmas', 'FMR',
               'FTrunkgFLegsg', 'Indexdistributionfat', 'FtrunkpFlimbsp',
               'FtrunkgFtotalg', 'FLegsgFtotalg',
               'FlimbsgFtotalg', 'LLegFgBMI', 'LLegFpBMI', 'Age')

disc.vars <- c("gender", "Age_cat", "Lipodistrophy", "Sarcopenia",
```

```
                "Osteoporosis", "BMI_cat")
```

As one can see in Appendix 1, there are quite a few variables with a non-normal representation, which is acceptable, even though some of them have extreme values which is a sign of either wrong data or rare samples. Either way, these extreme values have to be sent to the clinical team in order for them to correct or verify them for further studies. In this specific work no data has been sent to the clinical team due to time limitations, but in case further analysis has to be done this would definitely be required to proceed.

In the following code output one can see the complete list of potential outlier values for every continuous variable in the dataset. In the materials section the *.xls* file containing these values can be found, and it is the exact same file that would be sent to the clinical team.

```
##              TotalBMD                Height                  RAFp
##                    17                     1                    17
##                 RAFg                  RALg                  LAFp
##                    36                    13                    20
##                 LAFg                  LALg                BothAFp
##                    38                    14                    19
##               BothAFg               BothALg                  RLFp
##                    36                    13                    12
##                 RLFg                  RLLg                  LLFp
##                    30                     6                    12
##                 LLFg                  LLLg                BothLFp
##                    30                     7                    12
##               BothLFg               BothLLg                   TFp
##                    29                     5                     0
##                  TFg                   TLg                TotalFp
##                    11                    13                     6
##               TotalFg               TotalLg                 L1BMD
##                    26                     9                    16
##                 L1T                   L1Z                  L2BMD
##                    16                     8                    16
##                 L2T                   L2Z                  L3BMD
##                    14                    13                    15
##                 L3T                   L3Z                  L4BMD
##                    11                     9                    15
##                 L4T                   L4Z                L1L4BMD
##                    15                    10                    12
##               L1L4T                 L1L4Z                L2L4BMD
##                    11                    13                    10
##               L2L4T                 L2L4Z               NeckFBMD
##                    11                    16                    12
##              NeckFT                NeckFZ               WardsBMD
##                    18                    24                    15
##              WardsT                WardsZ               TrochBMD
##                    16                    29                    16
##              TrochT                TrochZ              TotalFBMD
##                    12                    17                    19
##              TotalFT               TotalFZ                Weight
##                    11                    16                    10
##                 BMI                   FMI                  FFMI
##                    32                    31                    16
##   Apendicularleanmas                   FMR          FTrunkgFLegsg
##                    10                    60                     6
## Indexdistributionfat       FtrunkpFlimbsp       FtrunkgFtotalg
```

```
##                  40                  58                   6
##        FLegsgFtotalg       FlimbsgFtotalg           LLegFgBMI
##                   3                   5                  14
##           LLegFpBMI                 Age
##                   9                  13
```

These outlier values can be explained in different ways, but since records of this data can be quite old, some of the data was transcripted by hand from the DEXAS tool to the database. Thus, data inconsistencies and voids were frequent, and before starting to analyze the data itself one has to check and prove that the data is indeed consistent and that related variables match as they should. For instance: the left part, and the right part of the body should contain similar values.

In the following code chunk several comparison operations are made. The aim of these comparisons is to prove that the data provided is consistent and that no major transcription errors have been made. The comparisons that are computed in the following code chunk are:

$$L_1BMD + L_2BMD + L_3BMD + L_4BMD \equiv L_1L_4BMD$$

$$gender \equiv gender\_num$$

$$Age \equiv Age\_cat$$

$$LAFp \equiv RAFp, \quad LAFg \equiv RAFg, \quad LALg \equiv RALg, \quad LLFp \equiv RLFp, \quad LLFg \equiv RLFg, \quad LLLg \equiv RLLg$$

$$BothAFp \equiv LAFp + RAFp, \quad BothAFg \equiv LAFg + RAFg, \quad BothALg \equiv LALg + RALg$$

```
data <- as.data.frame(raw_data)
for (i in 1:n) {
  # L1LABMD = sum(LiBMB)
  Li <- mean(as.numeric(data[i, c("L1BMD", "L2BMD", "L3BMD", "L4BMD")]))
  data.cons$errLi[i] <- abs(Li - data[i, 'L1L4BMD'])

  # Gender must be properly transformed
  data.cons$errGender[i] <- ifelse(data$gender[i] == 'M', 1, 2) - data$gender_num[i]

  # Age must be properly categorized
  data.cons$errAge[i] <- ifelse(data$Age[i] < 50, 0, 1) - data$Age_cat[i]

  # Left and right part of the body have to be similar
  data.cons$errXAFp[i] <- abs(data$LAFp[i] - data$RAFp[i])
  data.cons$errXAFg[i] <- abs(data$LAFg[i] - data$RAFg[i])
  data.cons$errXALg[i] <- abs(data$LALg[i] - data$RALg[i])
  data.cons$errXLFp[i] <- abs(data$LLFp[i] - data$RLFp[i])
  data.cons$errXLFg[i] <- abs(data$LLFg[i] - data$RLFg[i])
  data.cons$errXLLg[i] <- abs(data$LLLg[i] - data$RLLg[i])

  # BothAFp BothAFg BothALg
  data.cons$errBothAFp[i] <- abs(data$BothAFp[i] - data$LAFp[i] - data$RAFp[i])
```

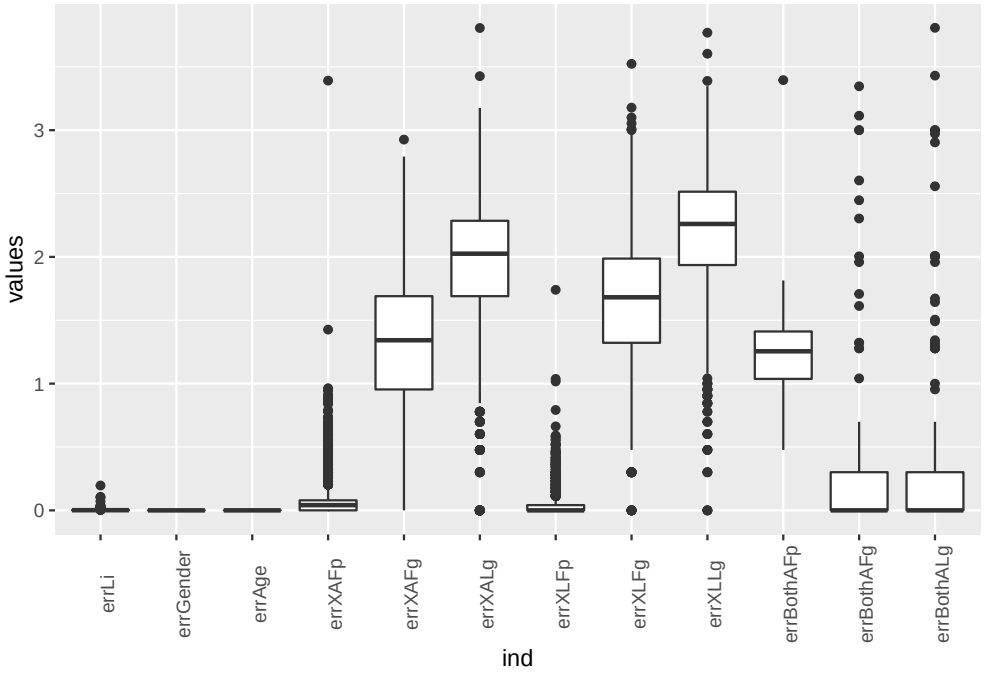


Figure 7: Logarithmic distribution of the errors between related variables

```
  data.cons$errBothAFg[i] <- abs(data$BothAFg[i] - data$LAFg[i] - data$RAFg[i])
  data.cons$errBothALg[i] <- abs(data$BothALg[i] - data$LALg[i] - data$RALg[i])
}
```

From this first bloxpot set in Figure 7, one can make some assumptions. Since a base 10 logarithmic transformation has been made in order to make it easy to see how the errors behave, one of the easiest assumptions to make is that there is a quite relevant amount of outliers, that can possibly be a result of a bad data transcription. On the other hand, another fact that comes easily to the eye is that there are several variables (`errXAFg, errXALg, errXLFg, errXLLg, errBothAFp`) that their mean is far from zero. Maybe, if one tries to scale the data for these variables, the assumption that the left and the right variables follow different scales can be proven. To do so, the `scale()` function will be applied to certain variables, and another boxplot will be shown.

Even though there are still some outlier values, it becomes clear in Figure 8 that once being scaled the variables behave as expected. This means that the clinical team should review them and return to us an updated dataset with equal scales and fixed data for all similar variables.

Data clean-up

It was just proved that the variables, once scaled, behave as expected, thus one can now check how features behave together, and check their correlations. To do so it is useful to analyze these correlations to possibly apply a dimension reduction of the main dataset, by seeing which and how many variables could be redundant. Before proceeding to the next steps, the continuous features in the dataset are scaled in the following code chunk using the R function `scale`, that first centers the data distribution at 0 and then scaling is done by dividing the (centered) columns of the data frame by their standard deviations, standarizing distributions for all the continuous features.

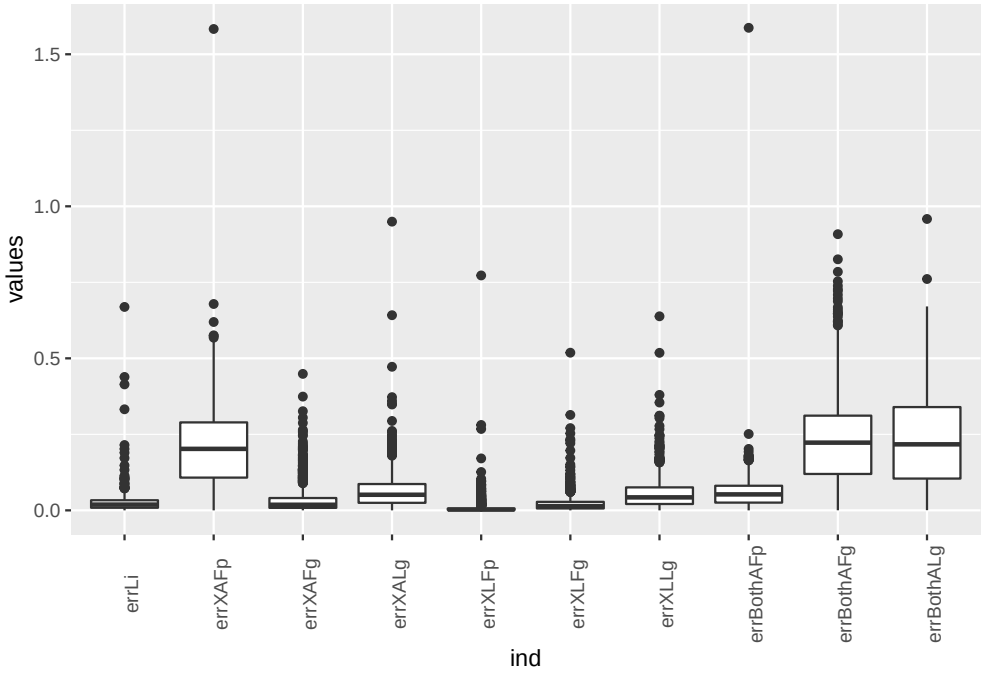


Figure 8: Logarithmic distribution of the errors between scaled related variables

