Estimation of obesity levels in individuals from Colombia, Peru and Mexico

Marwan Agourram (967349)

Accademic Year 2021-2022

Abstract The aim of this project is to analyse, develop predictions models and define clusters of the data ”Obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico” from UCI Machine Learning Repository[1].
The first part of the study is focused on developing prediction models using Supervised Learning techniques in order to predict the Obesity Level for each individual. The second part is focused on clustering and data reduction using Unsupervised Learning techniques.

Contents

[1]Estimation of obesity levels based on eating habits and physical condition . (2019). UCI Machine Learning Repository.

List of Figures

List of Tables

1 Part 1: Supervised Learning

1.1 Introduction

In this report I am going to use "*Dataset for estimation of obesity levels based on eating habits and physical condition in individuals from Colombia, Peru and Mexico*" paper from Mendoza F., et al. as a starting point. The paper presents data for the estimation of obesity level in individuals from Mexico, Perù and Colombia based on their eating habits and physical conditions.

1.1.1 Data Inspection

The data contains informations for the estimation of the obesity level in individuals from three different countries, with ages between 14 and 61 years old and different eating and physical conditions habits. The data was collected using a web platform with a survey where anonymous answered each question, specifying their age and gender and some physical attributes, such as height and weight.
These collected information end up consisting in a dataset with 17 attributes for 2,111 records. As mentioned above, four of these attributes are physical attributes for each individual:

- Gender: individual's gender (binary: "Male", "Female");
- Age: individual's age (numeric value: from 14 to 61);
- Height: individual's height in meters (numeric value: from 1.45 m to 1.98 m);
- Weight: individual's weight in kilograms (numeric value: from 39 kg to 173 kg).

The remaing 12 attributes are obtained from the survey's response to the reported questions:

- family_history_with_overweight: "Has a family member suffered or suffers from overweight?" (nominal: "yes", "no");
- FAVC: "Do you eat high caloric food frequently?" (nominal: "yes", "no");
- FCVC: "Do you usually eat vegetables in your meals?" (numeric: "1" - never, "2" - sometimes, "3" - always);
- NCP: "How many main meals do you have daily?" (numeric: "1" - between 1 or 2 meals, "2" - three meals, "3" - more than three meals);
- CAEC: "Do you eat any food between meals?" (nominal: "no", "sometimes", "frequently", "always");
- SMOKE: "Do you smoke?" (nominal: "yes", "no");
- CH2O: "How much water do you drink daily?" (numeric: "1" - less than 1 liter, "2" - 1 or 2 liters, "3" - more than 2 liters);
- SCC: "Do you monitor the calories you eat daily?" (nominal: "yes", "no");
- FAF: "How often do you have physical activity?" (numeric: "0" - no physical activity, "1" - 1 or 2 days, "2" - 2 or 3 days, "3" - 4 or 5 days):
- TUE: "How much time do you use technological devices such as cell phone, videogames, television, computer and others?" (numeric: "0" - 0-2 hours, "1" - 3-5 hours, "2" - more than 5 hours);
- CALC: "How often do you drink alcohol?" (nominal: "no", "sometimes", "frequently", "always");
- MTRANS: "Which transportation do you usually use?" (nominal: "automobile", "motorbike", "bike", "public transportation", "walking").

Then the class variable NObeyesdad was created using the Body Mass Index (BMI) equation as stated by the World Health Organization[2]

$$\text{BMI} = \frac{\text{Weight}}{\text{Height}^2}, \tag{1}$$

which values allows us to obtain the following nutritional status:

- Insufficient Weight: BMI < 18.5;
- Normal Weight: $18.5 \leq$ BMI ≤ 24.9;
- Overweight Level I and II: $25.0 \leq$ BMI ≤ 29.9;
- Obesity Type I: $30 \leq$ BMI ≤ 34.9;
- Obesity Type II: $35.0 \leq$ BMI ≤ 39.9;
- Obesity Type III: BMI ≥ 40.

In Figure 1 we can observe the distribution of all the numeric variable, the number of missing values, and the range of the data with their respective mean and standard deviation.

```
> describe_distribution(data)
Variable |  Mean |    SD |   IQR |           Range |  Skewness | Kurtosis |    n | n_Missing
--------------------------------------------------------------------------------------------
Age      | 24.32 |  6.36 |  6.00 |  [14.00, 61.00] |      1.52 |     2.80 | 2111 |         0
Height   |  1.70 |  0.09 |  0.14 |    [1.45, 1.98] | -9.12e-03 |    -0.57 | 2111 |         0
Weight   | 86.59 | 26.19 | 42.06 | [39.00, 173.00] |      0.26 |    -0.70 | 2111 |         0
FCVC     |  2.42 |  0.58 |  1.00 |    [1.00, 3.00] |     -0.43 |    -0.71 | 2111 |         0
NCP      |  2.62 |  0.73 |  0.00 |    [1.00, 3.00] |     -1.56 |     0.67 | 2111 |         0
CH2O     |  2.01 |  0.69 |  0.00 |    [1.00, 3.00] |     -0.02 |    -0.89 | 2111 |         0
FAF      |  1.01 |  0.90 |  2.00 |    [0.00, 3.00] |      0.46 |    -0.71 | 2111 |         0
TUE      |  0.66 |  0.67 |  1.00 |    [0.00, 2.00] |      0.52 |    -0.76 | 2111 |         0
```

Figure 1: Distribution of all numerical variables in the data.

Before going any further, I would like to highlight the fact that the physical features (Age, Height and Weight) have a large difference in variance and in the standard deviation compared to the other numerical variables. Here in Figure 2 we can have a look at these three variable distributions, represented with an histogram. In the following pages I will deal with these three variables.

[2]https://www.who.int/europe/news-room/fact-sheets/item/a-healthy-lifestyle—who-recommendations

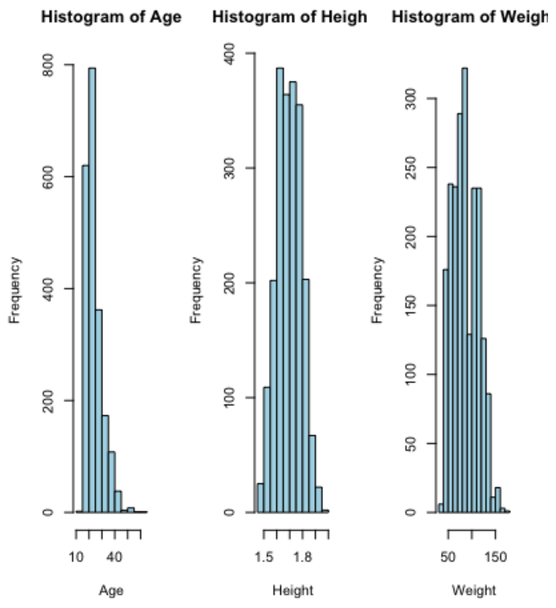


Figure 2: Distribution of Age, Height and Weight.

1.1.2 Data Cleaning and Pre-processing

This report is based on a dataset which is composed by qualitative variables and quantitative ones. As we have seen while looking at the different features, most of the variables are responses based on multiple-choices answers, while some of the variables are a representation of the physical features of each individual who answered the survey.
In Figure 1 is evident that some of the numerical variables have different distributions with respect to all the other variables. In particular, in Figure 2 I displayed the distribution of variables that are more noticeable. The boxplot representations of these three variables are shown in Figure 3. It is easy to see that the variables Age and Weight have some outliers. In order to detech and remove these outliers, I applied the Interquartile Range rule. First, I computed the first and third quartile for Age and Weight, then I computed the lower limit (Equation 2) and upper limit (Equation 3) for both variables. An observation is considered to be an outlier when it exceeds the upper limit or it is lower than the lower limit.

$$\text{LL} = Q_1 - 1.5 * IQR, \tag{2}$$

$$\text{UP} = Q_3 + 1.5 * IQR \tag{3}$$

From computing the interquartile range, it results that 161 observations are considered to be outliers and, therefore, I proceeded to remove them from the dataset. Figure 3 shows the distribution for the three considered numerical variables after outliers have been removed. Therefore, I will mainly deal with obseervations coming from individuals whose age ranges between 14 and 35 years old, excluding the elder individuals. The rationale behind this choice is that the main goal of this part is to predict potential obesity problem in an individual based on their habits, and including these outliers may lead to some wrong conclusions. Also, I excluded individuals whose weight was exceeding the interquantile range. Once again, the rationale behind this decision is to ensure that the following conclusions are not affected by noise.

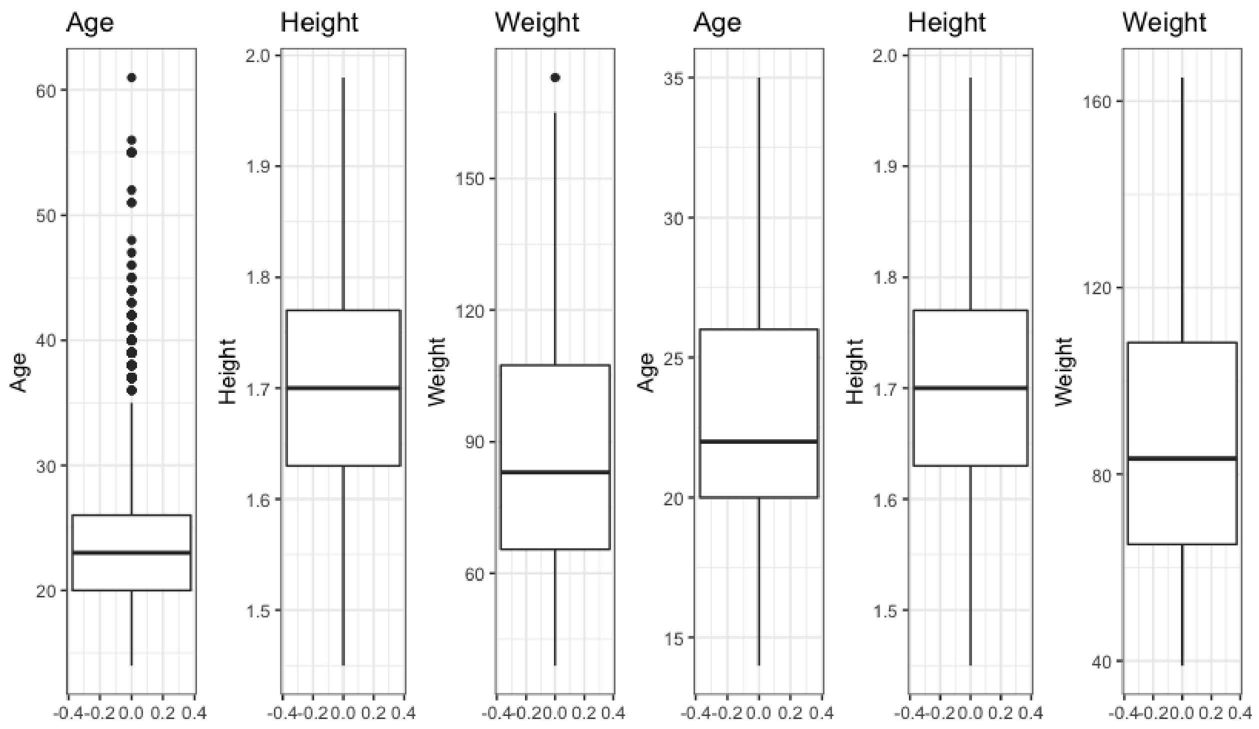


Figure 3: *Left*: Boxplot with outliers included. *Right*: Boxplot without outliers.

After removing the outliers observations, I proceeded to apply one-hot encoding technique on those categorical features that has a binary response class: Gender, family_history_with_overweight, FAVC, SMOKE, SCC, and MTRANS. One-hot encoding will generate dummy variables for each of the considered features. Then, I proceeded to apply ordinal encoding technique for the remaining categorical variables, that is, the ones which response classes were not binary: CAEC, CALC, and NObeyesdad.
Figure 4 reports the distribution of all the features in the dataset after applying previously mentioned techniques. It is worth to notice that now our data set consists of 26 features, due to one-hot encoding on binary categorical response class features, and 1950 observations, due to outliers detection and removal.

```
> describe_distribution(data_iqr)
Variable                           |     Mean |    SD |   IQR |           Range | Skewness | Kurtosis |    n | n_Missing
-----------------------------------------------------------------------------------------------------------------------
GenderFemale                       |     0.49 |  0.50 |  1.00 |    [0.00, 1.00] |     0.06 |    -2.00 | 1950 |         0
GenderMale                         |     0.51 |  0.50 |  1.00 |    [0.00, 1.00] |    -0.06 |    -2.00 | 1950 |         0
Age                                |    23.00 |  4.38 |  6.00 |  [14.00, 35.00] |     0.79 |     0.02 | 1950 |         0
Height                             |     1.71 |  0.09 |  0.14 |    [1.45, 1.98] | 6.95e-03 |    -0.52 | 1950 |         0
Weight                             |    86.59 | 26.76 | 43.25 | [39.00, 165.06] |     0.23 |    -0.80 | 1950 |         0
family_history_with_overweightno   |     0.19 |  0.39 |  0.00 |    [0.00, 1.00] |     1.58 |     0.48 | 1950 |         0
family_history_with_overweightyes  |     0.81 |  0.39 |  0.00 |    [0.00, 1.00] |    -1.58 |     0.48 | 1950 |         0
FAVCno                             |     0.12 |  0.32 |  0.00 |    [0.00, 1.00] |     2.37 |     3.62 | 1950 |         0
FAVCyes                            |     0.88 |  0.32 |  0.00 |    [0.00, 1.00] |    -2.37 |     3.62 | 1950 |         0
FCVC                               |     2.43 |  0.59 |  1.00 |    [1.00, 3.00] |    -0.49 |    -0.66 | 1950 |         0
NCP                                |     2.62 |  0.73 |  0.00 |    [1.00, 3.00] |    -1.58 |     0.74 | 1950 |         0
SMOKEno                            |     0.98 |  0.14 |  0.00 |    [0.00, 1.00] |    -6.86 |    45.14 | 1950 |         0
SMOKEyes                           |     0.02 |  0.14 |  0.00 |    [0.00, 1.00] |     6.86 |    45.14 | 1950 |         0
CH2O                               |     2.02 |  0.69 |  0.00 |    [1.00, 3.00] |    -0.03 |    -0.89 | 1950 |         0
SCCno                              |     0.95 |  0.21 |  0.00 |    [0.00, 1.00] |    -4.22 |    15.84 | 1950 |         0
SCCyes                             |     0.05 |  0.21 |  0.00 |    [0.00, 1.00] |     4.22 |    15.84 | 1950 |         0
FAF                                |     1.02 |  0.89 |  2.00 |    [0.00, 3.00] |     0.44 |    -0.72 | 1950 |         0
TUE                                |     0.70 |  0.68 |  1.00 |    [0.00, 2.00] |     0.44 |    -0.81 | 1950 |         0
MTRANSAutomobile                   |     0.16 |  0.37 |  0.00 |    [0.00, 1.00] |     1.87 |     1.51 | 1950 |         0
MTRANSBike                         | 3.08e-03 |  0.06 |  0.00 |    [0.00, 1.00] |    17.96 |   320.83 | 1950 |         0
MTRANSMotorbike                    | 4.62e-03 |  0.07 |  0.00 |    [0.00, 1.00] |    14.63 |   212.22 | 1950 |         0
MTRANSPublic_Transportation        |     0.81 |  0.40 |  0.00 |    [0.00, 1.00] |    -1.55 |     0.39 | 1950 |         0
MTRANSWalking                      |     0.03 |  0.17 |  0.00 |    [0.00, 1.00] |     5.70 |    30.56 | 1950 |         0
```

Figure 4: Distribution of all the features, including the dummies, after one-hot and ordinal encoding was applied.