```
raw_data[,cont.vars] <- scale(raw_data[,cont.vars])
data <- as.data.frame(raw_data)
```

Dimension reduction

As explained in Fodor (2002), *"one of the problems with high-dimensional datasets is that, in many cases, not all the measured variables are "important" for understanding the underlying phenomena of interest. While certain computationally expensive novel methods can construct predictive models with high accuracy from high-dimensional data, it is still of interest in many applications to reduce the dimension of the original data prior to any modeling of the data".*

So, it is trivial that in a bi to multivariate way, variables to be reduced together will have to be correlated. To start this analysis in the most naive way, and to have a global map of how the features work together it is a good idea to plot a bivariate correlation plot to check which variables are directly correlated and see if the classification in four subsets (general, fat, bone and lean) makes sense.

The correlation plot in Figure 9, even though is too big and not very specific lets one see how the variables are clearly classified in groups, as previously suggested. The correlations within the groups are usually high (or negatively low), which makes sense, and it looks like feature reduction is without a doubt feasible.

Another way to display related variables and map it graphically is through the `qgraph` function, from the homonymous package, that creates the network graph representation of the pearson correlations matrix plotted in Figure 9. The same representation obtained in the traditional correlation heat plot can be seen in Figure 10, and the same four groups are clear, even though it is not the clearest representation method of the data.

From both Figure9 and Figure 10 one can make the assumption that the lean and fat sets are lightly related, maybe not enought to apply feature reduction between sets but it is clear that their clusters are explicitly related in the network representation.

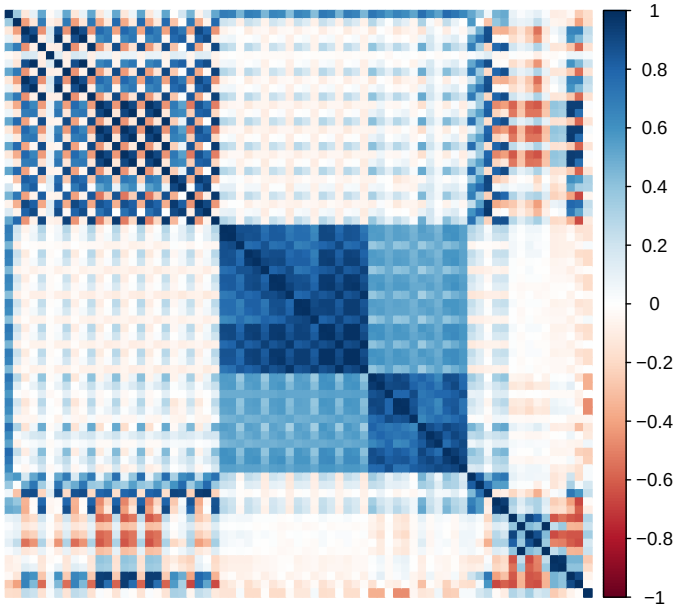


Figure 9: Correlation plot including all the continuous variables in the dataset.

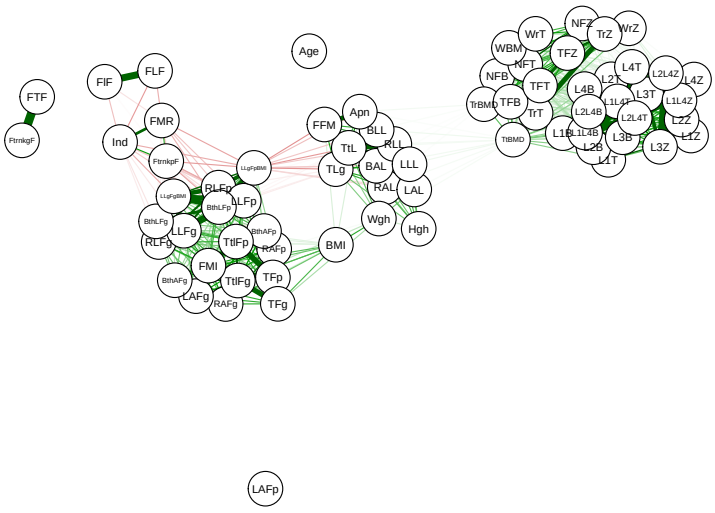
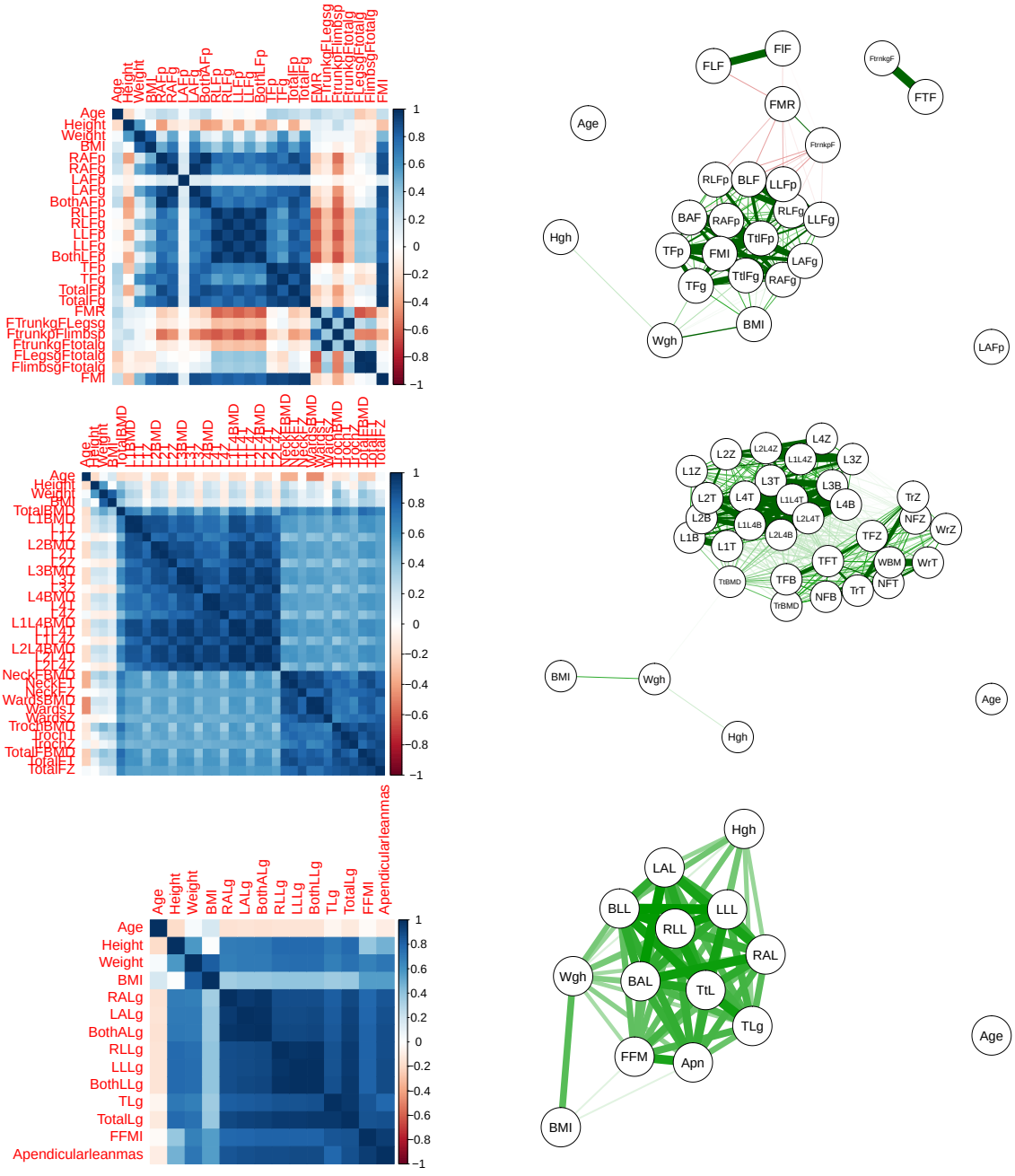


Figure 10: Correlation network graph using `qgraph()`

Since it becomes hard to visualize and describe the feature grouping phenomena that is happening within the subsets, one could think about restarting this process and repeating it for each group in the dataset.



From this big amount of plots and graphs one can take some conclusions:

- It looks like some features are globally correlated.
- It looks like there are features that provide little to no value to the prediction.
- It looks like certain features that are strong in some predictions can be weak in other predictions.
- It looks like it is possible to remove features without damaging the model to avoid redundant relationships.
- It looks like it will be hard to apply some dimension reduction when predicting the lean disease.

Principal component analysis

From this first analysis using correlation plots and graphs, it is hard to decypher specific multivariate relationships beyond the bivariate obvious pairing that can be seen in both the network graph and the correlation plot. This is why a little bit of data engineering is required in order to proceed reducing the deimension of the classification problem.

> *"Principal component analysis (PCA) is the best, in the mean-square error sense, linear dimension reduction technique being based on the covariance matrix of the variables, [...]. In various fields, it is also known as the singular value decomposition (SVD) , the Karhunen-Lohe transform, the Hotelling transform, and the empirical orthogonal function (EOF) method. In essence, PCA seeks to reduce the dimension of the data by finding a few orthogonal linear combinations (the PCs) of the original variables with the largest variance. The first PC, SI, is the linear combination with the largest variance. We have* $s_1 = x^t w_1$ *, where the pdimensional coefficient vector* $w_1 = (w_{1,1}, ..., w_{1,p})$ *solves* $w_1 = arg\ max_{||w=1||} Var(x^t w)$.
>
> *The second PC is the linear combination with the second largest variance and orthogonal to the first PC, and so on. There are as many PCs as the number of the original variables. For many datasets, the first several PCs explain most of the variance, so that the rest can be disregarded with minimal loss of information. Since the variance depends on the scale of the variables, it is customary to first standardize each variable to have mean zero and standard deviation one. After the standardization, the original variables with possibly different units of measurement are all in comparable units."*
>
> Fodor (2002)

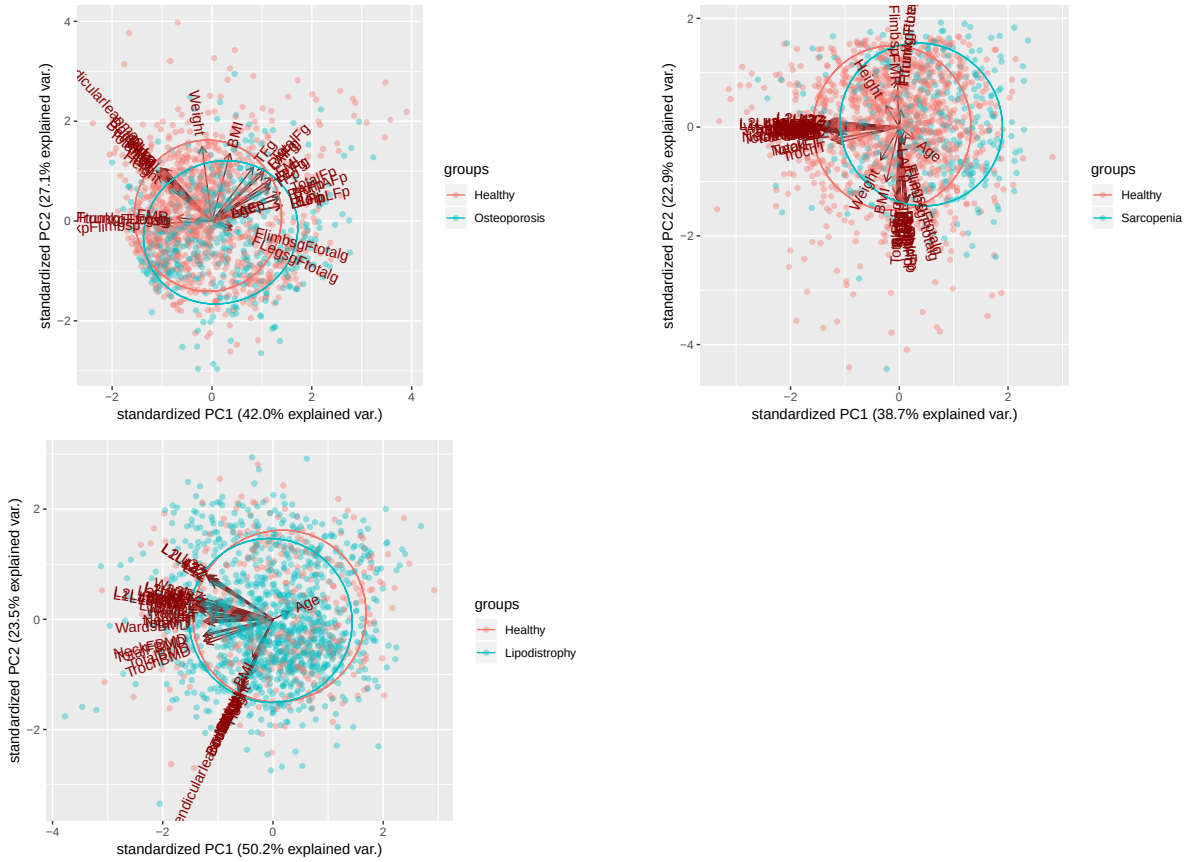
The idea of doing a PCA is to keep the most important PCs and use them as explanatory variables. As the number of "relevant enough" (in terms of variance effect) PCs is always less or equal to the number of original features, this implies necessairly a dimension reduction but also a less interpretable dataset, and interpretability is something bad to lose.

> *"Although they are uncorrelated variables constructed as lineal combinations of the original variables and have some desirable properties, they do not necessarily correspond to meaningful physical quantities. In some cases, such loss of interpretability is not satisfactory to the domain scientists.*
>
> *As an alternative way to reduce the dimension of a dataset using PCA [...] instead of using the PCs as the new variables, this method uses the information in the PCs to find important variables in the original dataset."*
>
> Fodor (2002)

As suggested in Fodor (2002), the other option is to, for example, graphically display the most important PCs, and analyze how variables are grouped. A way to do so is to plot it using two dimensional plts comparing by pairs the principal components.

Using the `ggbiplot` of the homonymous package, available on github (Tang, Horikoshi, & Li, 2016) one can compare two PCs. For simplicity, and just to prove the hypotheses made with the bivariate correlation plots and graphs, only the first two PCs will be compared. The idea would be to first calculate $0 < k \leq$ number of features as the number of "relevant enough" PCs to compare, and compare them two by two in a biplot.

The following figures are the representation of the results of applying `ggbiplot()` to the general subset with fat, bone and lean subsets two by two. The plot is a 2-dimensional representation of the plane generated by the eigenvectors related to the two first PCs, that have in fact the highest eigenvalues, and allows to easily describe which variables behave alike in this plane. In addition, two sets of points are painted in each plot to distinguish individuals who are positive in Osteoporosis, Lipodystrophy or Sarcopenia, which may indicate which are the variables that could be candidates of being the most decisive in a classification problem.



After plotting the PC plots, and analyzing them carefully, one can make some conclusions:

- The features seem to be paired just as in the correlation plots.
- The classification of Sarcopenia seems to be the easiest to predict, since the ellipses are more separate than in the other two cases. And even though the two sets are not perfectly distinguished, it looks like there exists a group of variables that have a slight direct effect on the classification.
- Data reduction without using the PCs as variables is proven to be feasible and effective since there are a lot of variables that behave alike in the three plots.

Cleaned up data

From the previous graphs and plots it has been proven for the features to be grouped by their correlation and the PC representation. Thus, carefully analyzing the correlation and PC plots, the following groups are defined:

- L1T, L1BMD, L1L4BMD, L1L4T
- L2T, L2BMD, L2L4BMD, L2L4T
- L3T, L3BMD, L3Z
- L4T, L4BMD, L4Z, L1L4Z, L1Z, L2L4Z, L2Z
- L1L4BMD, L2L4BMD
- WardsT, WardsBMD
- NeckFT, NeckFBMD

- TotalFBMD, TotalFT, TrochBMD
- TotalFZ, TrochZ, TotalFT
- TLg, TotalLg
- FFMI, Apendicularleanmas
- RALg, LALg, BothALg
- RLLg, LLLg, BothLLg
- RAFg, LAFg
- RLFp, RLFg, LLFp, LLFg
- TFp, TFg
- FTrunkgFLegsg, FtrunkgFtotalg
- FlimbsgFtotalg, FlegsgFtotalg

So, by keeping one of each group, as the PC and correlation plots and graphs displayed, the dataset will not lose major information, and the probability of encountering an overfitted model will decrease.