Finally, some variables have been rescaled in order to let the models learn faster and to better perform in prediction, while others (such as Age, Height, Weight) were not scaled.

1.2 Techniques and implemented algorithms

In this first part I applied Supervised Learning techniques in order to analyse the data and generate prediction models in order to infer about the obesity level in the considered individuals. Each implemented model will be explained in the following pages and, finally, I will provide a summarized table for all the considered models in order to comprehend which one performed better.
Before diving into more detailed explanations, I computed the correlation between the data in order to highlight the dummy variable trap, a problem that may arise when techniques such as one-hot encoding or ordinal encoding are applied to categorical variables. This relatively known problem turns out to be easily manageable, since one of the best approaches is to consider only $K - 1$ dummy variables when encoding a categorical variable with K response classes. Therefore, I proceed to remove the following dummy variables:

- ”GenderMale”;
- ”family_history_with_overweightno”;
- ”FAVCno”;
- ”Smokeno”;
- ”SCCno”;
- ”MTRANSWalking”, the only dummy variable excluded from the ordinal encoding step.

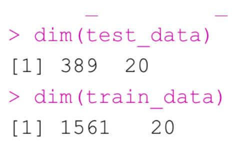
The Supervised Learning approaches that I implemented in order to make inference are:

- Linear Regression;
- Stepwise Linear Regression;
- LASSO Linear Regression;
- Decision Tree:
- Random Forest.

1.3 Linear Regression

1.3.1 Simple Linear Regression

At this point our data set has 20 features and 1.950 observations. First of all, we start by splitting the dataset into a training set, which is randomly generated by including 80% of the total observations, and a test set, which has the remaining 20% of the observations.



```
> dim(test_data)
[1] 389  20
> dim(train_data)
[1] 1561   20
```

Figure 5: Training set and Test set dimensions.

The first Linear Regression model that we run includes all the 19 features and it is trained on the train set.

Figure 7 to Figure 9 shows some of the diagnostic on the Linear Regression model. It is noticeable a collinearity problem, also confirmed in Figure 8.
As we can see from Figure 6, some of the features included in the Linear Regression model turns out to be statistically unsignificant, that is, we may exclude them from our Linear Regression model in order to improve our fit. Moreover, we can see that approximately 96% of the variance is explained by the considered features.
By diving deeper into the generated fit we can notice that the main important features regards physical attributes, such as Height and Weight, but also the Age of the individuals. Surprisingly, the model did not consider various health habits to be statistically significant, although individual's Gender seems to have a little role in predicting obesity, with Males being more subject to this risk.
It seems that having previous or current cases of obesity in the family increases the risk of incurring into obesity problems, but also individual's age (i.e., the older it is, the higher the risk of obesity). Lastly, consuming more meals during the day may lead to higher risk.

In the next sections we will deal with the unsignificant features by implementing Stepwise and LASSO apporaches to the Linear Regression model.

```
Call:
lm(formula = NObeyesdad ~ ., data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.15433 -0.27280  0.00656  0.26539  1.74405 

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)                        8.4058397  0.3205138  26.226  < 2e-16 ***
GenderFemale                      -0.0391868  0.0148161  -2.645  0.00826 ** 
Age                                0.0297857  0.0031863   9.348  < 2e-16 ***
Height                            -7.4044427  0.1877735 -39.433  < 2e-16 ***
Weight                             0.0762834  0.0006236 122.334  < 2e-16 ***
family_history_with_overweightyes  0.1296325  0.0133242   9.729  < 2e-16 ***
FAVCyes                            0.0073720  0.0118607   0.622  0.53433    
FCVC                               0.0046327  0.0119734   0.387  0.69887    
NCP                                0.0303537  0.0115722   2.623  0.00880 ** 
CAEC                              -0.0659815  0.0119822  -5.507 4.28e-08 ***
SMOKEyes                          -0.0168590  0.0119268  -1.414  0.15770    
CH2O                               0.0010716  0.0114044   0.094  0.92515    
SCCyes                            -0.0090180  0.0113328  -0.796  0.42631    
FAF                               -0.0788982  0.0121400  -6.499 1.09e-10 ***
TUE                               -0.0106891  0.0113773  -0.940  0.34761    
CALC                              -0.0495735  0.0118756  -4.174 3.15e-05 ***
MTRANSAutomobile                  -0.0308286  0.0267811  -1.151  0.24986    
MTRANSBike                        -0.0032149  0.0113243  -0.284  0.77653    
MTRANSMotorbike                    0.0108023  0.0113201   0.954  0.34010    
MTRANSPublic_Transportation        0.0514304  0.0268500   1.915  0.05562 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4303 on 1541 degrees of freedom
Multiple R-squared:  0.9558,	Adjusted R-squared:  0.9553 
F-statistic:  1756 on 19 and 1541 DF,  p-value: < 2.2e-16
```

Figure 6: Linear Regression model on all of the 19 features, with response variable NObeyesdad.

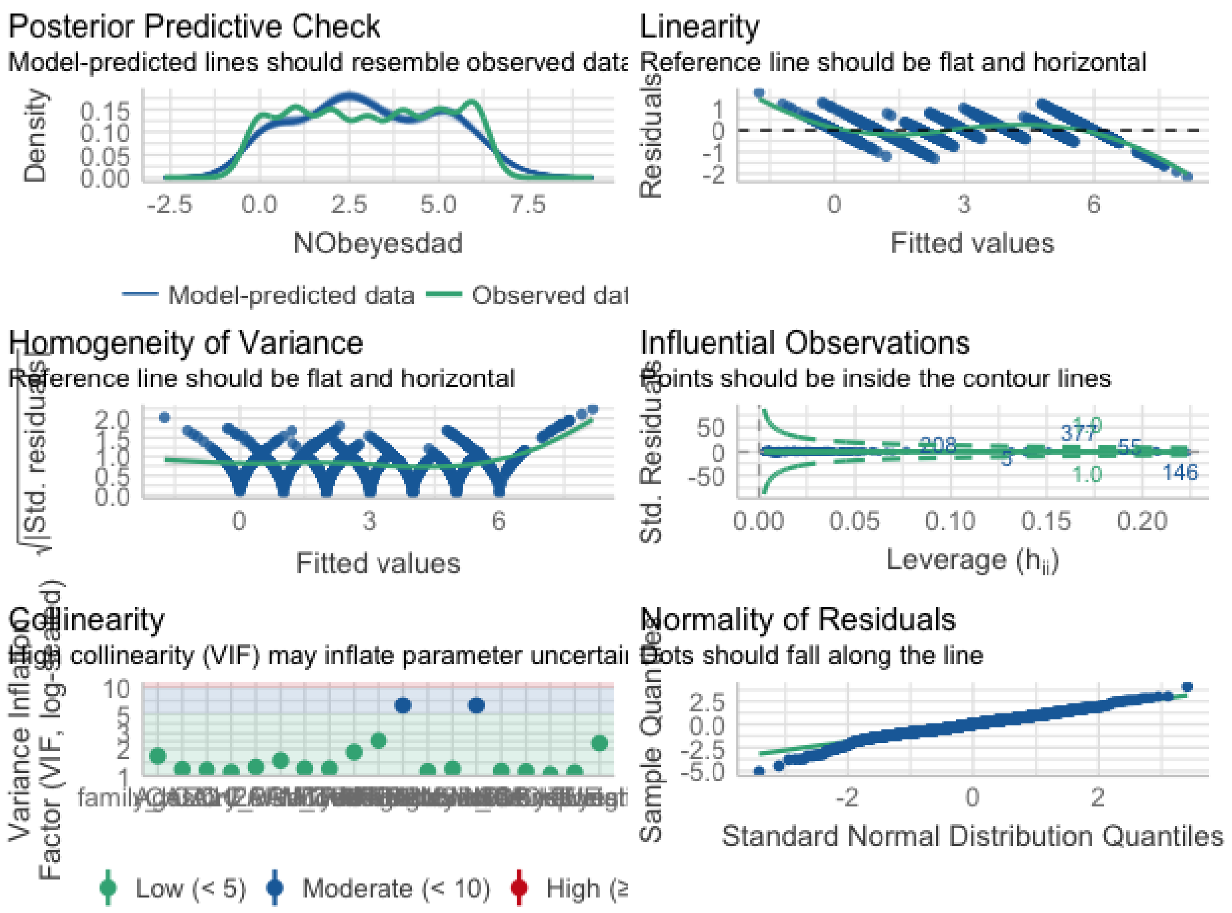
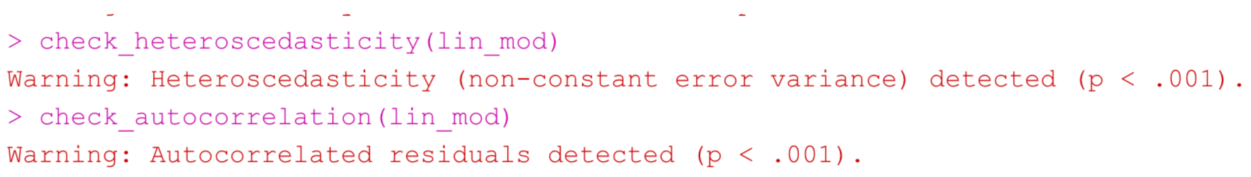
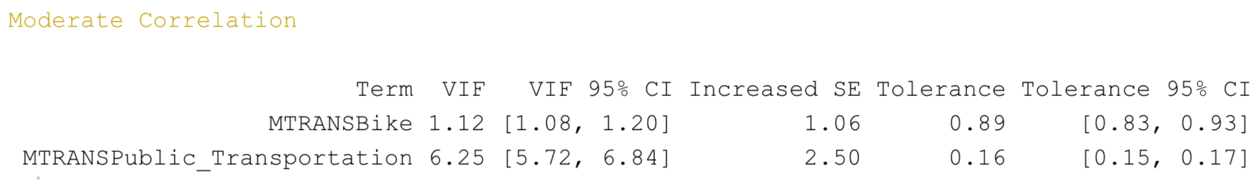


Figure 7: Linear Regression Model performance on different aspects.



```
> check_heteroscedasticity(lin_mod)
Warning: Heteroscedasticity (non-constant error variance) detected (p < .001).
> check_autocorrelation(lin_mod)
Warning: Autocorrelated residuals detected (p < .001).
```

Figure 8: Heteroscedasticity and Autocorrelation check on Linear Regression model.



```
Moderate Correlation

                        Term  VIF   VIF 95% CI Increased SE Tolerance Tolerance 95% CI
                 MTRANSBike 1.12 [1.08, 1.20]         1.06      0.89     [0.83, 0.93]
 MTRANSPublic_Transportation 6.25 [5.72, 6.84]         2.50      0.16     [0.15, 0.17]
```

Figure 9: Moderate correlations resulting from Autocorrelation check on Linear Regression model.

1.3.2 Stepwise Regression

Stepwise linear regression is a method of fitting multiple regression variables while simultaneously removing the variables that results to be less explanatory. As I mentioned above, performing simple Linear Regression on all of the 19 considered features resulted in a regression in model in which approximately 10 features where unsignificant. By implementing the stepwise approach, that is, a mixed technique that implement both backward and forward selection, I was able to achieve the same exact result as performing a simple Linear Regression model. As shown in Figure 10, in the resulted model almost all the selected features results to be effectively significant, with an exception for MTRANSMotorbike and SMOKEyes. Moreover, we can observe how the number of features decreased from 19 to 12.

```
> lin_step_sum

Call:
lm(formula = NObeyesdad ~ GenderFemale + Age + Height + Weight + 
    family_history_with_overweightyes + NCP + CAEC + SMOKEyes + 
    FAF + CALC + MTRANSMotorbike + MTRANSPublic_Transportation, 
    data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.16370 -0.27167  0.00343  0.26847  1.75450 

Coefficients:
                                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)                       8.3648859  0.3164537  26.433  < 2e-16 ***
GenderFemale                     -0.0376138  0.0143121  -2.628  0.00867 ** 
Age                               0.0298959  0.0030328   9.857  < 2e-16 ***
Height                           -7.3897537  0.1862685 -39.673  < 2e-16 ***
Weight                            0.0764353  0.0006002 127.340  < 2e-16 ***
family_history_with_overweightyes  0.1294937  0.0131444   9.852  < 2e-16 ***
NCP                               0.0294123  0.0115088   2.556  0.01069 *  
CAEC                             -0.0661820  0.0117561  -5.630 2.14e-08 ***
SMOKEyes                         -0.0182464  0.0117915  -1.547  0.12197    
FAF                              -0.0791130  0.0119141  -6.640 4.32e-11 ***
CALC                             -0.0496976  0.0118033  -4.210 2.69e-05 ***
MTRANSMotorbike                   0.0157269  0.0105327   1.493  0.13560    
MTRANSPublic_Transportation       0.0797287  0.0127766   6.240 5.63e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4297 on 1548 degrees of freedom
Multiple R-squared:  0.9558,    Adjusted R-squared:  0.9554 
F-statistic:  2786 on 12 and 1548 DF,  p-value: < 2.2e-16
```

Figure 10: Stepwise Linear Regression model on a selected subset of features, with response variable NObeyedad.

As we can see from Figure 11 almost everything remains unchanged with respect to the simple Linear Re-

gression model. The only difference with the simple Linear Regression model is that the stepwise regression does not include any feature that have moderate or high level of correlation, as opposed to the simple Linear Regression model in which two variables where detected to be moderately correlated.
Also the portion of explained variance is exactly the same as the one in the simple Linear Regression model, achieving an $R^2 \approx 96\%$.

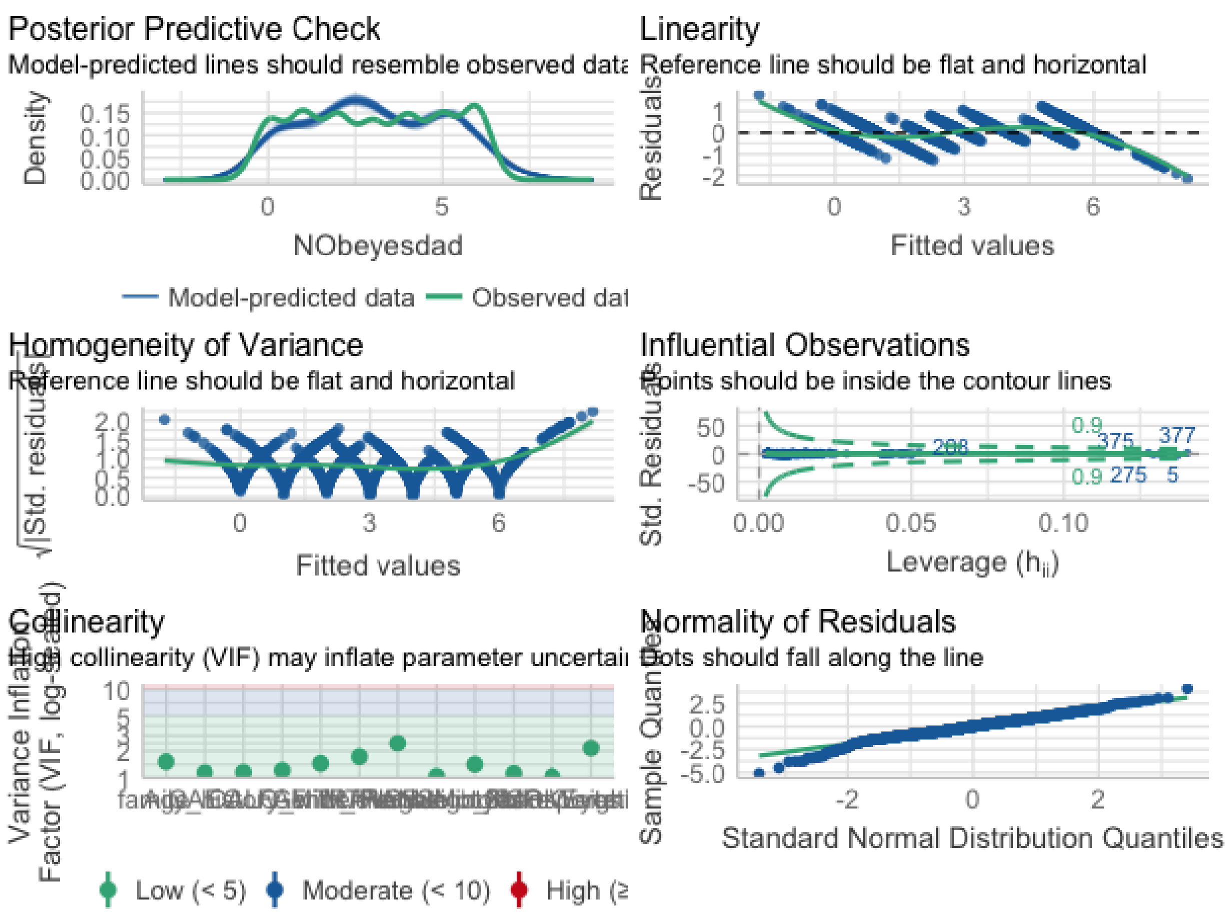


Figure 11: Stepwise Linear Regression model performance of different aspects.

1.3.3 LASSO Regression

LASSO, which stands for Least Absolute Shrinkage and Selection Operator, is particular type of regression model in which are implemented shrinkage techniques. These techniques allows to shrink the coefficient estimates towards a central point (which is generally 0), and to obtain sparse models, that is, models in which we have fewer features.

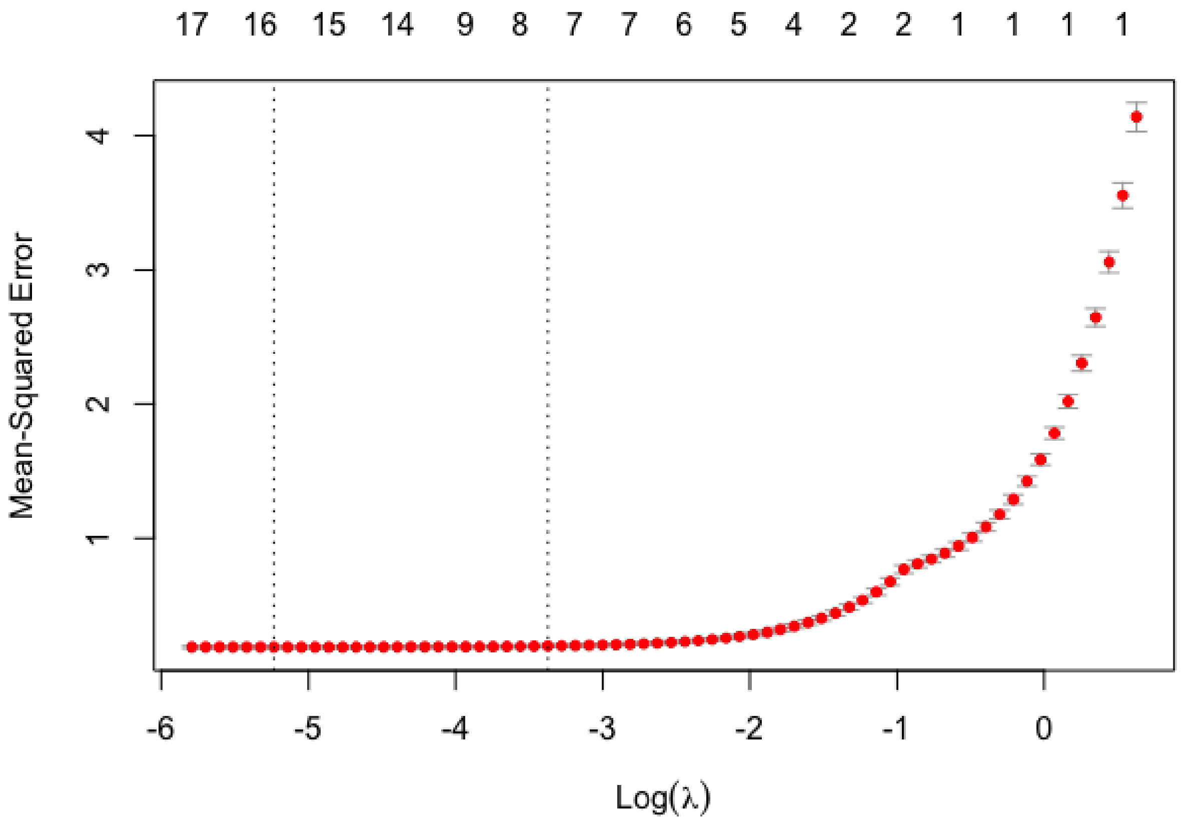
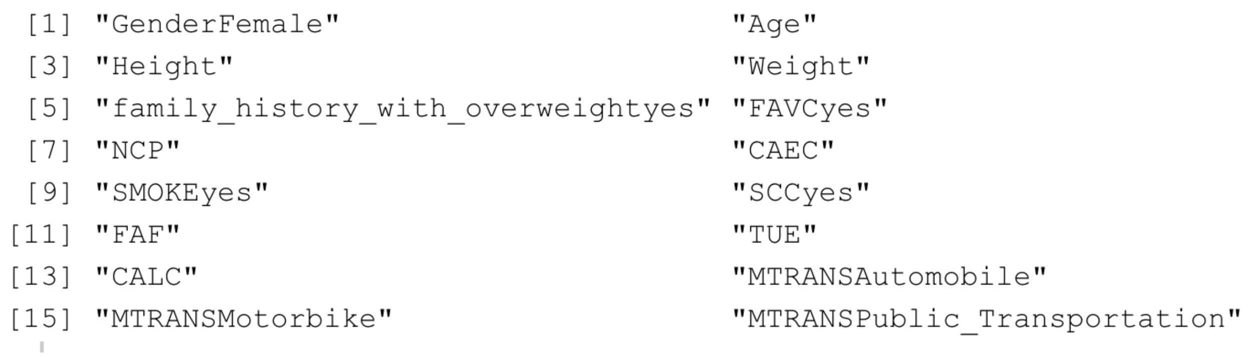


Figure 12: Plot of the minimum $\log(\lambda)$ selected through cross validation on the training set

After we had computed the minimum value for the penalty term λ, the LASSO selected the features that are reported in Figure 13.



```
 [1] "GenderFemale"                       "Age"
 [3] "Height"                             "Weight"
 [5] "family_history_with_overweightyes" "FAVCyes"
 [7] "NCP"                                "CAEC"
 [9] "SMOKEyes"                           "SCCyes"
[11] "FAF"                                "TUE"
[13] "CALC"                               "MTRANSAutomobile"
[15] "MTRANSMotorbike"                    "MTRANSPublic_Transportation"
```

Figure 13: Selected features using LASSO approach.

It is easy to see that in this case LASSO selects more features than the stepwise approach. Figure 14 does not highlight any major difference with respect to the simple Linear Regression model. Furthermore, Figure 15 highlights two regressors with moderate correlation, the same ones that was captured in the simple Linear Regression model.

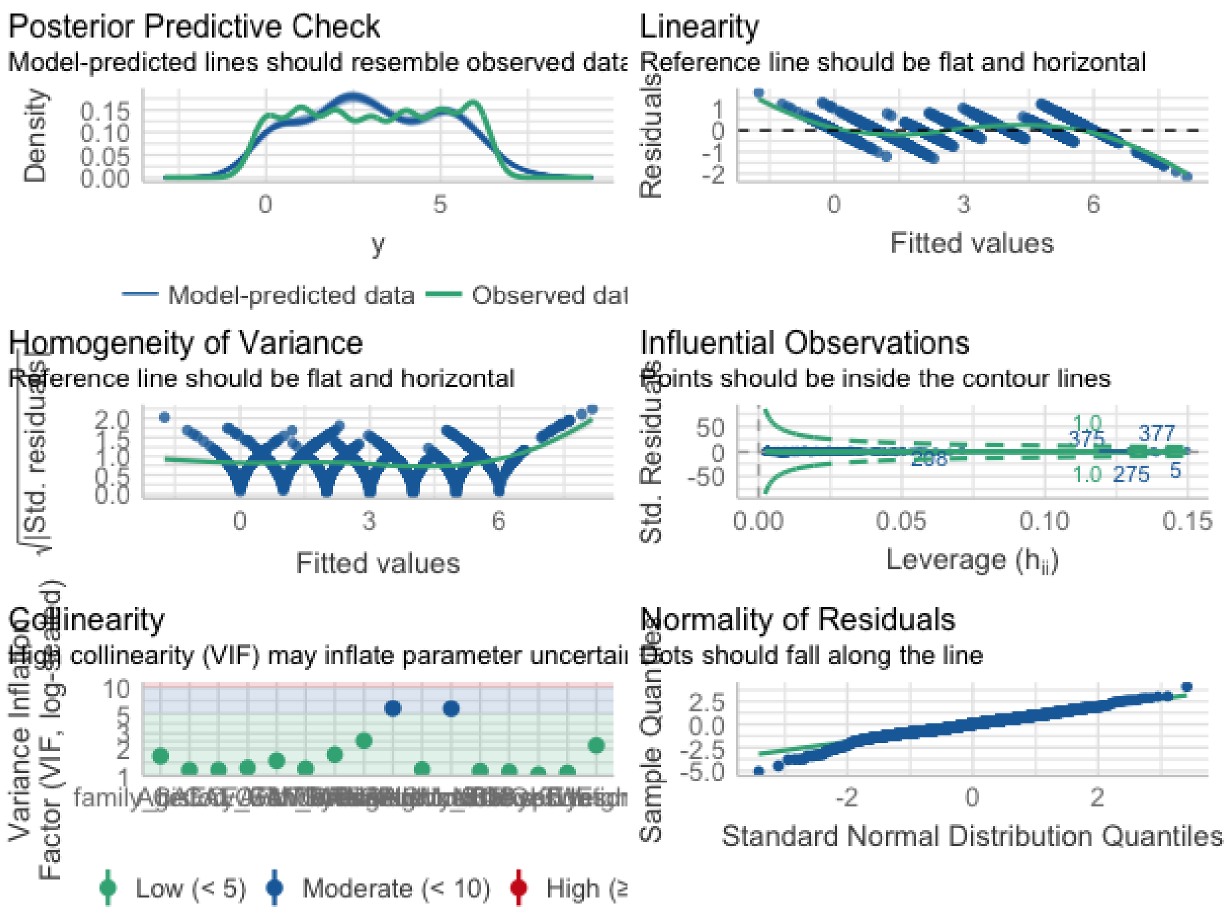


Figure 14: LASSO Linear Regression model performance of differente aspects.

```
Moderate Correlation

                           Term  VIF   VIF 95% CI Increased SE Tolerance Tolerance 95% CI
               MTRANSMotorbike 1.18 [1.12, 1.26]         1.09      0.85     [0.79, 0.89]
 MTRANSPublic_Transportation 5.72 [5.23, 6.26]         2.39      0.17     [0.16, 0.19]
```

Figure 15: Moderate correlations resulting from Autocorrelation check on LASSO Linear Regression model .

Finally, Figure 16 shows the LASSO regression model coefficients with the relative p-values. In this case no variables were excluded with respect to the simple Linear Regression model. Moreover, it seems that no big difference had occured at all, even in the R^2 being approximately equal to 96%.

```
Call:
lm(formula = fmla, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.15513 -0.27285  0.00809  0.26682  1.74999 

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)    
(Intercept)                        8.3986555  0.3196590  26.274  < 2e-16 ***
GenderFemale                      -0.0375599  0.0143338  -2.620  0.00887 ** 
Age                                0.0297837  0.0031793   9.368  < 2e-16 ***
Height                            -7.4031530  0.1873537 -39.514  < 2e-16 ***
Weight                             0.0763414  0.0006052 126.139  < 2e-16 ***
family_history_with_overweightyes  0.1294656  0.0132694   9.757  < 2e-16 ***
FAVCyes                            0.0074027  0.0118160   0.627  0.53108    
NCP                                0.0302945  0.0115483   2.623  0.00879 ** 
CAEC                              -0.0656003  0.0118337  -5.544 3.48e-08 ***
SMOKEyes                          -0.0167839  0.0118839  -1.412  0.15806    
SCCyes                            -0.0086872  0.0112730  -0.771  0.44105    
FAF                               -0.0785613  0.0119945  -6.550 7.82e-11 ***
TUE                               -0.0110751  0.0113138  -0.979  0.32778    
CALC                              -0.0495580  0.0118385  -4.186 3.00e-05 ***
MTRANSAutomobile                  -0.0288208  0.0256868  -1.122  0.26203    
MTRANSMotorbike                    0.0112571  0.0112225   1.003  0.31598    
MTRANSPublic_Transportation        0.0535751  0.0256565   2.088  0.03695 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4299 on 1544 degrees of freedom
Multiple R-squared:  0.9558,    Adjusted R-squared:  0.9554 
F-statistic:  2089 on 16 and 1544 DF,  p-value: < 2.2e-16
```

Figure 16: LASSO Regression model on a selected subset of features, with response variable NObeyedad.

1.3.4 Linear Regression models comparison

We have implemented a simple Linear Regression model in order to analyse and predict the obesity level in our data. Then we adapted our linear model by implementing two different techniques that aimde at reshaping our base model by selecting only the most important regressors or by shrunking those features that does not improve our fit.
At this point we are interested in comparing those three different models, in order to comprehend which approach leads to better results.

```
# Comparison of Model Performance Indices

Name          | Model |    R2 | R2 (adj.) |  RMSE | Sigma | AIC weights | BIC weights | Performance-Score
-----------------------------------------------------------------------------------------------------------
lin_mod_step  |    lm | 0.956 |     0.955 | 0.428 | 0.430 |       0.922 |       1.000 |            66.67%
lin_lasso     |    lm | 0.956 |     0.955 | 0.428 | 0.430 |       0.074 |    1.80e-06 |            56.08%
lin_mod       |    lm | 0.956 |     0.955 | 0.428 | 0.430 |       0.004 |    3.31e-11 |            33.33%
```

Figure 17: Simple Linear Regression, LASSO Linear Regression and Stepwise Linear Regression comparison.