```
rm.variables <- c("L1BMD", "L1L4BMD", "L1L4T", "L2BMD", "L2L4BMD", "L2L4T",
                                "L2L4Z", "L2Z","L3BMD", "L3Z", "L4BMD", "L4Z",
                                "L1L4Z", "L1L4Z", "WardsBMD",
                                "NeckFBMD", "TotalFT", "TrochBMD", "TrochZ",
                                "TotalLg", "Apendicularleanmas", "RALg", "LALg",
                                "RLLg", "LLLg", "LAFg", "RLFg", "LLFp", "LLFg",
                                "TFg", "FTrunkgFLegsg", "FlimbsgFtotalg")
data <- select(data, -one_of(rm.variables))
dim(data)
```

```
## [1] 1458   53
```

Once the variables to be removed are defined, the four sets that defined general, bone, lean and fat variables are now described as follows.

```
n.general <- n.general[!(n.general %in% rm.variables)]
n.bone <- n.bone[!(n.bone %in% rm.variables)]
n.fat <- n.fat[!(n.fat %in% rm.variables)]
n.lean <- n.lean[!(n.lean %in% rm.variables)]
```

Machine learning: Classification problems and methods

In this chapter the goal is to first describe the classification problems to resolve, and next to find which methods are preferred to resolve said classification problems.

The classification problem

As stated in previous chapters, the amount of categorical variables in the dataset is in fact small. The only categorical variables that can be fount in the treated non-raw data frame are: `Lipodistrophy`, `Sarcopenia`, `Osteoporosis`, `Osteopenia`, `gender_num`, and `Tscore_3cat`.

Even though it would definitely be interesting to try to classify `gender_num` and `Tscore_3cat`, the obvious and most challenging task would be to try to guess if an individual is ill or healthy for a specific-tissue-related illness taking into account the data that does not belong tho that tissue's set. For instance, a possible goal

is to predict if an individual is affected by Lipodystrophy by using the data related just to muscles and bones. Both Osteopenia and Osteoporosis are bone-related illnesses, and in this chapter, for simplicity, just Osteoporosis will be predicted, since of the two it is the one causing more struggles to HIV infected patients, but for further studies it would be definitely useful and interesting to see how the classification for Osteopenia behaves.

Thus three challenges arise; to predict Sarcopenia, Lipodystrophy, and Osteoporosis in HIV infected individuals. For the three of them, as analyzed in previous chapters, the data, even though it is fairly unbalanced (and that is an issue when working with machine learning problems) is decent enough to work with, and as seen when plotting correlation displays and the PC representations, there exist variables than behave differently in the three classification problems that can, if not determine, at least have an effect on these classifcation algorithms, meaning that just by looking at the previously displayed plots it looked like none of the three classification problems were nonsensical. One possible concern is the unbalanced distribution of the response variables, displayed in Table 3.

Response variable distributions		
Variables	Normal	Ill
`Lipodistrophy`	386	1072
`Sarcopenia`	1091	367
`Osteoporosis`	993	465

Table 3: Distribution of the response variables

Classification methods

> *"The process of choosing a machine learning algorithm involves matching the characteristics of the data to be learned to the biases of the available approaches. Since the choice of a machine learning algorithm is largely dependent upon the type of data you are analyzing and the proposed task at hand, it is often helpful to be thinking about this process while you are gathering, exploring, and cleaning your data. It may be tempting to learn a couple of machine learning techniques and apply them to everything, but resist this temptation. No machine learning approach is best for every circumstance. This fact is described by the No Free Lunch theorem, introduced by David Wolpert in 1996."*
>
> Lantz (2013)

Even though the process of selecting the best machine learning algorithm for a specific problem goes from non trivial to non feasible, one can at least test several adequate methods and see how they perform with the treated dataset. To decide which method outperforms the others one has to first keep in mind that performance is just a set of conditions that one sets as comparison rules to determine if one result is better than the other, and it may vary depending on said conditions selection, the the randomness involved in selecting the train and test sets, or how the validation techniques are made. A good way to start this section is to remember the second *No free meal* Theorem.

Theorem 1 (No free meal Theorem 2)**.** *[...] if one algorithm outperforms another for certain kinds of cost function dynamics, then the reverse must be true on the set of all other cost function dynamics.*

Wolpert, Macready, & others (1997)

And as a result of the No free meal theorem 2, Wolpert wrote a corollary: *" the Noo free meal theorems mean that if an algorithm does particularly well on average for one class of problems then it must do worse on average over the remaining problems. In particular, if an algorithm performs better than random search on some class of problems then in must perform worse than random search on the remaining problems. Thus comparisons reporting the performance of a particular algorithm with a particular parameter setting on a few sample problems are of limited utility. While such results do indicate behavior on the narrow range of problems considered, one should be very wary of trying to generalize those results to other problems."* Wolpert et al. (1997)

So keeping this in mind it is wise to acknowledge that the best way to handle the three classification problems is to list, for instance, three classification methods and validate their performance on the three problems. This idea will result in nine learners that will have to be compared by their performance in every tissue classification data, and for every tissue one of the three methoths will for sure outperform the remaining two.

The three selected supervised classification methods are: Random Forest, XGBoost and Neural Networks.

Before starting to build the learners, the train and test sets have to be determined. The train set will consist of a random sample of a 90% of the treated dataset rows, and consequently, the test set, which will be complementary to the train group, will be defined by the remaining 10% of the rows.