In Figure 17 are reported the various statistic for each of the three considered models. As we have already stated various time, the three models have the same R^2. Also the RMSE turns out to be the same for the three models. The only noticeable differences are in the AIC weights[3], which represents the model's relative probability of leading good results. Moreover, Stepwise Linear Regression approach register the highest "Performance Score"[4], almost twice as the simple Linear Regression model. This could be attributed to the fact that the Stepwise approach selected only the effectively significant variables, excluding those that did not have a high explanatory power. Also the LASSO Linear Regression registered a good "Performance Score" with respect to the baseline, even though 16 of the 19 features were included.

Finally, Figure 18 allows us to visualize the results of the comparison using a Spider plot.

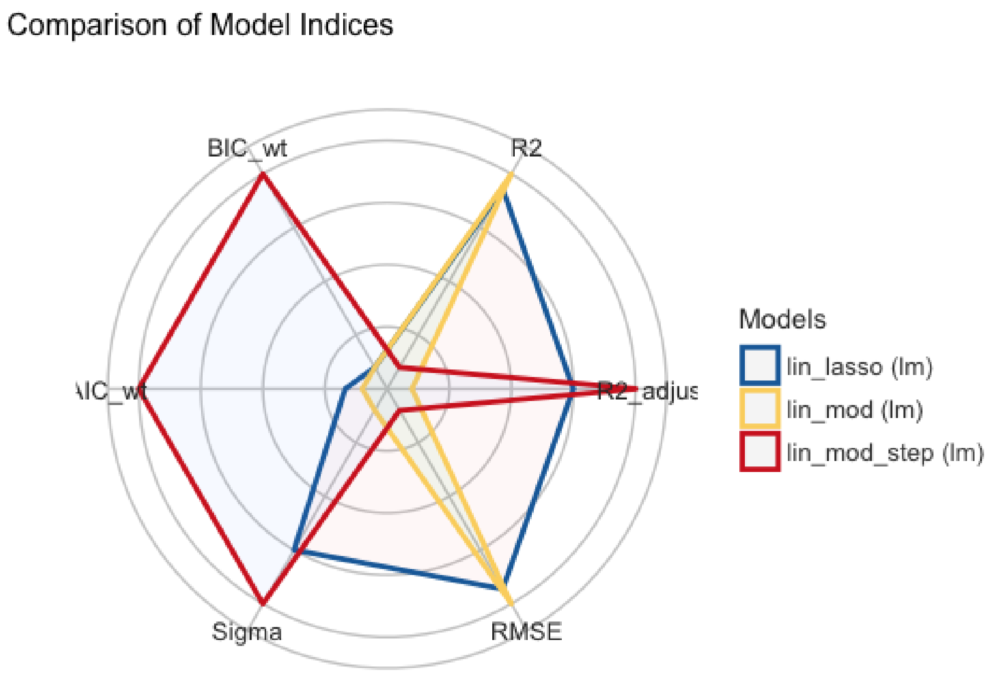


Figure 18: Simple Linear Regression, LASSO Linear Regression and Stepwise Linear Regression comparison using a Spider web plot.

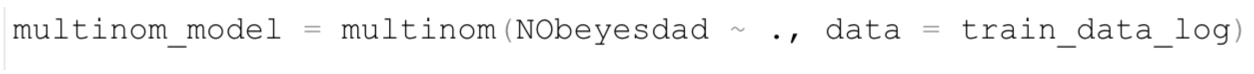
[3]https://link.springer.com/content/pdf/10.3758/BF03206482.pdf

[4]This score ranges from 0 to 100%. Note that all score value do not necessarily sum up to 100. Rather, calculation is based on normalizing all indices (i.e. rescaling them to a range from 0 to 1), and taking the mean value of all indices for each model. This is a rather quick heuristic, but might be helpful as exploratory index.

1.4 Classification

In statistics, the logistic model is a statistical tool that models the probability of one event taking place by having the log-odds for the event to be a linear combination of one or more indipenedent variables. On the other hand, Logistic Regression consists in estimating the parameters of a logistic model.
In this case, the response feature NObeyesdad is a categorical variable, that is, it has a discrete number of possible outcomes. Moreover, in this particular case the response features does not happen to be a binary response variable (i.e. it only has $K = 2$ response classes), but rather it has $K = 7$ different response classes. I will briefly dive into a specific type of classification algorithm, named Multinomial Logistic Regression.
The main difference with respect to the binary Logistic Regression is that Multinomial Logistic Regression allows to model the probability of an event that has multiclass outcomes ($K > 2$). In order to implement this model I used the *nnet*[5] package from Rstudio.

Before diving any further into our Multinomial Logistic Regression model, I decided to duplicate the previously created training and test set. This will be helpful later on, since we will develop the model without encoding and factorizing our response variable NObeyesdad, utilizing the function *relevel()*[6] to assign different values to our classes in the response variable.



```
multinom_model = multinom(NObeyesdad ~ ., data = train_data_log)
```

Figure 19: Multinomial Logistic Regression call in Rstudio.

Figure 19 shows the Multinomial Logistic Regression model call in Rstudio. Once we trained our model, we applied the exponential function to the coefficients in order to obtain the relative odds. The resulting model applied on the training set turns out to have an accuracy[7] of 96.82% on the train data.

R code 1.1: Trained Multinomial Logistic Regression outcome on train and test data.

```
> tab = table(train_data_log$NObeyesdad, train_data_log$ClassPredicted)
> tab

                        Insufficient_Weight Normal_Weight Overweight_Level_I
                          Overweight_Level_II
  Insufficient_Weight                   225             6                  0
                         0
  Normal_Weight                           7           207                  8
                         0
  Overweight_Level_I                      0            11                183
                         4
  Overweight_Level_II                     0             0                  2
                       216
  Obesity_Type_I                          0             0                  0
                         3
  Obesity_Type_II                         0             0                  0
                         0
  Obesity_Type_III                        0             0                  0
                         0

                        Obesity_Type_I Obesity_Type_II
  Insufficient_Weight                0               0
  Normal_Weight                      0               0
  Overweight_Level_I                 0               0
  Overweight_Level_II                2               0
```

[5]https://cran.r-project.org/web/packages/nnet/nnet.pdf
[6]The levels of a factor are re-ordered so that the level specified by ref is first and the others are moved down. This is useful for contr.treatment contrasts which take the first level as the reference.
[7]Sum of the diagonal observations divided by the sum of the total observations.

```
  Obesity_Type_I                  218              0
  Obesity_Type_II                   0            259
  Obesity_Type_III                  0              0
> round((sum(diag(tab))/sum(tab))*100,2)
[1] 96.82

> tab_1 = table(test_data_log$NObeyesdad, test_data_log$ClassPredicted)# Building
    classification table
> tab_1

                         Insufficient_Weight Normal_Weight Overweight_Level_I
                             Overweight_Level_II Obesity_Type_I
  Insufficient_Weight                     46             3                  0
                           0              0
  Normal_Weight                            1            45                  2
                           0              0
  Overweight_Level_I                       0             6                 47
                           3              0
  Overweight_Level_II                      0             0                  4
                          56              3
  Obesity_Type_I                           0             0                  0
                           1             45
  Obesity_Type_II                          0             0                  0
                           2              0
  Obesity_Type_III                         0             0                  0
                           0              0

                         Obesity_Type_II
  Insufficient_Weight                  0
  Normal_Weight                        1
  Overweight_Level_I                   0
  Overweight_Level_II                  0
  Obesity_Type_I                       1
  Obesity_Type_II                     62
  Obesity_Type_III                     0
> round((sum(diag(tab_1))/sum(tab_1))*100,2)
[1] 91.77
```

Then, in order to test our trained model, we applied the Multinomial Logistic Regression to the test data. The resulting accuracy, computed as before, is 91.77%.

1.5 Beyond Linearity

Now I will move to a different class of Supervised Learning techniques. This kind of approaches have the same basis as the ones presented until this point, except from the fact that they allow to consider non-linear effects on the model. In particular, this kind of model has the ability to intercept different interactions between features that a simple linear model is not able to consider, therefore improving the interpretability of the model. This advantage comes at a cost, that is, the increase in complexity.

1.5.1 Decision Trees

Decision tree is a Supervised Learning technique used as a predictive model to draw inference from a set of observations.
Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into continously smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes, which represents values for the attribute tested, and leaf nodes, which represent a decision on the numerical target.
In this case I did not use the *tree* package in Rstudio, but rather the *rpart*[8] package.

[8]https://cran.r-project.org/web/packages/rpart/rpart.pdf.

R code 1.2: Decision Regression Tree output in Rstudio.

```
> printcp(tree_1)

Classification tree:
rpart(formula = NObeyesdad ~ ., data = train_data_log, method = "class")

Variables actually used in tree construction:
[1] Age          CAEC         FAVCyes      FCVC         GenderFemale Height
    Weight

Root node error: 1092/1351 = 0.80829

n=1351 (210 osservazioni eliminate a causa di valori mancanti)

          CP nsplit rel error  xerror     xstd
1  0.211538      0   1.00000 1.00000 0.013250
2  0.198718      1   0.78846 0.78938 0.016176
3  0.132784      2   0.58974 0.60440 0.016825
4  0.064103      3   0.45696 0.45971 0.016265
5  0.036630      4   0.39286 0.41575 0.015899
6  0.027015      7   0.28297 0.26832 0.013872
7  0.020147      9   0.22894 0.22070 0.012886
8  0.016484     11   0.18864 0.19963 0.012382
9  0.014652     12   0.17216 0.19231 0.012196
10 0.010989     13   0.15751 0.18223 0.011929
11 0.010531     14   0.14652 0.17491 0.011727
12 0.010073     16   0.12546 0.17125 0.011624
13 0.010000     17   0.11538 0.15476 0.011135
```

As we can see in the above piece of code, the Classification tree selects 7 features in order to construct the three, that is, these features will pop-up in the upper part of the tree and they will resemble the most important decisions splits. Also, Figure 20 reports a brief summary of the computed tree where can see that the Complexity Parameter[9] *CP* decreases significantly when the size of the tree is equal to 10, and then slowly approaches its minimum as the size of the tree reaches its maximum.

[9]The complexity parameter (cp) in rpart is the minimum improvement in the model needed at each node.

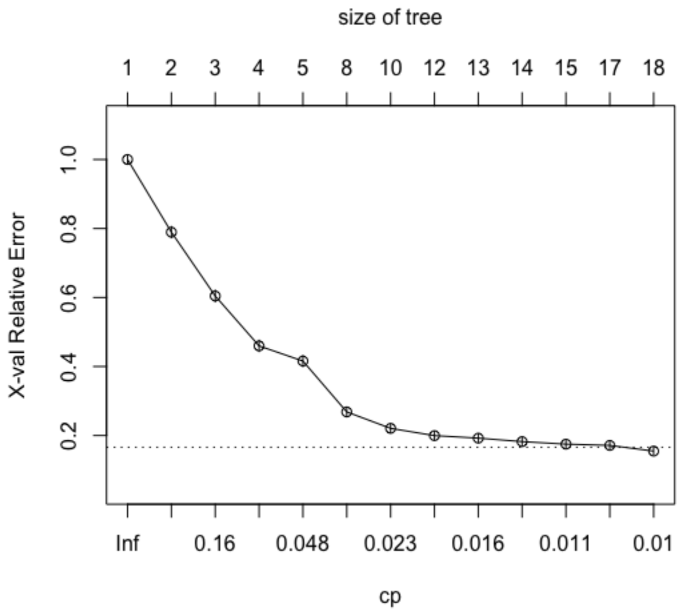


Figure 20: Complexitity parameter for the Classification Tree.

We then proceeded to prune the trained tree at a complexity level $cp = 0.020$, that is at the point in which increasing the complexity parameter does not lead any significant increase in the tree's performance.

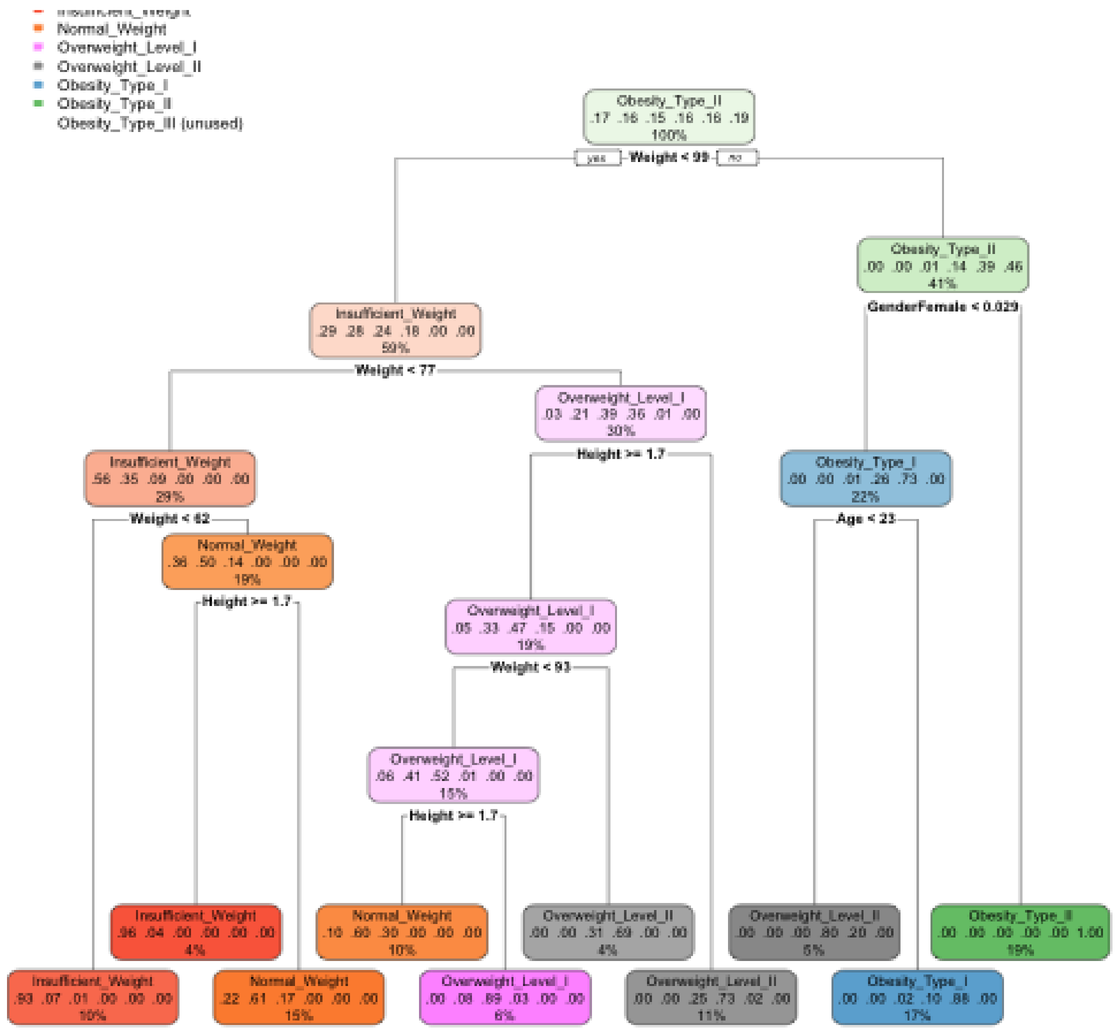

Normal_Weight
Overweight_Level_I
Overweight_Level_II
Obesity_Type_I
Obesity_Type_II
Obesity_Type_III (unused)
Obesity_Type_II .17 .16 .15 .16 .16 .19 100%
yes Weight < 99 no
Obesity_Type_II .00 .00 .01 .14 .39 .46 41%
GenderFemale < 0.029
Insufficient_Weight .29 .28 .24 .18 .00 .00 59%
Weight < 77
Overweight_Level_I .03 .21 .39 .36 .01 .00 30%
Height >= 1.7
Insufficient_Weight .56 .35 .09 .00 .00 .00 29%
Weight < 62
Obesity_Type_I .00 .00 .01 .26 .73 .00 22%
Age < 23
Normal_Weight .36 .50 .14 .00 .00 .00 19%
Height >= 1.7
Overweight_Level_I .05 .33 .47 .15 .00 .00 19%
Weight < 93
Overweight_Level_I .06 .41 .52 .01 .00 .00 15%
Height >= 1.7
Insufficient_Weight .96 .04 .00 .00 .00 .00 4%
Normal_Weight .10 .60 .30 .00 .00 .00 10%
Overweight_Level_II .00 .00 .31 .69 .00 .00 4%
Overweight_Level_II .00 .00 .00 .80 .20 .00 5%
Obesity_Type_II .00 .00 .00 .00 .00 1.00 19%
Insufficient_Weight .93 .07 .01 .00 .00 .00 10%
Normal_Weight .22 .61 .17 .00 .00 .00 15%
Overweight_Level_I .00 .08 .89 .03 .00 .00 6%
Overweight_Level_II .00 .00 .25 .73 .02 .00 11%
Obesity_Type_I .00 .00 .02 .10 .88 .00 17%

Figure 21: Pruned Classification Tree with $cp = 0.020$.

As we can see from Figure 21, with the selected level of Complexity we obtain a pruned tree with just 10 terminal nodes. Moreover, it is noticeable that the "Obesity_Type_III" response class is not even considered in the pruned tree. Also, the features that are used in the decision nodes drops from 7 to just 4, with Weight being the root of the tree. The most important features turns out to be Weight and Height, but this should be obvious since the BMI index relies on both of these features in order to be computed. Surprisingly, high level of BMI corresponding to Obesity Tipe II results to affect mostly males than females.

R code 1.3: RMSE, MAE and R^2.

```
> TREE = c(rmse(test_data$NObeyesdad, dt_1_pred),
+          mae(test_data$NObeyesdad, dt_1_pred),
+          R2(train_data$NObeyesdad, dt_1_yhat))
> tab_2 = table(test_data_log$NObeyesdad,dt_1_pred)
> TREE
[1] 0.5554127 0.2827763 0.9578041
> tab_2 = table(test_data_log$NObeyesdad,dt_1_pred)
> tab_accuracy = sum(diag(tab_2))/sum(tab_2)
> tab_accuracy
[1] 0.8658537
```

Finally, we can see from the above piece of code the Root Means Squared Error RMSE[10], the Mean Absolute Average MAE[11] and the R^2. I also computed the accuracy computing the MSE on the predicted observations on the test data with respect to the actual class value. The train Decision Tree performs quite well, with an accuracy level of approximately 87%.

1.5.2 Random Forest

Random forests algorithm is an ensemble learning method that operates by constructing a multitude of decision trees at training time. For regression tasks, the mean or average prediction of the individual trees is returned.
As always, the Random Forest is trained on the training data. At each split of tree it is possible to select a number of features equal to the square root of the total number of features, being $\sqrt{19} = 4$.

R code 1.4: Random Forest

```
> rf_1 = randomForest(NObeyesdad ~ .,data=train_data, mtry = sqrt(19))
> rf_1

Call:
 randomForest(formula = NObeyesdad ~ ., data = train_data, mtry = sqrt(19))
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 0.09918575
                    % Var explained: 97.6
```

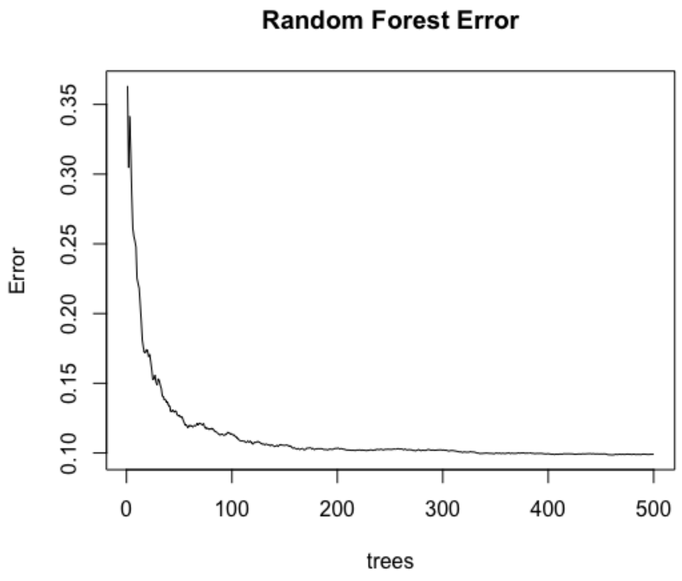


Figure 22: Random Forest Error with respect to the Number of Trees.

[10]RMSE is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE should be more useful when large errors are particularly undesirable.

[11]MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

We can see that the Number of trees turns out to be 500 but, as stated in Figure 22, it is more than sufficient to minimize the Random Forest error. In fact, with at least 100 trees generated by the Random Forest algorithm we have a significant decrease in the error.

As opposed to Decision Trees algorithm where the model is also supported by a graphical representation that allows to facilitate the readability and interpretability of the model, Random Forest does not provide such a feature. In order to comprehend which regressors are significant we can rely on Figure 23, which represents all the features included in the Random Forest and their relative importance, named *IncNodePurity*. Once again the most important feature by far is ”Weight”. It turns out that cases of obesity in the family may be significant in order to predict future obesity problem in the individual. This feature was not considered at all by the Decision Tree algorithm, but in all of the three Linear Regression models was considered to be statistically significant.

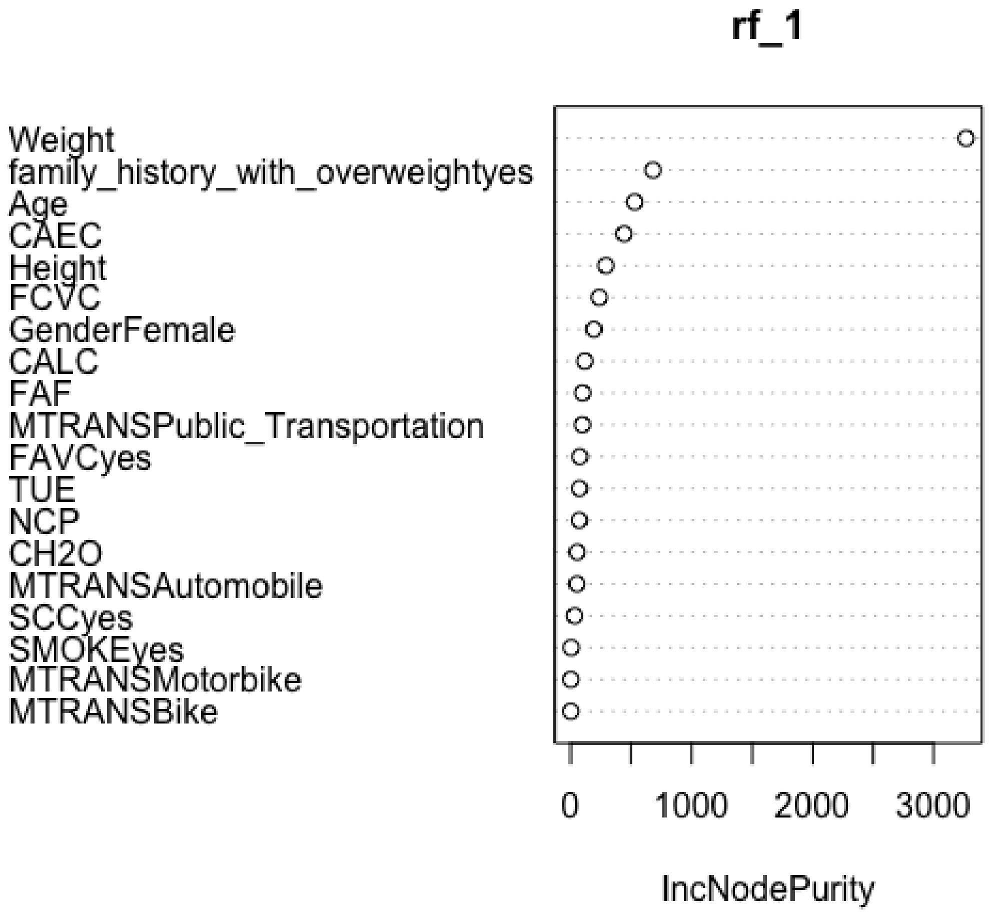


Figure 23: Random Forest incluede variables and relative importance.

Then, we applied the trained Random Forest algorithm to the test data in order to understand how it performes on new observations. Figure 24 displays the Out-Of-Bag Error, that is, the prediction error of the Random Forest, and also the true value of the response variable in the test set. As we can see, the model performs quite well.

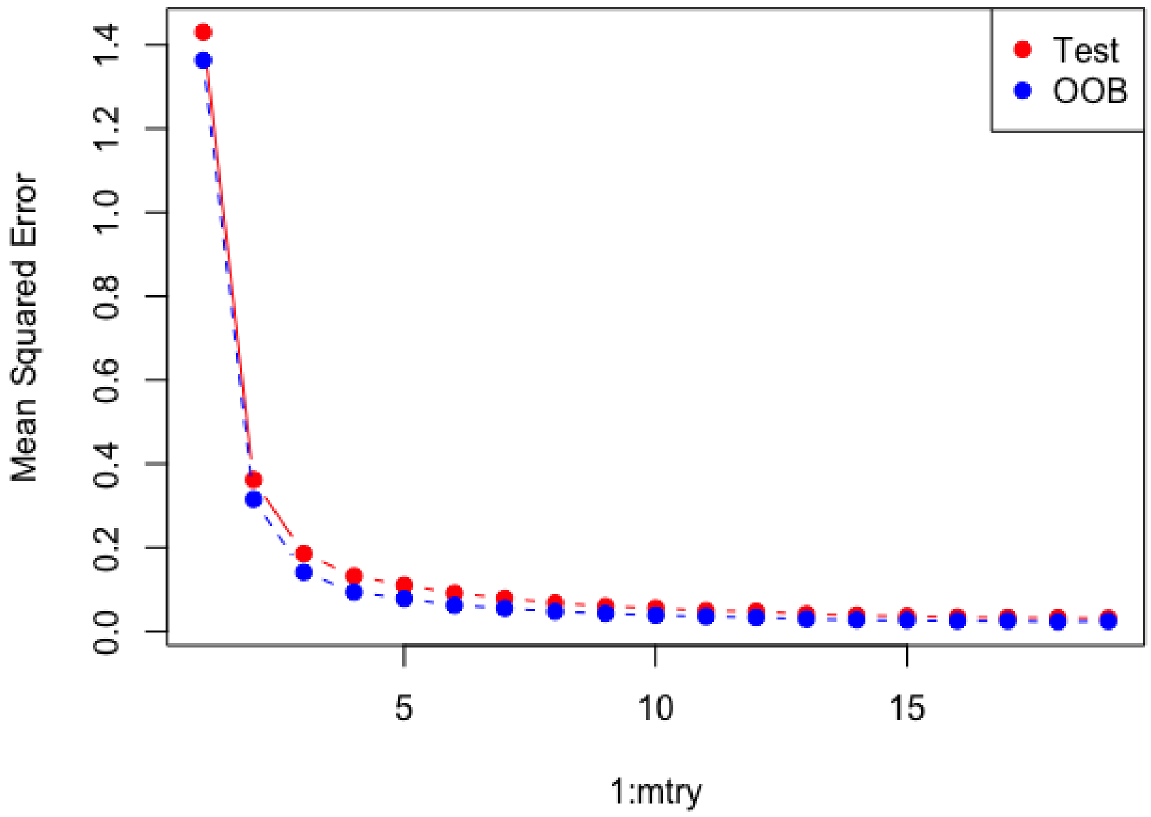


Figure 24: Random Forest OOB and Test Error.

1.6 Conclusions

In this first part of the report I implemented different Supervised Learning techniques in order to design different models that aims at predicting a possible Obesity problem in an individual. First of all, I implemented a simple Linear Regression model and, in order to improve the interpretability of the findings, I considered Stepwise and LASSO approaches to the Linear Regression model. The conclusions about these three models are reported in the previous sections.
Then, I considered a different class of methods, mainly known for capturing the non-linear relation between different features. The two selected models are Decision Tree and Random Forest. These two models allowed us to better capture any non-linear relationship between features. This resulted in various features that were considered by these class of methods and that did not appear to be significant in the linear model. Moreover, the main features selected to be the more significant across all the five models implemented turns out to be the same: Height, Weight and Age of the individuals plays a major role in predicting a possible Obesity problem. This should not be surprising since, as stated various time before, the Body Mass Index is directly computed from two of these three featurs. Also, Age plays an important role in predicting obesity level since it is a known fact that as an human gets old, it starts to lower his physical activities.
An interesting fact is that various features that resembles healths habits such as smoking, or movement transportations means such as preferring to walk instead of driving, did not played a major role in our models. Also different eating habits, such as consuming vegeatables during meals or measuring the consumed calories, played a little if not any role.
Considering the performance of the implemented models, all the five models had an overall high explanatory power and did performe pretty well in predicting new observations. Moreover, Random Forest turns out to

be the best model by all of the three performance parameters that I reported. It has by far the lowest RMSE and MAE, and the R^2 is little higher that the one reported for the Linear Regression models. Overall, in terms of accuracy power on unobserved observations, Decision Tree performed worst that all of the other models, striking a 55.5% RMSE.

R code 1.5: Model Comparisons

```
> metrics
           OLS  Stepwise     LASSO    Robust      TREE        RF
RMSE 0.4620516 0.4621797 0.4616688 0.4678218 0.5554127 0.3673668
MAE  0.3474481 0.3481118 0.3472340 0.3431884 0.2827763 0.2174774
R2   0.9558427 0.9557519 0.9558357 0.9555305 0.9578041 0.9689971
```

2 Part 2: Unsupervised Learning

Unsupervised Learning represents a variety of different algorithm that aims at learning patterns from the unlabeled data in order to obtain meaningful insights.
Considering our case, the response variable NObeyesdad resulted to be extremely useful when we applied Supervised Learning techniques to our data. Now, for this second part of the report I will try to perform Unsupervised Learning technique in order to cluster the data without relying on the information provided by the NObeyesdad feature which, as stated in the previous part, is directly computed from an individual's Height and Weight measurement.
The Unsupervised Learning algorithms that I will use in this second part are:

- MCA: Multiple Correspondence Analysis;
- Hierarchical Clustering.