```
set.seed(250897)
tr.set <- sample(n, n*0.9)
```

The package `mlr` provides useful and easy to use functions that make procedures like testing, tuning and cross validating really simple. The idea for this section is to apply the three said methods for the three classification problems using k-fold cross validation to reduce the error of the resulting parameters and avoid overfitting.

\begin{quote} *The mlr package offers a clean, easy-to-use, and flexible domain-specific language for machine learning experiments in R. It supports classification, regression, clustering, and survival analysis with more than 160 modelling techniques. Defining learning tasks, training models, making predictions, and evaluating their performance abstracts from the implementation of the underlying learner through an object-oriented interface. Replacing one learning algorithm with another becomes as easy as changing a string. mlr goes far beyond simply providing a unified interface. It implements a generic architecture that allows the assessment of generalization performance, comparison of different algorithms in a scientifically rigorous way, feature selection, and hyperparameter tuning for any method [...] Finally, mlr provides sophisticated visualization methods that allow to show effects of partial dependence of models. mlr's long term goal is to provide a high-level domain-specific language to express as many aspects of machine learning experiments as possible."*
\end{package} Bischl et al. (2016)

For more information on the `mlr` package, and about learners, tasks, learning or tuning, check the proposed bibliography (Bischl et al. (2016)).

Using the function `makeLearner`, from the package `mlr`, several learners can be made for more than 80 machine learning methods including regression and classification. For the three methods selected, the learners will be created as follows:

```
lrn.rf <- makeLearner("classif.randomForest")
lrn.xgb <- makeLearner("classif.xgboost", par.vals = list(
  objective = "binary:logistic",
  eval_metric = "error",
  nrounds = 10,
  verbose=0
))
lrn.nn <- makeLearner("classif.nnet", par.vals = list(
  size = 1,
  trace = F
))
```

Random Forest

The first supervised classification model candidate is the Random Forest algorithm. This choice was made due to the versatility and good overall performance of this method, which has been on the top of classification contests in combination with other methods and by itself.

"Two well-known methods are boosting and bagging of classification trees. In boosting, successive trees give extra weight to points incorrectly predicted by earlier predictors. In the end, a weighted vote is taken for prediction. In bagging, successive trees do not depend on earlier trees - each is independently constructed using a bootstrap sample of the data set. In the end, a simple majority vote is taken for prediction. [...]

In addition to constructing each tree using a different bootstrap sample of the data, random forests change how the classification or regression trees are constructed. In standard trees, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node. This somewhat counterintuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis, support vector machines and neural networks, and is robust against overfitting. In addition, it is very user-friendly in the sense that it has only two parameters (the number of variables in the random subset at each node and the number of trees in the forest), and is usually not very sensitive to their values."

Liaw, Wiener, & others (2002)

Table 4 groups the strengths and weaknesses of the Random Forest method.

Strengths	Weaknesses
• It is a classifier that performs nicely with most problems • It provides nice results under noisy data and both categorical and continuous variables • It becomes easy to find the most important variables • It can handle big data frames with a huge amount of features and rows	• It is hard to impossible to decypher the model • For certain datasets it can be difficult to adapt the data to the model

Table 4: Strengths and weaknesses of the Random Forests method

Lantz (2013)

To compute the classification using the Random Forest method, the learner `classif.randomForest`, that is the `mlr` version of the `RandomForest` package, will be used. This method that calls the `randomForest` function from the homonymous package, admits just two input parameters. In this chapter no parameter tuning will be made, but still, since later on these parameters will be configured, it is convenient to list and describe them.

- `nodesize` Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). The default value for classification problems is 5.
- `ntree` Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times. The default value is 500.

```
# 10-fold repeated stratified Cross Validation with Random Forest
# for the three body tissues

pred.fat.rf <- repcv(lrn.rf, traintask.fat,
                     folds=10, reps=params$RepsCV, stratify = T, measures = measList)
pred.lean.rf <- repcv(lrn.rf, traintask.lean,
                      folds=10, reps=params$RepsCV, stratify = T, measures = measList)
pred.bone.rf <- repcv(lrn.rf, traintask.bone,
                      folds=10, reps=params$RepsCV, stratify = T, measures = measList)
```

XGBoost

The second method candidate is a boosting algorithm called Extreme Gradient Boosting (or XGBoost)(Chen & Guestrin, 2016). This gradient boosting algorithm is popular due to its great performance in machine learning contests and its decent results in all kinds of classification and regression problems.

> *"Similar to bagging, boosting uses ensembles of models trained on resampled data and a vote to determine the final prediction. The key difference is that the resampled datasets in boosting are constructed specifically to generate complementary learners, and the vote is weighted based on each model's performance rather than giving each an equal vote. [...] The algorithm is based on the idea of generating weak leaners that iteratively learn a larger portion of the difficult-to-classify examples in the training data by paying more attention (that is, giving more weight) to often misclassified examples. Beginning from an unweighted dataset, the first classifier attempts to model the outcome. Examples that the classifier predicted correctly will be less likely to appear in the training dataset for the following classifier, and conversely, the difficult-to-classify examples will appear more frequently. As additional rounds of weak learners are added, they are trained on data with successively more difficult examples. The process continues until the desired overall error rate is reached or performance no longer improves. At that point, each classifier's vote is weighted according to its accuracy on the training data on which it was built."*
>
> Lantz (2013)

> *"XGBoost is a scalable machine learning system for tree boosting. The system is available as an open source package. The impact of the system has been widely recognized in a number of machine learning and data mining challenges [...] Among the 29 challenge winning solutions published at Kaggle's blog during 2015, 17 solutions used XGBoost. Among these solutions, eight solely used XGBoost to train the model, while most others combined XGBoost with neural nets in ensembles. For comparison, the second most popular method, deep neural nets, was used in 11 solutions. The success of the system was also witnessed in KDDCup 2015, where XGBoost was used by every winning team in the top10. Moreover, the winning teams reported that ensemble methods outperform a well configured XGBoost by only a small amount"*
>
> Chen & Guestrin (2016)

In Table 5 one can find the overall strengths and weaknesses of the XGBoost method.

Strengths	Weaknesses
• Its performance tends to be decent in all kinds of problems, including regression and classification. • One of the first choices in data competitions. • Usually outperforms other methods when facing difficult data. • It is a good choice if one does not know a lot about the data.	• It is much younger than its fellow competitors, which means that much less investigation has been done with or around it. • It may turn into an overfitted model unexpectedly.

Table 5: Strengths and weaknesses of the XGBoost method

Lantz (2013)

To compute the classification using XGBoost, the learner `classif.xgboost`, that is the `mlr` version of the `xgboost` package, will be used. This method calls the `xgboost` function from the homonymous package, and has a quite big amount of possible input parameters. In the tuning chapter the following parameters will be adjusted:

- `nrounds` Maximum number of boosting iterations.

- `max_depth` Maximum depth of a tree. The default value is 6.
- `lambda` L^2 regularization term on weights. The default value is 0.
- `eta` "Control the learning rate". Scale the contribution of each tree by a factor of $0 < eta < 1$ when it is added to the current approximation. Used to prevent overfitting by making the boosting process more conservative. Lower value for eta implies larger value for nrounds: low eta value means model more robust to overfitting but slower to compute. The default value is 0.3.
- `subsample` subsample ratio of the training instance. Setting it to 0.5 means that xgboost randomly collected half of the data instances to grow trees and this will prevent overfitting. It makes computation shorter (because less data to analyse). It is advised to use this parameter with eta and increase nrounds. The default value is 1.
- `colsample_bytree` Subsample ratio of columns when constructing each tree. The default value is 1.

```
# 10-fold repeated stratified Cross Validation with XGBoost for the three body tissues

pred.fat.xgb <- repcv(lrn.xgb, traintask.fat,
                      folds=10, reps=params$RepsCV, stratify = T, measures = measList)
pred.lean.xgb <- repcv(lrn.xgb, traintask.lean,
                      folds=10, reps=params$RepsCV, stratify = T, measures = measList)
pred.bone.xgb <- repcv(lrn.xgb, traintask.bone,
                      folds=10, reps=params$RepsCV, stratify = T, measures = measList)
```

Neural networks

An Artificial Neural Network (ANN) models the relationship between a set of input signals and an output signal using a model derived from our understanding of how a biological brain responds to stimuli from sensory inputs. Just as a brain uses a network of interconnected cells called neurons to create a massive parallel processor, the ANN uses a network of artificial neurons or nodes to solve learning problems.

Strengths	Weaknesses
• Can be adapted to classification or numeric prediction problems • Among the most accurate modeling approaches • Makes few assumptions about the data's underlying relationships	• Reputation of being computationally intensive and slow to train, particularly if the network topology is complex • Easy to overfit or underfit training data • Results in a complex black box model that is difficult if not impossible to interpret

Table 6: Strengths and weaknesses of the Neural Networks method

Lantz (2013)

To compute the classification using Neural Networks, the learner `classif.nnet`, that is the `mlr` version of the `nnet` package, will be used. This method calls the `nnet` function from the homonymous package, and has a quite big amount of possible input parameters. In the tuning chapter the following parameters will be adjusted:

- `size` Number of units in the hidden layer. Can be zero if there are skip-layer units.
- `skip` switch to add skip-layer connections from input to output. The default value is false.
- `maxit` Maximum number of iterations. The default value is 100.

- `abstol` Stop if the fit criterion falls below abstol, indicating an essentially perfect fit. The default value is 10^{-4}.

```
# 10-fold repeated stratified Cross Validation with Neural Networks
# for the three body tissues

pred.fat.nn <- repcv(lrn.nn, traintask.fat, folds=10,
                     reps=params$RepsCV, stratify = T, measures = measList)
pred.lean.nn <- repcv(lrn.nn, traintask.lean, folds=10,
                      reps=params$RepsCV, stratify = T, measures = measList)
pred.bone.nn <- repcv(lrn.nn, traintask.bone, folds=10,
                      reps=params$RepsCV, stratify = T, measures = measList)
```

Untuned results

Once the Random Forest, XGBoost, and Neural Networks methods have been trained for the three tissue related datasets, one has to compare their performance. Parameters like the accuracy never fail to be a good estimator of the performance, even though some other terms have to be checked, for example the specificity, the FNR and the Kappa coefficient. Since this specific problem is related to a clinic case, the number of false negatives should be as little as possible, thus between two methods with close performance indicators but different false negative numbers, one should use the one with the lowest false negative rate.

Fat Results			
	Random forest	XGBoost	Neural networks
Accuracy	0.749	0.736	0.721
Specificity	0.914	0.881	0.892
FNR	0.693	0.653	0.737
Kappa	0.257	0.253	0.148

Lean Results			
	Random forest	XGBoost	Neural networks
Accuracy	0.831	0.842	0.887
Specificity	0.521	0.623	0.854
FNR	0.063	0.084	0.102
Kappa	0.506	0.562	0.721

Bone Results			
	Random forest	XGBoost	Neural networks
Accuracy	0.696	0.694	0.755
Specificity	0.359	0.377	0.595
FNR	0.14	0.152	0.167
Kappa	0.24	0.244	0.435

Table 7: Results of the repeated cross validation for the untuned models

Comparing methods

The results obtained from the untuned training, displayed in Table 7, can be analyzed by comparing the four performance parameters. The first overall conclusion is that for every disease, the three methods behave alike, even though there is always one that performs slightly better. The Neural Network results for the lean and bone sets outperform the other two methods clearly, but for the fat set, even though the accuracies for the Random Forest and Neural Networks are close, by looking at the Kappa coefficient, and the specificity, one can take the Random Forest as the untrained method with the best performance for this set.

When introducing the methods, they were applied with the default parameters, which is a vague option to compute a classifier. Thus in this chapter the main goal is to find how the method parameters can be modified, and calculate the values for these parameters to obtain the model with the best performance.

Parameter tuning

The parameter tuning has been computed using the `tuneModel` function from the `mlr` package. This function makes parameter tuning really easy and simple, allowing the user to decide which parameters to tune, their range or fixed values, the priority of the measures to optimize (for example in this project accuracy, kappa and fnr are the ones that have been optimized in this same priority order), and the validation technique to apply.

The parameters to tune for each model have been previously stated even though no ranges or values have been set. In this section these sets or intervals will be determined in order to proceed to the tuning execution.

The Random Forest implementation from the `randomForest` package allows the user to modify the parameters `nodesize` and `nodetree`. These parameters must be positive integers and can be as big as the user wants. For the dataset that will be used in this computation, and taking into account the correlation plots and PC results displayed in previous sections, the values displayed in the following code block have been assigned to the parameters in the tuning process.