Before starting with the model construction and with the analysis, I decided to reload the original data without any type of transformation. Then, I subtituted some of the numeric feature values with the corresponding meaning, that is, I transformed the majority of the features in qualitative ones[12]. The only features that I did not change are: Height, Age, Weight, and those who had a nominal binary response (i.e. "yes" or "no").

2.1 Multiple Correspondence Analysis

Multiple Correspondence Analysis is a data analysis technique for nominal categorical data used to detect and represent underlying structures in a data set. It can be seen as the counter part of the Principal Component Analysis, a widely known technique used in data analysis to reduce the dimensionality of the data. Moreover, this latter approach is used mainly on continous variables.
Multiple Correspondance Analysis is a variation of the Principal Component Analysis that could be applied to nominal qualitative features, such as in our case. In order to implement this model I used the *FactoMineR*[13] package in Rstudio.
In this part I decided to use as *active variables*, that is the ones used in the MCA, the one related to eating behavior, while the remaining ones are assigned to be *supplementary information*, that is variables that will not be used directly in the model, but for which coordinates will be predicted. Therefore, the active variables are: FAVC, FCVC, NCP, CAEC, CH2O, SCC, and CALC.

[12]Technically Multiple Correspondence Analysis is obtained by using a standard correspon- dence analysis on an indicator matrix (i.e., a matrix whose entries are 0 or 1). It is used to analyze a set of observations described by a set of nominal variables. Each nominal variable comprises several levels, and each of these levels is coded as a binary variable. For example gender, (F vs. M) is one nominal variable with two levels. The pattern for a male respondent will be 0 1 and 1 0 for a female. The complete data table is composed of binary columns with one and only one column taking the value "1" per nominal variable. See https://personal.utdallas.edu/ herve/Abdi-MCA2007-pretty.pdf for more details.

[13]https://cran.r-project.org/web/packages/FactoMineR/FactoMineR.pdf.

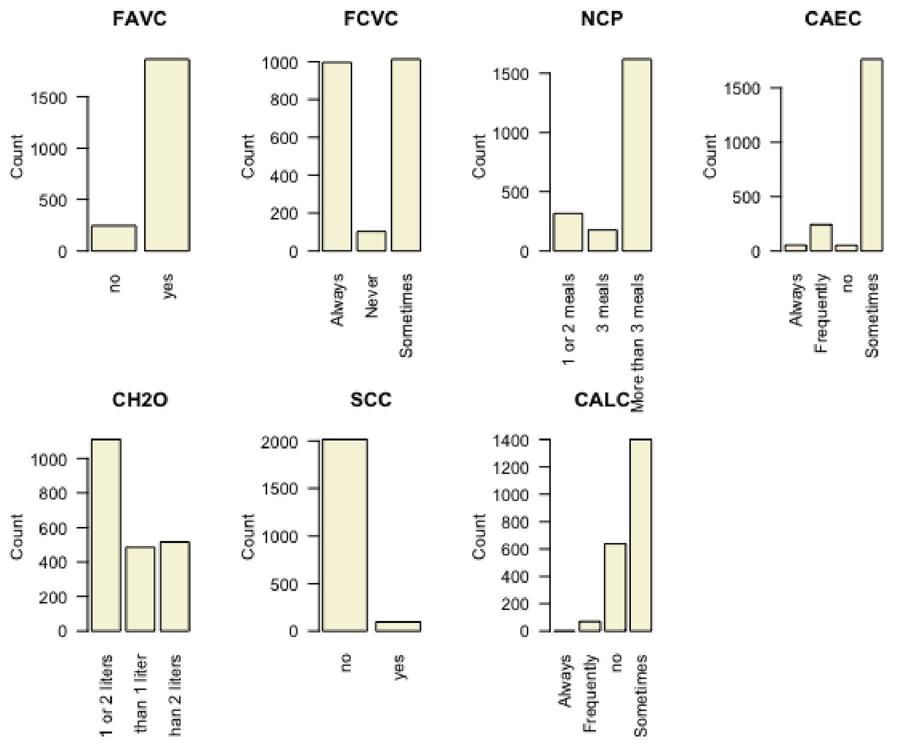


Figure 25: Frequency for the *Active Variables*.

I proceeded to run the model using the *MCA* function from the *FactoMineR* package which automatically detects the number of categorical variables (7 in our case). Since MCA (but also PCA) are known for being very sensitive to large outliers, I used the *outliers* function from the *FactoInvestigate*[14] package to detect them, that is, detect singular individuals that concentrates too much variance, properly called inertia. It resulted that 101 individuals are considered to be outliers[15], therefore I proceeded to remove them from the data.

Figure 26 (left) displays the percentage of explained variance in each dimension in the model, represented by the correspondent eigenvalue. Clearly, the first two dimensions are capable to explain approximately the 22% of the complessive variance in the data, measured by the percentage of variance explained in the model. In order to obtain a "good" Multiple Correspondence Analysis, we must consider using at least more than this two dimensions, since the majority of the variance is still unexplained. Moreover, considering at least ten dimensions will lead to explain the 87% of the variance in our data. This apparently bad performance can be explained by the fact that our original data have a small number of feature (17 in the beginning), and before applying the MCA algorithm we selected a subset with 7 active variables. Starting with these low number of features it is quite difficult to achieve a good performance in terms of dimension reduction without reducing the explanation of the variance.

R code 2.1: MCA summary, eigenvalues

```
> summary(res.mca)

Call:
MCA(X = data_active, ncp = 3, graph = TRUE)


```

[14]https://cran.r-project.org/web/packages/FactoInvestigate/FactoInvestigate.pdf.

[15]The aforementione function plots also the main features for which each individual results to concentrate inertia. The obtained results are not shown in the report but the function call is available in the Appendix code.

```
Eigenvalues
                       Dim.1   Dim.2   Dim.3   Dim.4   Dim.5   Dim.6   Dim.7
Variance               0.197   0.171   0.164   0.159   0.147   0.144   0.138
% of var.             11.512   9.966   9.554   9.252   8.574   8.410   8.032
Cumulative % of var.  11.512  21.479  31.033  40.285  48.859  57.269  65.301
                       Dim.8   Dim.9  Dim.10  Dim.11  Dim.12
Variance               0.132   0.124   0.120   0.118   0.101
% of var.              7.706   7.241   6.984   6.879   5.889
Cumulative % of var.  73.007  80.248  87.232  94.111 100.000
```

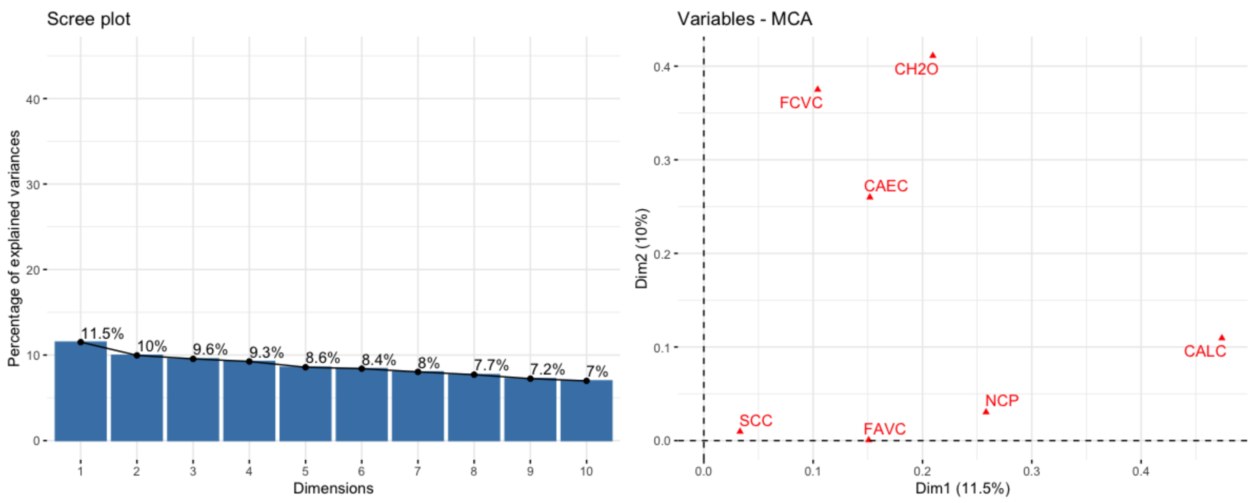


Figure 26: *Left*: Eigenvalues-Variance for MSE in each dimension. *Right*: Correlation between variables and principal dimensions.

On the other hand, Figure 26 (right) shows the correlation between the active variables and the first two dimensions. As we can see, the FCVC, CH2O and CAEC seems to be more correlated with the second dimension, while CALC and NCP with the first one. We can consider that good eating habits are better captured by the second dimension, while other aspects such as alcohol consumption or the number of meals per day are more concentrated in the first dimension. FAVC and also SCC seems to be correlated with both of the dimensions, that is, it may that higher dimension capture them the best way.

Figure 27 reports the quality of representation of variable categories, computed from the *squared cosine* (cos2), which measures the degree of association between variable categories and a particular axis (dimension). Below the figure there is a detail about each variable category with the related dimension. FAVC impacts only the first dimension and this explains the fact that is closest to the x-axis; also we can see that it reports 0 in the second dimension and 0.08, confirming the previous statement. It is noticeable that daily alcohol consumption seems to not influence the first three dimensions, while no alcohol or sporadic consumption shows their effect on the first dimension.

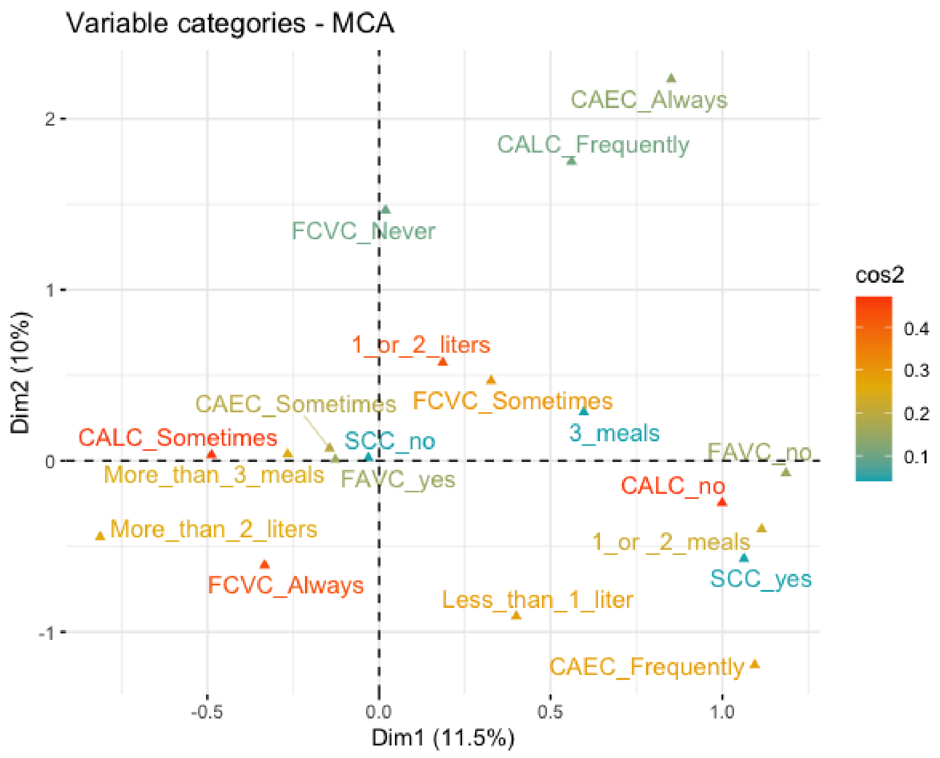


Figure 27: Quality of representation of variable categories.

R code 2.2: Model Comparisons

```
> round(var$cos2,2)
                     Dim 1 Dim 2 Dim 3
FAVC_no               0.15  0.00  0.08
FAVC_yes              0.15  0.00  0.08
FCVC_Always           0.10  0.34  0.06
FCVC_Never            0.00  0.10  0.03
FCVC_Sometimes        0.10  0.20  0.10
1_or _2_meals         0.20  0.03  0.00
3_meals               0.03  0.01  0.34
More_than_3_meals     0.24  0.01  0.15
CAEC_Always           0.02  0.12  0.25
CAEC_Frequently       0.13  0.15  0.18
CAEC_Sometimes        0.15  0.04  0.39
1_or_2_liters         0.04  0.39  0.00
Less_than_1_liter     0.05  0.24  0.00
More_than_2_liters    0.20  0.06  0.00
SCC_no                0.03  0.01  0.01
SCC_yes               0.03  0.01  0.01
CALC_Frequently       0.01  0.09  0.13
CALC_no               0.44  0.03  0.04
CALC_Sometimes        0.47  0.00  0.00
```

Looking at the individuals, we can see in Figure 28 the contribution, once again measured using sqaured cosine. The plot displays the individuals contribution, grouping the observation on the variable FAVC. As we could imagine, the observations are distributed along the x-axis . Moreover, people who tend to not consume frequently high caloric meals are positively correlated with the first dimension.

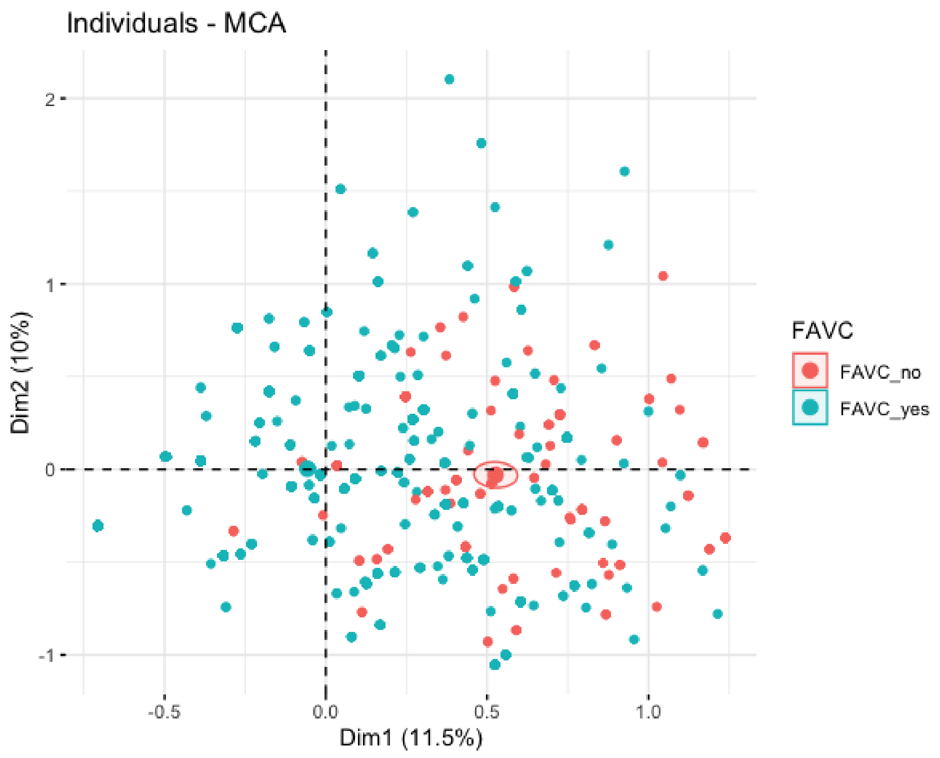


Figure 28: Quality of representation of individuals.