```
makeDiscreteParam("nodesize",values = c(1, 3, 5, 7, 15), default=5),
makeDiscreteParam("ntree", values = c(200, 500, 800, 1200), default=500)
```

The XGBoost method from the `xgboost` package is highly customizable. To prevent computational time to be massive, a decent set has been selected to be adjusted in the tuning process as displayed in the following code chunk.

```
makeDiscreteParam("nrounds", values=c(2, 5, 10, 50, 100), default=2),
makeDiscreteParam("max_depth", values=c(3, 6, 10, 20), default=6),
makeDiscreteParam("lambda",values=c(0, 0.2, 0.6), default=0),
makeDiscreteParam("eta", values = c(0.01, 0.1, 0.3, 0.4), default=0.3),
makeDiscreteParam("subsample", values = c(0.3, 0.5, 1), default=1),
makeDiscreteParam("colsample_bytree", values = c(0.3, 0.5, 0.7, 1), default=1)
```

To conclude, the Neural Network customizable parameters from the package `nnet` that have been chosen to be modified are the following:

```
makeIntegerParam("size",lower=1,upper=5, default = 1),
makeLogicalParam("skip", default=FALSE),
makeDiscreteParam("maxit",values=c(80, 100, 200), default=100),
makeDiscreteParam("abstol", values = c(1e-3, 1e-4, 1e-5), default=1e-4)
```

The complete original algorithm involved in the tuning task can be found in Appendix 2. This said algorithm combines parameter tuning for the nine classification problems, resulting from combing the three machine learning methods with the three body composition sets, with a 10-fold stratified repeated cross-validation. The estimated computation time to obtain the optimal parameters goes from 4 to 32 hours, depending on the number of repetitions for every cross-validation iteration. Since the computation can be quite time-consuming, the tuning results are locally saved with `.RData` extentions, so they do not have to be computed for every code compilation.

Fat Results			
	Random forest	XGBoost	Neural networks
Accuracy	0.752	0.754	0.755
Specificity	0.925	0.941	0.895
FNR	0.714	0.748	0.624
Kappa	0.25	0.236	0.303

Lean Results			
	Random forest	XGBoost	Neural networks
Accuracy	0.824	0.842	0.897
Specificity	0.504	0.635	0.802
FNR	0.066	0.087	0.071
Kappa	0.486	0.567	0.729

Bone Results			
	Random forest	XGBoost	Neural networks
Accuracy	0.716	0.721	0.76
Specificity	0.337	0.327	0.572
FNR	0.106	0.095	0.151
Kappa	0.261	0.264	0.428

Table 8: Results of the repeated cross validation for the tuned models

So, by analyzing the performance results for the tuned methods displayed at Table 8, one can compare the three supervised tuned models for the three sets.

Taking into account the results obtained with the lean set, the Neural Networks approach clearly outperforms the other two methods, all its performance parameters are more optimal than the ones of the other two models, and it is safe to say that its accuracy and Kappa values are really positive.

Comparing the perfomance results over the bone set, the Neural Network approach is again the most optimal, delivering an accuracy and Kappa values that clearly diverge from the other two methods.

Finally, the fat set is the one with tightest results, and the decision of choosing the most adequate algorithm is far from trivial. If one compares the accuracies, the XGBoost model is the one delivering the best value, even though it differs minimally from its competitors. The Kappa value, in this case is less trivial to examine, and the Neural Networks algorithm delivers the best value. Since the FNR for the Neural Networks results is also clearly more optimal, one would choose this algorithm over the other two.

On this phase of the project, several models have been tried, and what has happened some times, for weak unstratified tuning approaches, is that the presence of unbalanced data has led to overfitted and poorly trained models. These overfitted models had both the Kappa value and the FNR equal to 0, and an accuracy equal to the No Information rate, meaning that the algorithm tends to return always either negative or positive outputs. To clarify, if a dataset contains a huge amout of healthy individuals (let's say a 99%), and just a residual part of infected observations (the remaining 1%) the algoritm may assume that the proportion of outputs will probably be as well 1% - 99%, and thus, if the prediction always classifies individuals as healthy, regardless of their variables, the accuracy will always be around 99%, which is good in most cases but not in this one.

To not stratify data and priorize indicators like TPR in the tuning process can produce highly accurate nonesense models that when applied to a stratified population deliver poor results.

In order to conclude, the test prediction statistics will be displayed, as well as the tuned model optimal parameters.

Table 9: Parameters for the Neural Networks method for the Fat variable

size	skip	maxit	abstol
1	1	200	1e-04

Table 10: Parameters for the Neural Networks method for the Lean variable

size	skip	maxit	abstol
3	0	100	1e-04

Table 11: Parameters for the Neural Networks method for the Bone variable

size	skip	maxit	abstol
1	0	80	0.001

Testing with the tuned models

Training the tuned models with the train test and applying predictions to the set, is a good way to obtain a result of the performance of the final methods over data that has never been in touch with them. Using the function `train` the tuned methods are trained, and the prediction can be computed calling the function `predict`.

```
fat.model <- setHyperPars(lrn.nn, par.vals=Fat.RNN$x)
lean.model <- setHyperPars(lrn.nn, par.vals=Lean.RNN$x)
bone.model <- setHyperPars(lrn.nn, par.vals=Bone.RNN$x)

fat.model.trained <- train(fat.model, traintask.fat)
fat.pred <- predict(fat.model.trained, testtask.fat)

lean.model.trained <- train(lean.model, traintask.lean)
lean.pred <- predict(lean.model.trained, testtask.lean)

bone.model.trained <- train(bone.model, traintask.bone)
bone.pred <- predict(bone.model.trained, testtask.bone)
```

Table 12: Test results of the Neural Networks model for the fat set

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.753	0.184	0.675	0.821	0.795	0.906	0.405

Table 13: Test results of the Neural Networks model for the lean set

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.877	0.632	0.812	0.925	0.767	0.001	0.239

Table 14: Test results of the Neural Networks model for the bone set

Accuracy	Kappa	AccuracyLower	AccuracyUpper	AccuracyNull	AccuracyPValue	McnemarPValue
0.726	0.276	0.646	0.797	0.753	0.807	0.874

The parameter that determines if a model or a prediction is significant is the accuracy p-value. This p-value

is calculated under the assumption that the prediction is independent of the class distribution in the train subset; This assumption, known as the no information rate is a good indicator of significance in machine learning models and predictions. In this case, the only p-value < 0.05 is the one found at Table 13, which validates the assumption that the neural network tuned model is significant, thus its mean accuracy, which is 0.877 and differs from the no information rate (0.767), can be taken as a good indicator for the accuracy of future predictions. On the other hand the McNemmar p-value indicates if the distribution of false negatives and false positives is alike. Since in the three cases its value is greater than 0.05, one can assume that the optimal classification models tend to return unequal values for FPR and FNR, which can be a consequence of the fact that the subsets test and train are not stratified.

Even though the predictive methods for the fat and bone diseases are not statistically significative, the fact that the lean disease neural network classifier is statistically valid is enough to set the goal of this project as achieved. This proves the initial assumption that the lean ageing disease related to sarcopenia could be successfully predicted using just the features belonging to the bone and fat subsets as explanatory variables.

Variable importance

The Neural Networks method has been selected as the most optimal for the three body composition sets. For the Random Forest it is easy to compute the variable importance, and there exist a lot of packages that perform variable importance operations. In this case the method to obtain information about the importance that the features have, is more experimental but it has proven to deliver accurate results in literature. (Zhang et al., 2018)

> *"Garson (1991) proposed a method, later modified by Goh (1995), for partitioning the neural network connection weights in order to determine the relative importance of each input variable in the network. [...] It is important to note that Garson's algorithm uses the absolute values of the connection weights when calculating variable contributions, and therefore does not provide the direction of the relationship between the input and output variables."*
>
> Olden & Jackson (2002)

> *"Here, we provide a more appropriate comparison of the different methodologies by using Monte Carlo simulations with data exhibiting defined (and consequently known) numeric relationships. Our results show that a Connection Weight Approach that uses raw input-hidden and hidden-output connection weights in the neural network provides the best methodology for accurately quantifying variable importance and should be favored over the other approaches commonly used in the [...] literature."*
>
> *"We found that the Connection Weight Approach provides the best overall methodology for accurately quantifying variable importance and should be favored over the other approaches examined in this study. This approach successfully identified the true importance of all the variables in the neural network, including variables that exhibit both strong and weak correlations with the response variable. Moreover, a randomization test for the Connection Weight Approach has been recently developed [...], which provides a tool for pruning null-connection weights and neurons from the network and determining the statistical significance of variable contributions.""*
>
> Olden, Joy, & Death (2004)

To perform the Garson and Olden algorithms, the `NeuralNetTools` package will be used. This package has built-in functions to perform Neural Networks analyze their topology or performance. Since the only computed method that provided significative statistic values is the Neural Networks lean classification model, for simplicity the Garson and Olden algorithms will only be applied to it, but before computing the feature importance, a graphical representation of the net will be displayed using the function `plotnet` (Beck, 2018), including painted vertices according to the weight distribution. The remaining Garson, Olden and network topology plots can be found in Appendix 3.

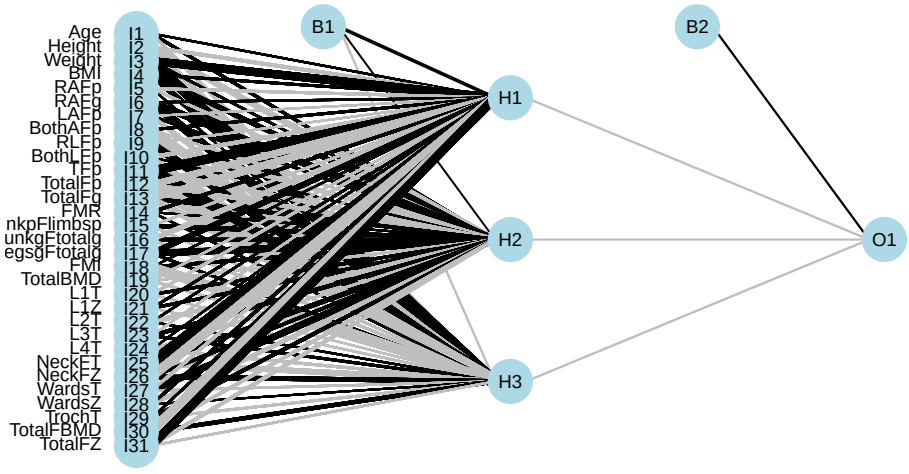


Figure 11: Graphical representation of the network topology from the lean disease predictor

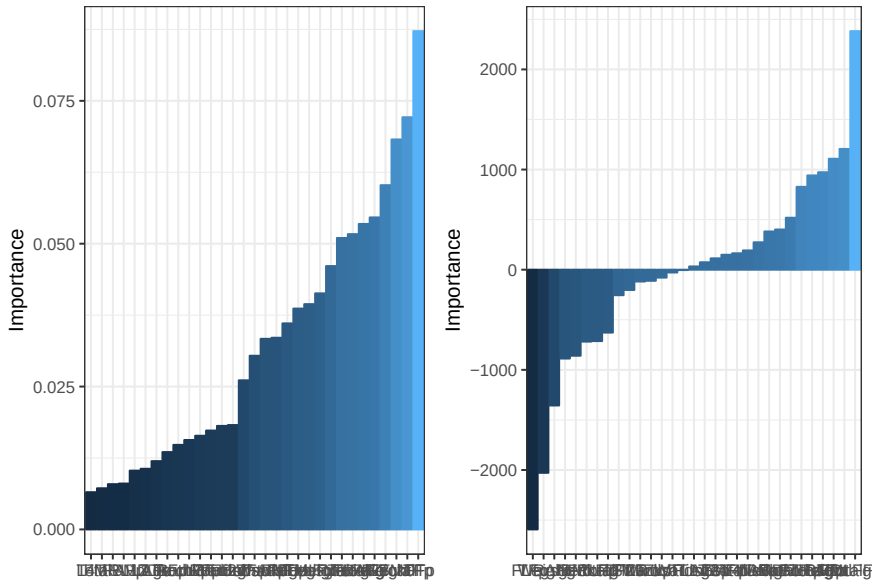


Figure 12: Garson and Olden plots for the lean disease neural network predictor

Now that the net is displayed in Figure 11, even though it is not very easy on the eye, the Garson and Olden plots can be displayed returning feature importance information.

Table 15: Feature importance provided by the Garson and Olden algorithms

garson	olden
TFp	TFp
TotalFp	TotalFp
NeckFT	Weight
Weight	RAFg
TotalFBMD	TotalFg
RAFg	FMI

Both in Table 15 and in Figure 12 one can see how relevant the dataset features are, and it is easy to see that a set of approximately half a dozen features are clearly more important than the rest. These features are displayed in Table 15, and since as stated in (Olden et al., 2004) the Olden approach is more reliable, one should see which features are placed on top of the Olden list, and which features can be found in both lists. This way, the "black box" core of the neural network can be read, and once analyzed, a proper biological interpretation of the importance of the variables may be made providing in the best of cases valuable information to the clinical team.

Conclusions

Having the chance to work with real data, and in particular data from the city you live in, is always remarkable and challenging.

In this project I had the chance to develop a complete data analysis machine learning study starting from a dataset with samples from an actual clinic project that FLSIDA has collected since the early 2000. The goals of the project have been achieved, since a tuned classification method has been developed and proven to be actually significative. Even that the goals had been achieved, to conclude, additional information about the model's insights is provided, which might be useful to the clinical team of FLSIDA.

Since it was my first time developing from start to finish a machine learning project, I was initially willing to start playing with the predictive methods as soon as possible. My advisors wisely suggested me to start ananlyzing the dataset carefully, and to understand every feature, and once that was reached, to check if the dataset was consistent or not. Once this phase was reached, the data analysis section seemed more intriguing, and new challenges such as dimension reduction arosed. This part of the project which initially seemed less challenging ended being the most essential part of the work, and without it the quality that the final product has could not have been achieved.

On the other hand, I would have liked to develop this project with more time, since more data consistency analysis would have been handy. For instance the outliers and feature consistency was not delivered to the clinical team due to time constraints, and this information would have definitely improved the quality of the dataset.

For further studies I would suggest more communication with the clinical team, and more time spent on the analysis of the quality of the data. I am proud of how the prediction methods were controlled, even though the stratified repeated 10-fold cross-validation was really time consuming, and with a regular laptop, the computation time with 10 repetitions per round was over 30 hours, which is incompatible with agile programming.

The original planned schedule was inconsistent with the project study line. This was a consequence of my unfamiliarity with machine learning project management, even though the first meeting with the project advisors fixed this issue on time and the reschedule was both feasible and positive.

To conclude, I am grateful for having had the chance to participate in such a challenging project; I am greatful for the support and advice that my advisors have given me, it has made this journey really smooth and interesting, and I feel like I have definitely learned a lot, both autonomously and with their support. Beyond the fact that the goals have been succesfully reached, I am proud of this dissertation as it has taught me uncountable interesting concepts and subjects related to HIV, AIDS, health science, clinical procedures, and of course data analysis and machine learning.

Glossary

- **FLSIDA:** Fundació lluita contra la sida (Fight AIDS foundation)
- **AIDS:** Acquired immunodeficency syndrome
- **HIV:** Human immunodeficency virus
- **DEXAS:** Dual-Energy X-Ray Absorptiometry
- **cART:** Combination antiretroviral therapy: combinations of drugs that are used to keep HIV infections under control.
- **BMD:** Bone mineral density
- **BMC:** Bone mineral content
- **BA:** Bone area
- **FMI:** Fat mass index
- **FMR:** Fat mass ratio
- **PC:** Principal Component
- **PCA:** Principal Component Analysis
- **XGBoost:** Extreme gradient boosting
- **FNR:** False negative rate
- **FPR:** False positive rate
- **TPR:** True positive rate

References

Ayyappan, S., Niveditha, B., & Breur, G. J. (2017). Determination of baseline bone mineral density using dual energy x-ray absorptiometry in suffolk-dorset hybrid ewes. *International Journal of Veterinary Science and Medicine*, *5*(1), 41–46.

Beck, M. W. (2018). NeuralNetTools: Visualization and analysis tools for neural networks. *Journal of Statistical Software*, *85*(11), 1.

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., ... Jones, Z. M. (2016). Mlr: Machine learning in r. *The Journal of Machine Learning Research*, *17*(1), 5938–5942.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794. ACM.

Cosman, F., Beur, S. J. de, LeBoff, M., Lewiecki, E., Tanner, B., Randall, S., & Lindsay, R. (2014). Clinician's guide to prevention and treatment of osteoporosis. *Osteoporosis International*, *25*(10), 2359–2381.

Echeverría, P., Bonjoch, A., Puig, J., Estany, C., Ornelas, A., Clotet, B., & Negredo, E. (2018). High prevalence of sarcopenia in hiv-infected individuals. *BioMed Research International*, *2018*.

Elkoushy, M. A., Jundi, M., Lee, T. T., & Andonian, S. (2014). Bone mineral density status in urolithiasis patients with vitamin d inadequacy followed at a tertiary stone centre. *Canadian Urological Association Journal*, *8*(9-10), 323.

Ellis, K. J., Shypailo, R. J., Hardin, D. S., Perez, M. D., Motil, K. J., Wong, W. W., & Abrams, S. A. (2001). Z score prediction model for assessment of bone mineral content in pediatric diseases. *Journal of Bone and Mineral Research*, *16*(9), 1658–1664.

Fodor, I. K. (2002). *A survey of dimension reduction techniques*. Lawrence Livermore National Lab., CA (US).

Freitas, P., Santos, A. C., Carvalho, D., Pereira, J., Marques, R., Martinez, E., ... Medina, J. L. (2010). Fat mass ratio: An objective tool to define lipodystrophy in hiv-infected patients under antiretroviral therapy. *Journal of Clinical Densitometry*, *13*(2), 197–203.

Grenha, I., Oliveira, J., Lau, E., Santos, A. C., Sarmento, A., Pereira, J., ... Freitas, P. (2018). HIV-infected patients with and without lipodystrophy under combined antiretroviral therapy: Evaluation of body composition. *Journal of Clinical Densitometry*, *21*(1), 75–82.

Lantz, B. (2013). *Machine learning with r*. Packt Publishing Ltd.

Liaw, A., Wiener, M., & others. (2002). Classification and regression by randomForest. *R News*, *2*(3), 18–22.

Loenneke, J. P., Wilson, J. M., Wray, M. E., Barnes, J. T., Kearney, M. L., & Pujol, T. J. (2012). The estimation of the fat free mass index in athletes. *Asian Journal of Sports Medicine*, *3*(3), 200.

Meir-Shafrir, K., & Pollack, S. (2012). Accelerated aging in hiv patients. *Rambam Maimonides Medical Journal*, *3*(4).

Olden, J. D., & Jackson, D. A. (2002). Illuminating the "black box": A randomization approach for understanding variable contributions in artificial neural networks. *Ecological Modelling*, *154*(1-2), 135–150.

Olden, J. D., Joy, M. K., & Death, R. G. (2004). An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data. *Ecological Modelling*, *178*(3-4), 389–397.

Pampaka, M., Hutcheson, G., & Williams, J. (2016). Handling missing data: Analysis of a challenging data set using multiple imputation. *International Journal of Research & Method in Education*, *39*(1), 19–37.

Peltz, G., Aguirre, M. T., Sanderson, M., & Fadden, M. K. (2010). The role of fat mass index in determining obesity. *American Journal of Human Biology*, *22*(5), 639–647.

Savage, D. B., Petersen, K. F., & Shulman, G. I. (2007). Disordered lipid metabolism and the pathogenesis

of insulin resistance. *Physiological Reviews*, *87*(2), 507–520.

Sözen, T., Özişik, L., & Başaran, N. Ç. (2017). An overview and management of osteoporosis. *European Journal of Rheumatology*, *4*(1), 46.

Tang, Y., Horikoshi, M., & Li, W. (2016). Ggfortify: Unified interface to visualize statistical results of popular r packages. *The R Journal*, *8*(2), 478–489.

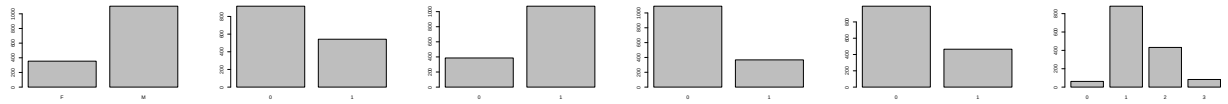
Wolpert, D. H., Macready, W. G., & others. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82.

Zhang, Z., Beck, M. W., Winkler, D. A., Huang, B., Sibanda, W., Goyal, H., & others. (2018). Opening the black box of neural networks: Methods for interpreting neural network models in clinical applications. *Annals of Translational Medicine*, *6*(11).

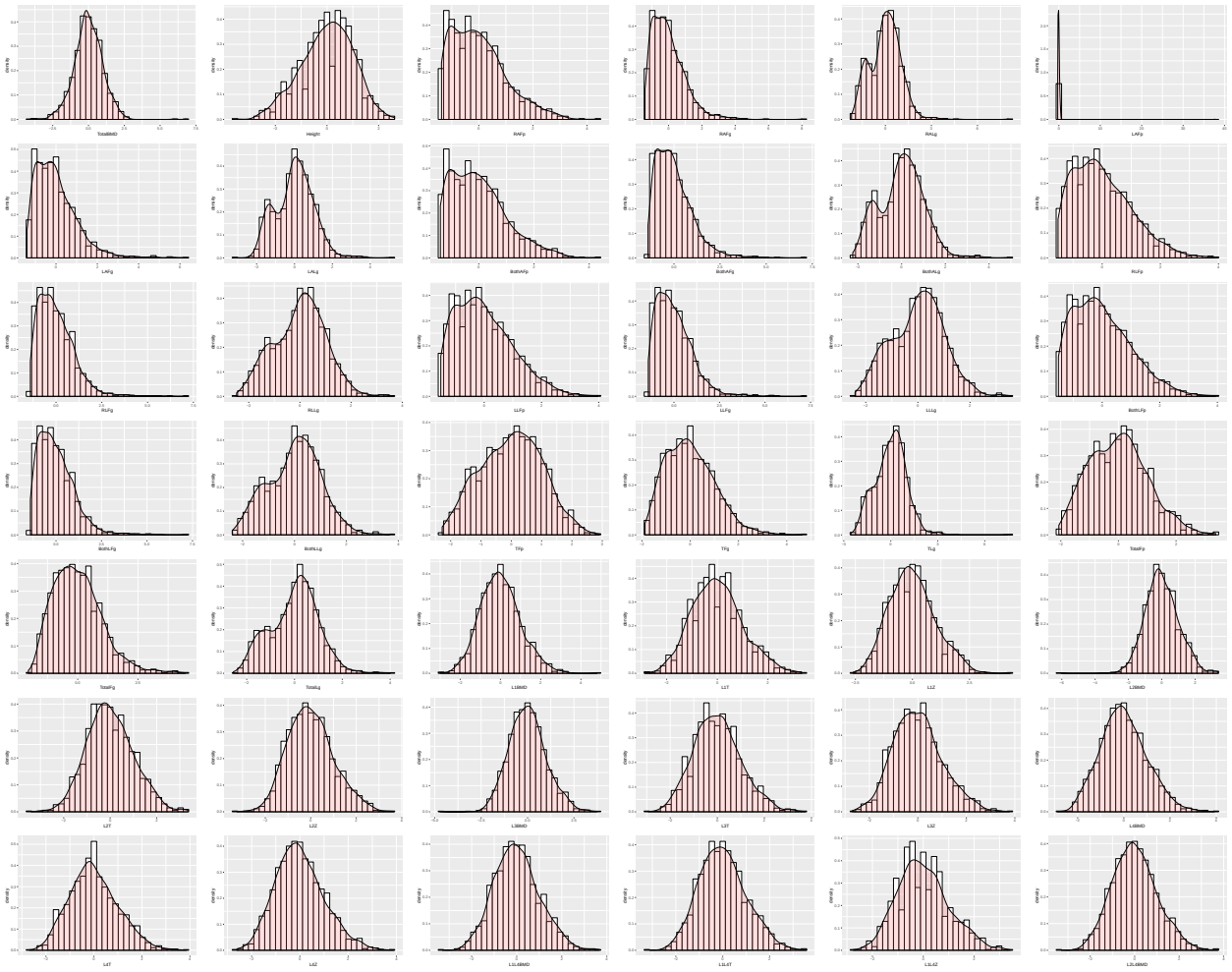
Appendices

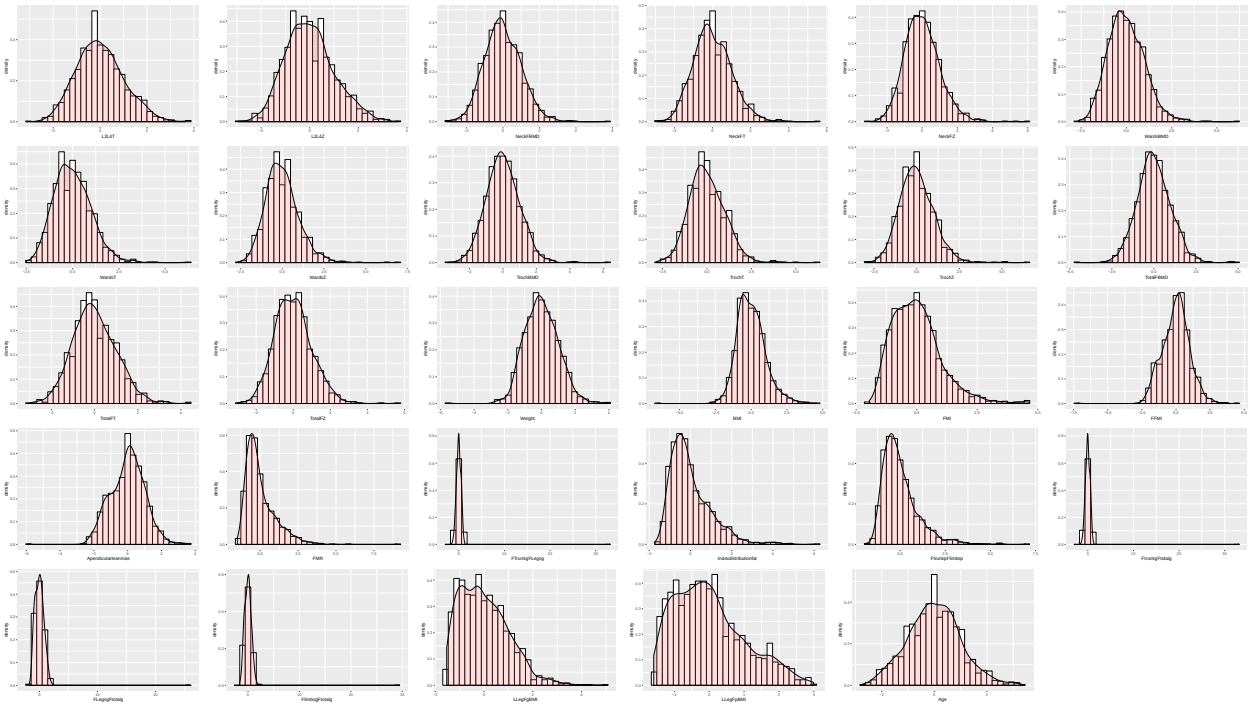
Appendix 1: Distribution of the continuous and discrete variables