2.2 Hierarchical Clustering on Principal Components

Hierarchical Clustering is an algorithm that groups similar objects into groups called clusters. In particular clusters are considered to have different aspects one from the other, and each object contained in a certain cluster have more common things across other objects in the same cluster than objects contained in different clusters.
By treating each observation as a single cluster, the algorithm tries to find the two clusters that are closests together, in order to merge them in a single cluster. This operation is iterated until no merge is possible between two clusters.

In order to perform Hierarchical Clusering on Principal Components, we will continue the previous work obtained applying Multiple Correspondence Analysis.

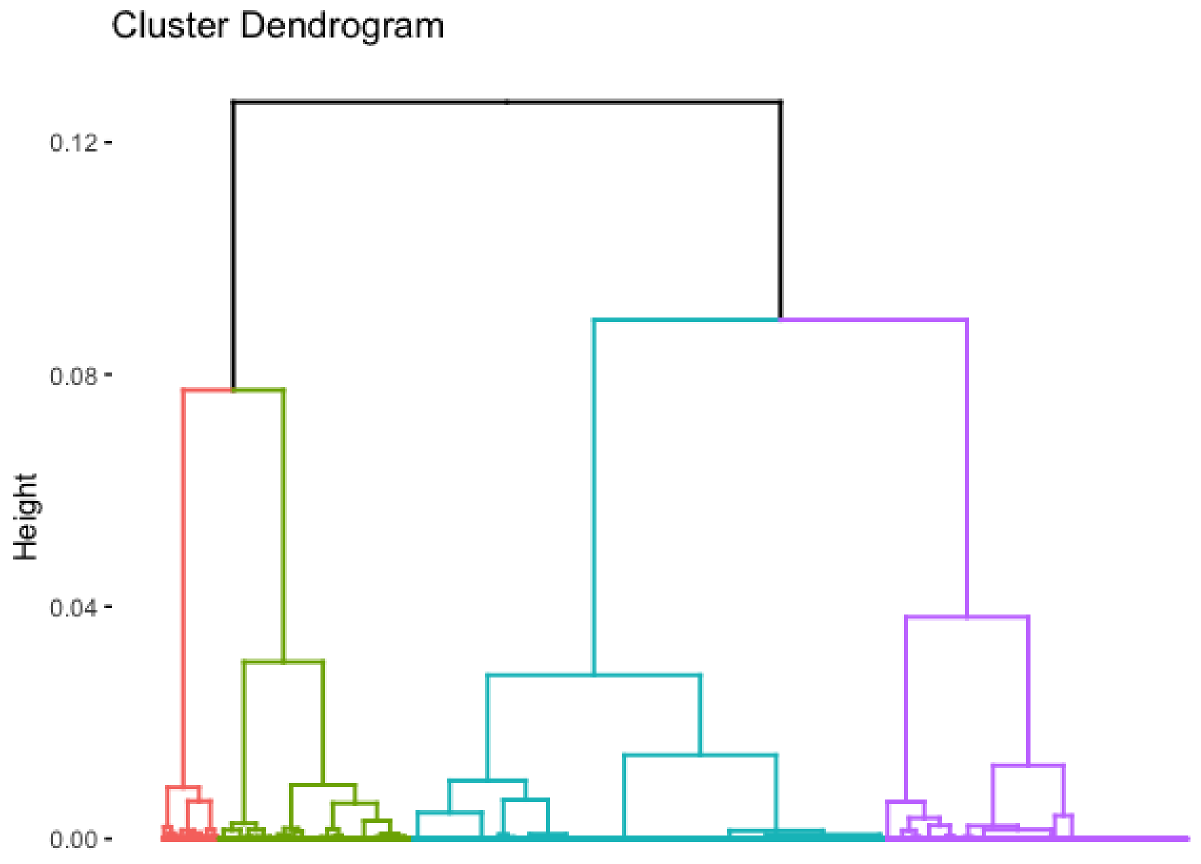


Figure 29: Dendogram obtained after Hierarchical Clustering.

Figure 29 displays the coloured dendogram obtained from the Multiple Correspondence Analysis. Compared to other algorithms, such as K- means, Hierarchical Clustering does not require to specify the cluster number. The algorithm detects 4 different clusters, each represented with a different colour. At Height being around 0.10, we can see that two main clusters are detected. Figure 30 displays the factor map for each considered invidual.

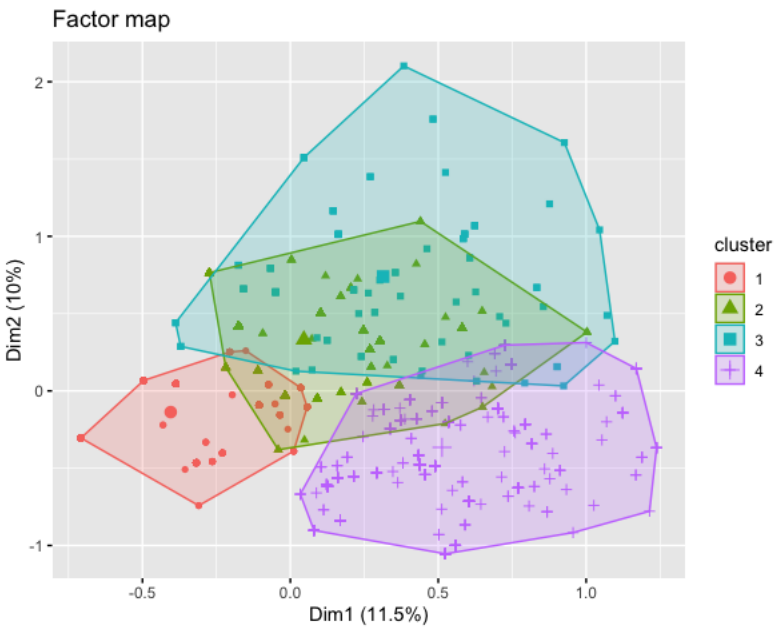


Figure 30: Factor map for the Hierarchical Clustering.

As we can see from the chunk of code below, clusters can be well described by the variables and/or categories, by the principal axes and by individuals. The first cluster appear to be composed by individuals who consume more than 3 meals on the daily basis, some of them being high-caloric meals, and whom consume alcohol in some occasions. On the other hand, the second clusters groups individuals who does not drink alcohol and who prefer to consume one or two meals per day. This consideration are assumed on the basis of the relevancy for each category in the cluster. In fact, we can see that in the first cluster are grouped individuals who drinks more than two liters per day of water, but also those that consume's less than one liter per day. In other words, the obtained clusters overlaps because different individuals with different habits turns out to be included in the same cluster. This happens because largest source of variation is similar among different groups.
These common variation seems to be concentrated in the second cluster, that is the one who overlaps with all the other three clusters. In fact, if we imagine for a bit to exclude it from Figure 30, we can clearly see that the three clusters are quite definite and not overlapping, exception made for a bunch of individuals that are considered to be in the fourth and in the third cluster.
Lastly, in the below chunk of code we can see some details about some individuals in each cluster. In particular we can look at the individuals who results to be closest to the centre of each cluster (indicated by *para*) and those who are more distant from the centre (via *dist*), with the relative distance.

R code 2.3: HC

```
> #Description of the 10 most relevant variables
> head(res.HCPC$desc.var$test.chi2,10)
           p.value df
CALC  0.000000e+00  6
CAEC 2.380674e-307  6
FCVC 5.479560e-135  6
CH2O 9.911078e-132  6
NCP  9.060267e-119  6
FAVC  1.318048e-40  3
SCC   3.167638e-12  3

> #Description of the 10 most relevant categories


> head(res.HCPC$desc.var$category,10)
```

```
$`1`
                         Cla/Mod    Mod/Cla    Global        p.value     v.test
CALC=CALC_Sometimes     57.379518 98.8326848 66.333666 8.403490e-167  27.526743
FCVC=FCVC_Always        61.684211 76.0051881 47.452547  2.300822e-94  20.608537
NCP=More_than_3_meals   47.840103 96.2386511 77.472527  3.149713e-68  17.455074
CH2O=More_than_2_liters 67.446809 41.1154345 23.476523  3.127435e-48  14.592664
CAEC=CAEC_Sometimes     43.238636 98.7029831 87.912088  3.660482e-40  13.265688
FAVC=FAVC_yes           40.818584 95.7198444 90.309690  1.062135e-11   6.797821
SCC=SCC_no              39.125964 98.7029831 97.152847  6.006064e-04   3.431340
SCC=SCC_yes             17.543860  1.2970169  2.847153  6.006064e-04  -3.431340
FCVC=FCVC_Never         18.390805  2.0752270  4.345654  4.031692e-05  -4.105656
CAEC=CAEC_Always         0.000000  0.0000000  2.347652  8.382741e-11  -6.493567
FAVC=FAVC_no            17.010309  4.2801556  9.690310  1.062135e-11  -6.797821
CALC=CALC_Frequently     0.000000  0.0000000  2.947053  1.996402e-13  -7.349037
CAEC=CAEC_Frequently     5.128205  1.2970169  9.740260  9.404716e-30 -11.329212
NCP=1_or _2_meals        9.352518  3.3722438 13.886114  1.289401e-31 -11.699016
NCP=3_meals              1.734104  0.3891051  8.641359  6.393022e-34 -12.141139
CH2O=1_or_2_liters      25.208526 35.2788586 53.896104  2.885254e-40 -13.283515
FCVC=FCVC_Sometimes     17.512953 21.9195850 48.201798  1.062408e-80 -19.024825
CALC=CALC_no             1.463415  1.1673152 30.719281 4.742403e-146 -25.735024

$`2`
                         Cla/Mod    Mod/Cla    Global        p.value     v.test
FCVC=FCVC_Sometimes     53.886010  80.996885 48.201798  4.854305e-95  20.683726
CH2O=1_or_2_liters      51.065802  85.825545 53.896104  1.345431e-93  20.522866
CAEC=CAEC_Sometimes     36.477273 100.000000 87.912088  1.098423e-44  14.024851
NCP=3_meals             76.878613  20.716511  8.641359  7.107568e-37  12.685591
FAVC=FAVC_yes           34.292035  96.573209 90.309690  2.185631e-12   7.022097
FCVC=FCVC_Never         63.218391   8.566978  4.345654  1.288682e-09   6.068802
CALC=CALC_no            38.699187  37.071651 30.719281  2.735689e-05   4.194422
SCC=SCC_no              32.647815  98.909657 97.152847  5.568374e-04   3.451811
CALC=CALC_Sometimes     29.894578  61.838006 66.333666  3.637718e-03  -2.907980
NCP=More_than_3_meals   30.109607  72.741433 77.472527  5.808125e-04  -3.440420
SCC=SCC_yes             12.280702   1.090343  2.847153  5.568374e-04  -3.451811
CALC=CALC_Frequently    11.864407   1.090343  2.947053  3.190934e-04  -3.599285
CAEC=CAEC_Always         0.000000   0.000000  2.347652  9.877629e-09  -5.732817
NCP=1_or _2_meals       15.107914   6.542056 13.886114  5.758978e-12  -6.885488
FAVC=FAVC_no            11.340206   3.426791  9.690310  2.185631e-12  -7.022097
CH2O=More_than_2_liters 15.531915  11.370717 23.476523  4.098730e-20  -9.185434
CAEC=CAEC_Frequently     0.000000   0.000000  9.740260  1.402396e-35 -12.449775
CH2O=Less_than_1_liter   3.973510   2.803738 22.627373  9.431830e-61 -16.442886
FCVC=FCVC_Always         7.052632  10.436137 47.452547 1.968473e-127 -24.014366

$`3`
                         Cla/Mod  Mod/Cla    Global      p.value     v.test
CAEC=CAEC_Always       100.000000 47.95918  2.347652 7.099873e-68  17.408600
CALC=CALC_Frequently    88.135593 53.06122  2.947053 2.045747e-67  17.347904
FCVC=FCVC_Never         13.793103 12.24490  4.345654 1.005280e-03   3.289045
FAVC=FAVC_no             9.278351 18.36735  9.690310 6.754817e-03   2.708696
CH2O=1_or_2_liters       6.024096 66.32653 53.896104 1.103054e-02   2.541730
CH2O=Less_than_1_liter   3.090508 14.28571 22.627373 3.714290e-02  -2.084190
FAVC=FAVC_yes            4.424779 81.63265 90.309690 6.754817e-03  -2.708696
CALC=CALC_no             2.926829 18.36735 30.719281 4.986010e-03  -2.807936
NCP=3_meals              0.000000  0.00000  8.641359 1.128468e-04  -3.861167
CALC=CALC_Sometimes      2.108434 28.57143 66.333666 6.227266e-15  -7.799257
CAEC=CAEC_Sometimes      2.215909 39.79592 87.912088 5.891423e-32 -11.765303

$`4`
                         Cla/Mod   Mod/Cla    Global        p.value     v.test
CALC=CALC_no            56.910569 71.283096 30.719281 1.137925e-105  21.832588
CAEC=CAEC_Frequently    88.717949 35.234216  9.740260  4.613437e-91  20.237121
NCP=1_or _2_meals       69.064748 39.103870 13.886114  4.011970e-66  17.176068
```

```
CH2O=Less_than_1_liter  52.759382 48.676171 22.627373  2.122971e-51  15.082133
FAVC=FAVC_no            62.371134 24.643585  9.690310  2.140888e-32  11.850430
SCC=SCC_yes             66.666667  7.739308  2.847153  1.194836e-11   6.780835
FCVC=FCVC_Always        27.263158 52.749491 47.452547  6.917136e-03   2.700807
CAEC=CAEC_Always         0.000000  0.000000  2.347652  1.510138e-06  -4.809889
FCVC=FCVC_Never          4.597701  0.814664  4.345654  5.037817e-07  -5.024867
CALC=CALC_Frequently     0.000000  0.000000  2.947053  4.643107e-08  -5.464463
SCC=SCC_no              23.290488 92.260692 97.152847  1.194836e-11  -6.780835
CH2O=More_than_2_liters 12.978723 12.423625 23.476523  3.202307e-12  -6.968547
CH2O=1_or_2_liters      17.701576 38.900204 53.896104  1.754237e-14  -7.667469
FAVC=FAVC_yes           20.464602 75.356415 90.309690  2.140888e-32 -11.850430
NCP=More_than_3_meals   16.892328 53.360489 77.472527  1.546803e-44 -14.000545
CAEC=CAEC_Sometimes     18.068182 64.765784 87.912088  3.076376e-62 -16.649012
CALC=CALC_Sometimes     10.617470 28.716904 66.333666  2.219414e-88 -19.930364

> res.HCPC$desc.ind$para
Cluster: 1
       23       111       152       196       228
0.1875102 0.1875102 0.1875102 0.1875102 0.1875102
-------------------------------------------------------------
Cluster: 2
      106       107       151       169       171
0.2381431 0.2381431 0.2381431 0.2381431 0.2381431
-------------------------------------------------------------
Cluster: 3
       391        108        202        286        459
0.04851977 0.13238048 0.13238048 0.13238048 0.13238048
-------------------------------------------------------------
Cluster: 4
       22        53        59       124       251
0.2375862 0.2375862 0.2375862 0.2375862 0.2375862

> #Individuals farest from the centre
> res.HCPC$desc.ind$dist
Cluster: 1
     141       11       51      216      231
1.078936 1.078936 1.052467 1.052467 1.052467
-------------------------------------------------------------
Cluster: 2
     746      793      894      981     1012
1.305863 1.305863 1.305863 1.305863 1.305863
-------------------------------------------------------------
Cluster: 3
      67       68      126      415      134
3.031465 2.535664 2.535664 2.512287 2.505512
-------------------------------------------------------------
Cluster: 4
     199      385      259      328      577
1.671027 1.545289 1.519273 1.480808 1.480808
```

2.3 Conclusion

As we have seen, implementing a particular type of Principal Component Analysis helped us understanding the relation between our features, although I was not able to reduce the dimensionality of the data. Various algorithms could have been used, such as Factor Analysis for Mixed Data, or also a classic Principal Component. The former would have been useful in order to detect relations in the data assuming the presence of a latent factor. The latter require to factorize and scale all the data in order to obtain significant results. I ended up prefering Multiple Correspondence Analysis since our data was collected from a survey, a case in which this type of algorithm were purposely implemented.
It is possible to implement the results obtained with the MCA by applying some different type of algorithms,

such as ones that could capture non-linearity or complex relationship between features. Moreover, we are satisfied with the achieved results.