```
for (vname in disc.vars) {
  barplot(table(raw_data[,vname]))
}
```



```
for (vname in cont.vars) {
  p1 <- ggplot(raw_data, aes(x=eval(parse(text =vname)))) +
    geom_histogram(aes(y=..density..), colour="black", fill="white") +
    geom_density(alpha=.2, fill="#FF6666") + labs(x = vname)
  grid.arrange(p1, ncol=1)
}
```





Appendix 2: Parameter tuning

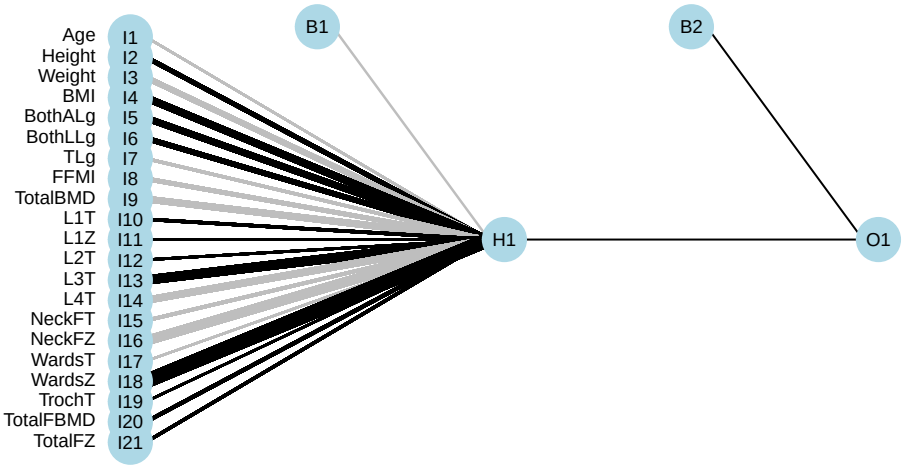
```
set.seed(230994)
cv.tr.set <- sample(n, n*0.9)
measList <- list(acc, kappa, fnr, tpr, fpr, tnr )

cv.data <- data
cv.data$Lipodistrophy <- as.factor(cv.data$Lipodistrophy)
cv.data$Sarcopenia <- as.factor(cv.data$Sarcopenia)
cv.data$Osteoporosis <- as.factor(cv.data$Osteoporosis)

getRRF = function(Rsets, expl.vars, pred.var) {
  n.expl <- c(n.general, expl.vars, pred.var)

  cv.train <- cv.data[Rsets, n.expl]

  traintask <- makeClassifTask(data = cv.train,target = pred.var)

  #set parameter space
  params.rf <- makeParamSet(
    makeDiscreteParam("nodesize",values = c(1, 3, 5, 7, 15), default=5),
    makeDiscreteParam("ntree", values = c(200, 500, 800, 1200), default=500)
  )

  #set validation strategy
  rf.rdesc <- makeResampleDesc("RepCV",folds=10, reps=params$RepsCV, stratify = T)

  #set optimization technique
  rf.ctrl <- makeTuneControlRandom()

  var.name <-  paste('rrf',pred.var, sep="")
  if (paste(var.name,'.RData', sep="") %in% list.files()) {
    load(paste(getwd(), paste('/rrf',pred.var, sep=""),'.RData', sep=""))
    r.rf <- eval(parse(text = var.name))
  } else {
    r.rf <- tuneParams(learner = lrn.rf
     ,task = traintask
     ,resampling = rf.rdesc
     ,measures = measList
     ,par.set = params.rf
     ,control = rf.ctrl
     ,show.info = F)
    assign(var.name, r.rf)
    save(list=c(var.name), file=paste(var.name,'.RData', sep=""))
  }
  return(r.rf)
}
getRXGB = function(Rsets, expl.vars, pred.var) {
  n.expl <- c(n.general, expl.vars, pred.var)

  cv.train <- cv.data[Rsets, n.expl]

  traintask <- makeClassifTask(data = cv.train,target = pred.var)
```

```
  #set parameter space
  params.xgb <- makeParamSet(
    makeDiscreteParam("nrounds", values=c(2, 5, 10, 50, 100), default=2),
    makeDiscreteParam("max_depth", values=c(3, 6, 10, 20), default=6),
    makeDiscreteParam("lambda",values=c(0, 0.2, 0.6), default=0),
    makeDiscreteParam("eta", values = c(0.01, 0.1, 0.3, 0.4), default=0.3),
    makeDiscreteParam("subsample", values = c(0.3, 0.5, 1), default=1),
    makeDiscreteParam("colsample_bytree", values = c(0.3, 0.5, 0.7, 1), default=1)
  )

  # define search function
  xgb.ctrl <- makeTuneControlRandom()

  # k fold cross validation
  xgb.rdesc <- makeResampleDesc("RepCV",folds = 10, reps=params$RepsCV, stratify = T)

  var.name <-  paste('rxgb',pred.var, sep="")
  if (paste(var.name,'.RData', sep="") %in% list.files()) {
    load(paste(getwd(), paste('/rxgb',pred.var, sep=""),'.RData', sep=""))
    r.xgb <- eval(parse(text = var.name))
  } else {
     r.xgb <- tuneParams(learner = lrn.xgb
                         ,task = traintask
                         ,resampling = xgb.rdesc
                         ,measures = measList
                         ,par.set = params.xgb
                         ,control = xgb.ctrl
                         ,show.info = F)
    assign(var.name, r.xgb)
    save(list=c(var.name), file=paste(var.name,'.RData', sep=""))
  }
  return(r.xgb)
}
getRNN = function(Rsets, expl.vars, pred.var) {
  n.expl <- c(n.general, expl.vars, pred.var)

  cv.train <- cv.data[Rsets, n.expl]
  traintask <- makeClassifTask(data = cv.train,target = pred.var)

  #set parameter space
  params.nn <- makeParamSet(
    makeIntegerParam("size",lower=1,upper=5, default = 1),
    makeLogicalParam("skip", default=FALSE),
    makeDiscreteParam("maxit",values=c(80, 100, 200), default=100),
    makeDiscreteParam("abstol", values = c(1e-3, 1e-4, 1e-5), default=1e-4)
  )

  #define search function
  nn.ctrl <- makeTuneControlRandom()

  #10 fold cross validation
  nn.rdesc <- makeResampleDesc("RepCV", folds = 10, reps=params$RepsCV, stratify = T)
```

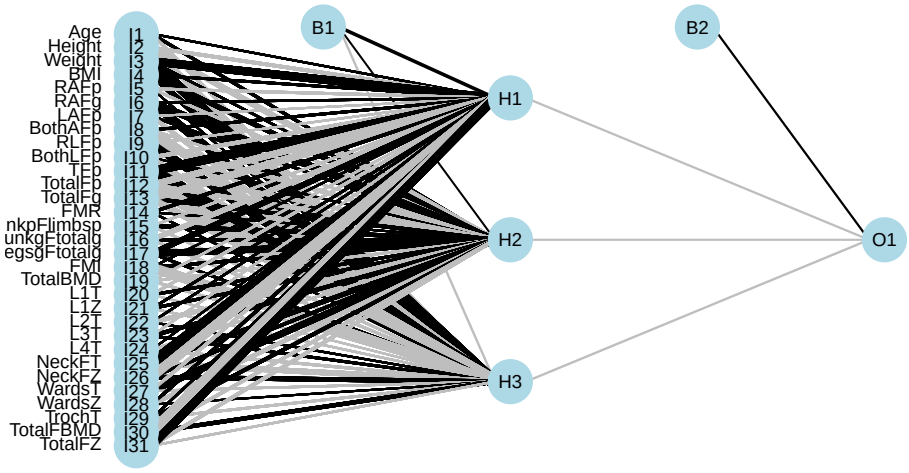
```
  var.name <-  paste('rnn',pred.var, sep="")
  if (paste(var.name,'.RData', sep="") %in% list.files()) {
    load(paste(getwd(), paste('/rnn',pred.var, sep=""),'.RData', sep=""))
    r.nn <- eval(parse(text = var.name))
  } else {
     r.nn <- tuneParams(learner = lrn.nn
                        ,task = traintask
                        ,resampling = nn.rdesc
                        ,measures = measList
                        ,par.set = params.nn
                        ,control = nn.ctrl
                        ,show.info = F)
   assign(var.name, r.nn)
    save(list=c(var.name), file=paste(var.name,'.RData', sep=""))
  }
  return(r.nn)
}
```

Appendix 3: Feature importance and net topology

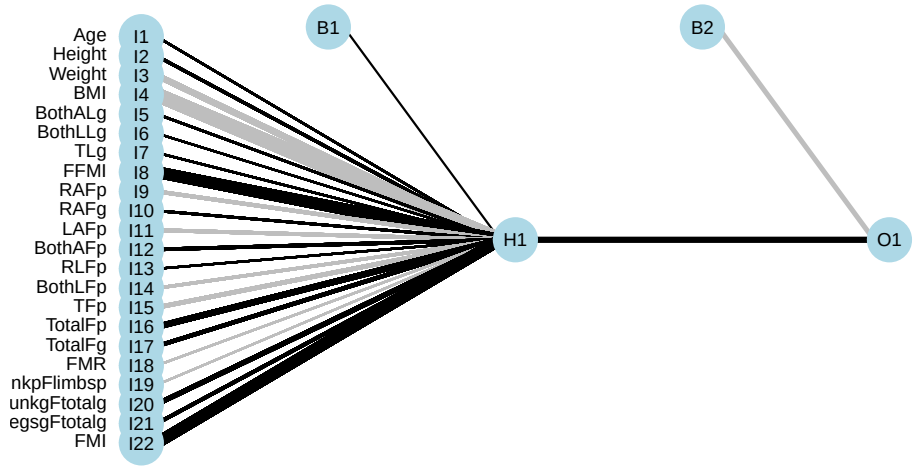
```
fat.model.trained$learner.model$call$formula = 'Lipodistrophy ~ .'
lean.model.trained$learner.model$call$formula = 'Sarcopenia ~ .'
bone.model.trained$learner.model$call$formula = 'Osteoporosis ~ .'

plotnet(fat.model.trained$learner.model, max_sp=T, cex_val=0.6, circle_cex=3)
```




```
plotnet(lean.model.trained$learner.model, max_sp=T, cex_val=0.6, circle_cex=3)
```




```
plotnet(bone.model.trained$learner.model, max_sp=T, cex_val=0.6, circle_cex=3)
```




```
print('FAT')
```

```
## [1] "FAT"
```

```
# The garson and olden algorithms do not admit nets with skip layes,
# so the importance of the Random forest will be displayed
# garson(fat.model.trained$learner.model)
# olden(fat.model.trained$learner.model$wts, struct = c(21,1,1),
#  x_lab = fat.model.trained$features)
#
# gar.fat <- garson(fat.model.trained$learner.model, bar_plot=F)
# old.fat <- olden(fat.model.trained$learner.model$wts, struct = c(21,1,1),
#  x_lab = fat.model.trained$features, bar_plot=F)
#
# knitr::kable(data.frame(garson=head(rownames(gar.fat)[order(
#  gar.fat$rel_imp, decreasing = T)]),
#   olden=head(rownames(old.fat)[order(abs(old.fat$importance), decreasing = T)])))
fat.model.rf <- setHyperPars(lrn.rf, par.vals=Fat.RRF$x)
fat.model.trained.rf <- train(fat.model.rf, traintask.fat)
fat.importance <- fat.model.trained.rf$learner.model$importance
fat.importance.o <- fat.importance[order(fat.importance, decreasing = T),]

knitr::kable(head(fat.importance.o), col.names=c('Importance'))
```

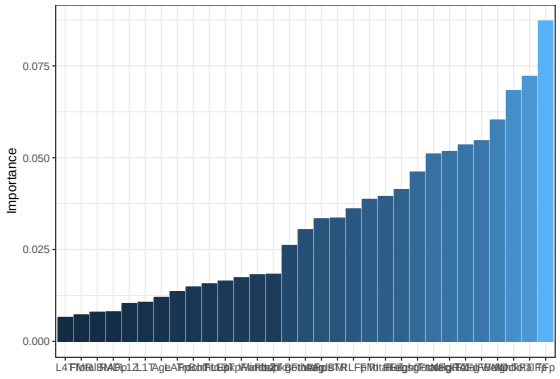
	Importance
Age	71.9
BMI	32.0

	Importance
WardsT	28.1
FFMI	27.4
Weight	25.4
L1Z	25.0

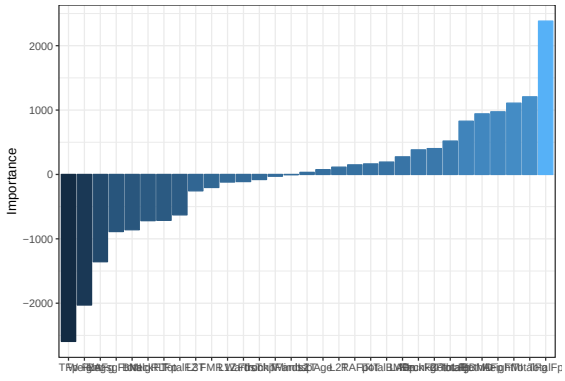
```
print('LEAN')
```

```
## [1] "LEAN"
```

```
garson(lean.model.trained$learner.model)
```




```
olden(lean.model.trained$learner.model$wts, struct = c(31,3,1),
      x_lab = lean.model.trained$features)
```




```
gar.lean <- garson(lean.model.trained$learner.model, bar_plot=F)
old.lean <- olden(lean.model.trained$learner.model$wts, struct = c(31,3,1),
                  x_lab = lean.model.trained$features, bar_plot=F)

knitr::kable(data.frame(garson=head(rownames(gar.lean)[order(
  gar.lean$rel_imp, decreasing = T)]), olden=head(rownames(old.lean)[order(
    abs(old.lean$importance), decreasing = T)])))
```

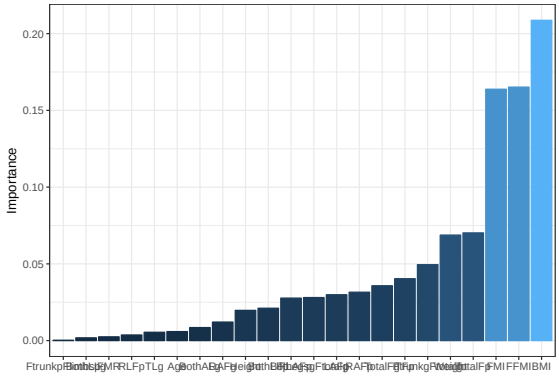
garson	olden
TFp	TFp
TotalFp	TotalFp
NeckFT	Weight
Weight	RAFg

garson	olden
TotalFBMD	TotalFg
RAFg	FMI

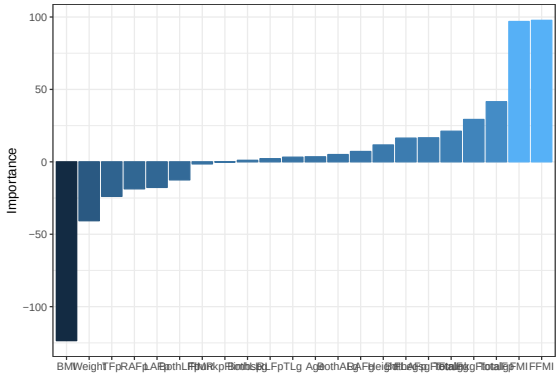
```
print('BONE')
```

```
## [1] "BONE"
```

```
garson(bone.model.trained$learner.model)
```




```
olden(bone.model.trained$learner.model$wts, struct = c(22,1,1),
      x_lab = bone.model.trained$features)
```




```
gar.bone <- garson(bone.model.trained$learner.model, bar_plot=F)
old.bone <- olden(bone.model.trained$learner.model$wts, struct = c(22,1,1),
                  x_lab = bone.model.trained$features, bar_plot=F)

knitr::kable(data.frame(garson=head(rownames(gar.bone)[order(
  gar.bone$rel_imp, decreasing = T)]), olden=head(rownames(old.bone)[order(
    abs(old.bone$importance), decreasing = T)])))
```

garson	olden
BMI	BMI
FFMI	FFMI
FMI	FMI
TotalFp	TotalFp
Weight	Weight
FtrunkgFtotalg	FtrunkgFtotalg

garson	olden