A Code

```
#### Introduction ####
#Libraries and packages required
require(readxl)
require(sandwich)
require(lmtest)
require(expss)
require(MASS)
require(corrplot)
require(dplyr)
require(janitor)
require(data.table)
require(mltools)
require(plyr)
require(caret)
require(tidyr)
require(car)
require(rpart)
require(rpart.plot)
require(tree)
require(partykit)
require(randomForest)
require(nnet)
require(gbm)
require(ggmosaic)
require(grid)
require(gridExtra)
require(parameters)
require(performance)
require(glmnet)
require(Metrics)
require(FactoMineR)
require(factoextra)
require(RColorBrewer)

#Working directory
setwd("~/Desktop/FantaCalcio_Project/Obesity")

#### Data import and cleaning ####
#Data import
data = read.csv(file = "~/Desktop/FantaCalcio_Project/Obesity/data/data.csv")

# 1) Formatting
colnames(data)
data$Age = round(data$Age, 0)
data$Height = round(data$Height, 2)
data$Weight = round(data$Weight, 2)
data$FCVC = round(data$FCVC, 0)
data$NCP = round(data$NCP, 0)
data = data %>%
  mutate(NPC=ifelse(NCP==4,3,NCP))
data$NCP = data$NPC
data = data[, -18]
data$CH2O = round(data$CH2O, 0)
data$FAF = round(data$FAF, 0)
data$TUE = round(data$TUE,0)

describe_distribution(data)

#2) Subset DF into categorical variables and numerical variables
```

```
data_num = data[ , c("Age", "Height", "Weight")]
data_cha = data[ , c("Gender", "family_history_with_overweight", "FAVC", "FCVC",
    "NCP", "CAEC", "SMOKE", "CH2O", "SCC", "FAF", "TUE", "CALC", "MTRANS",
    "NObeyesdad")]

#3) Distribution of all numerical variables, represented via Histograms
colnames(data_num)
hist(data_num$Age,  main = "Histogram␣of␣Age", xlab = "Age", col = "lightblue")
hist(data_num$Height,  main = "Histogram␣of␣Height", xlab = "Height", col =
    "lightblue")
hist(data_num$Weight,  main = "Histogram␣of␣Weight", xlab = "Weight", col =
    "lightblue")

#4) Correlation matrix and plot on numerical variables
cor_n = cor(data_num)
corrplot(cor_n, type = "upper", tl.col = "black", tl.srt = 45, title =
    "Correlation␣plot␣between␣numerical␣variables")

#5) Compare 'NObeyesdad' levels across 'Age', 'Height', 'Weight'
ddply(data, "NObeyesdad", summarize,
      Age = round(mean(Age),2),
      Height = round(mean(Height),2),
      Weight = round(mean(Weight),2))

#6) Distribution of some variables
#ggplot(data, aes(x = Gender, fill = Gender)) +
#  geom_bar(stat = "count") +
#  ggtitle("Distribution for Gender variable") +
#  xlab("Gender") + ylab("Number of records") +
# theme_bw() +
# geom_text(aes(label = ..count..), stat = 'count', vjust = -0.4) +
# labs(fill = "Obesity level") +
# theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

#ggplot(data, aes(x = SMOKE, fill = SMOKE)) +
# geom_bar(stat = "count") +
# ggtitle("Distribution for Smoking habits response") +
# xlab("Smokers") + ylab("Number of records") +
# theme_bw() +
# geom_text(aes(label = ..count..), stat = 'count', vjust = -0.4) +
# labs(fill = "Obesity level") +
# theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

#ggplot(data, aes(x = FAF, fill = FAF)) +
# geom_bar(stat = "count") +
# ggtitle("Distribution for Physical activity frequency  response") +
# xlab("Response") + ylab("Number of records") +
# theme_bw() +
# geom_text(aes(label = ..count..), stat = 'count', vjust = -0.4) +
# labs(fill = "Obesity level") +
# theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

#ggplot(data, aes(x = TUE, fill = TUE)) +
# geom_bar(stat = "count") +
# ggtitle("Distribution for Time using technology devices response") +
# xlab("Response") + ylab("Number of records") +
# theme_bw() +
# geom_text(aes(label = ..count..), stat = 'count', vjust = -0.4) +
# labs(fill = "Obesity level") +
# theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

#ggplot(data, aes(x = NObeyesdad, fill = NObeyesdad)) +
# geom_bar(stat = "count") +
# ggtitle("Distribution across different obesity levels") +
# xlab("Obesity level") + ylab("Number of records") +
# theme_bw() +
# geom_text(aes(label = ..count..), stat = 'count', vjust = -0.4) +
# labs(fill = "Obesity level") +
# theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))

# 6) Distribution of some variables; Gender wise distribution for Weight and Height
#ggplot(data, aes(y = Height, color = Gender)) + #fig3
# geom_boxplot() +
# ggtitle("Gender wise distribution of Height") +
# theme_bw()

#ggplot(data, aes(y = Weight, color = Gender)) + #fig4
# geom_boxplot() +
# ggtitle("Gender wise distribution of Weight") +
# theme_bw()

# 6) Distribution of some variables: Analyzing the preferred Mode of Transport by
   Gender
#ggplot(data) +
# geom_bar(aes(y = MTRANS, fill = Gender),position = position_dodge()) +
# ylab("Mode of Transport") +
# scale_color_manual(values = c("blue", "red"), aesthetics = "fill") +
# ggtitle("Transport preferences by Gender") +
# theme_mosaic()

# 7) Comparing Compare 'NObeyesdad' levels across categorical variables
tabyl(data, NObeyesdad, Gender)
tabyl(data, NObeyesdad, family_history_with_overweight)
tabyl(data, NObeyesdad, FAVC)
tabyl(data, NObeyesdad, FCVC)
tabyl(data, NObeyesdad, NCP)
tabyl(data, NObeyesdad, CAEC)
tabyl(data, NObeyesdad, SMOKE)
tabyl(data, NObeyesdad, CH2O)
tabyl(data, NObeyesdad, SCC)
tabyl(data, NObeyesdad, MTRANS)

# 8) Outliers detection
#fig_bp1 = ggplot(data, aes(y = Age, color = Age)) + #fig3
# geom_boxplot() +
# ggtitle("Age") +
# theme_bw()
#fig_bp2 = ggplot(data, aes(y = Height, color = Height)) + #fig3
# geom_boxplot() +
# ggtitle("Height") +
# theme_bw()
#fig_bp3 = ggplot(data, aes(y = Weight, color = Weight)) + #fig3
# geom_boxplot() +
# ggtitle("Weight") +
# theme_bw()
#grid.arrange(fig_bp1, fig_bp2, fig_bp3, ncol = 3)

# 9) Check outliers with Interquartile Range: Percentile: Age column
data_out = data[, c("Age", "Height", "Weight")]
out_summary = as.data.frame(apply(data_out,2,summary))

age_p25 = out_summary[2,1]
age_p75 = out_summary[5,1]
```

```
# 9) Check outliers with Interquartile Range: Percentile: Weight column
wei_p25 = out_summary[2,3]
wei_p75 = out_summary[5,3]

# 9) Check outliers with Interquartile Range: Inte-rquantile Ranges
IR_age = age_p75 - age_p25
IR_wei = wei_p75 - wei_p25

# 9) Check outliers with Interquartile Range: Determine the lower and upper limit
lower_lim_age = age_p25 - (1.5 * IR_age)
upper_lim_age = age_p75 + (1.5 * IR_age)

lower_lim_weight = wei_p25 - (1.5 * IR_wei)
upper_lim_weight = wei_p75 + (1.5 * IR_wei)

# 9) Check outliers with Interquartile Range: Dropping observations which do not
    satisfy IQR outliers approach
data_iqr = data
data_iqr = data_iqr[!(data_iqr$Age > upper_lim_age | data_iqr$Age <
    lower_lim_age),]
data_iqr = data_iqr[!(data_iqr$Weight > upper_lim_weight | data_iqr$Weight <
    lower_lim_weight),] #same outliers for Weight and Height

# 9) Check outliers with Interquartile Range: boxplot without outliers
#fig_bp4 = ggplot(data_iqr, aes(y = Age, color = Age)) + #fig3
# geom_boxplot() +
# ggtitle("Age") +
# theme_bw()
#fig_bp5 = ggplot(data_iqr, aes(y = Height, color = Height)) + #fig3
# geom_boxplot() +
# ggtitle("Height") +
# theme_bw()
#fig_bp6 = ggplot(data_iqr, aes(y = Weight, color = Weight)) + #fig3
# geom_boxplot() +
# ggtitle("Weight") +
# theme_bw()
#grid.arrange(fig_bp4, fig_bp5, fig_bp6, ncol = 3)

#rm(data_cha, data_num, age_p25, age_p50, age_p75, hei_p25, hei_p50, hei_p75,
    wei_p25, wei_p50, wei_p75, IR_age, IR_wei, IR_hei,
#    lower_lim_age, lower_lim_height, lower_lim_weight, upper_lim_age,
    upper_lim_height, upper_lim_weight, outliers,
#   data_out, cor_n, out_summary, find_outliers_age, find_outliers_weight)
#rm(fig_bp1, fig_bp2, fig_bp3, fig_bp4, fig_bp5, fig_bp6)

#### Data preparation for regression\classification ####
# 1) Data Pre-Processing
sapply(data_iqr,class)

# 2) one-hot-encoding for binary categorical variables
data_cha_ohe = data_iqr[, c("Gender", "family_history_with_overweight", "FAVC",
    "SMOKE", "SCC", "MTRANS")]
dmy = dummyVars("␣~␣.", data = data_cha_ohe)
dat_transformed = data.frame(predict(dmy, newdata = data_cha_ohe))

data_iqr = cbind(data_iqr, dat_transformed)
data_iqr = data_iqr[, -c(1, 5, 6, 10, 12, 16 )]
rm(data_cha_ohe, dmy, dat_transformed)

# 3) Ordinal Encoding: used for ordinal categorical variables "CAEC", "CALC",
    "NObeyesdad"
```

```r
data_iqr$CAEC = factor(data_iqr$CAEC, levels = c("no", "Sometimes", "Frequently",
    "Always"), labels = c(1, 2, 3, 4))
data_iqr$CALC = factor(data_iqr$CALC, levels = c("no", "Sometimes", "Frequently",
    "Always"), labels = c(1, 2, 3, 4))
data_iqr$NObeyesdad = factor(data_iqr$NObeyesdad, levels =
    c("Insufficient_Weight", "Normal_Weight", "Overweight_Level_I",
    "Overweight_Level_II", "Obesity_Type_I", "Obesity_Type_II",
    "Obesity_Type_III"), labels = c(0:6))

col_order = c("GenderFemale", "GenderMale", "Age", "Height", "Weight",
    "family_history_with_overweightno", "family_history_with_overweightyes",
               "FAVCno", "FAVCyes", "FCVC", "NCP", "CAEC",
               "SMOKEno", "SMOKEyes", "CH2O", "SCCno", "SCCyes", "FAF", "TUE",
               "CALC", "MTRANSAutomobile", "MTRANSBike", "MTRANSMotorbike",
                  "MTRANSPublic_Transportation",
               "MTRANSWalking", "NObeyesdad")
data_iqr = data_iqr[, col_order]
describe_distribution(data_iqr)

dev.off()
boxplot(data_iqr)
title("Comparing␣boxplot()s")

# 4) Scaling variables
data_iqr$CAEC = as.numeric(levels(data_iqr$CAEC))[data_iqr$CAEC]
data_iqr$CALC = as.numeric(levels(data_iqr$CALC))[data_iqr$CALC]
data_iqr$NObeyesdad = as.numeric(levels(data_iqr$NObeyesdad))[data_iqr$NObeyesdad]
data_iqr_2 = scale(x = as.data.table(data_iqr), center = TRUE, scale = TRUE)

boxplot(data_iqr_2)
title("Comparing␣boxplot()s,␣scaled")
summary(data_iqr_2)
colnames(data_iqr)
data_iqr = data_iqr[,-c(2, 6, 8, 13, 16, 25)]

### 5) DF with only categorical variable scaled and numerical ones unchanged
scaled = as.data.frame(scale(x = as.data.table(data_iqr[,-c(2:4, 20)]), center =
    TRUE, scale = TRUE))

true = data_iqr[, c(2:4, 20)]
sc_data = cbind(scaled, true)

col_order = c("GenderFemale", "Age", "Height", "Weight",
    "family_history_with_overweightyes",
               "FAVCyes", "FCVC", "NCP", "CAEC",
               "SMOKEyes", "CH2O", "SCCyes", "FAF", "TUE",
               "CALC", "MTRANSAutomobile", "MTRANSBike", "MTRANSMotorbike",
                  "MTRANSPublic_Transportation",
               "NObeyesdad")
sc_data = sc_data[, col_order]
rm(true, scaled)

#### before OLS REGRESSION ####
# Training and Test splitting set
set.seed(123)
split_train_test = createDataPartition(y = sc_data$NObeyesdad, p=0.8, list = F)
train_data = sc_data[split_train_test ,]
test_data = sc_data[-split_train_test ,]
dim(test_data)
dim(train_data)
#### 0) Part 1: LINEAR REGRESSION ####
lin_mod = lm(formula = NObeyesdad ~ ., data = train_data)
```

```
lin_sum = summary(lin_mod)

par(mfrow = c(2,2))
plot(lin_mod)
dev.off()

check_model(lin_mod)
check_normality(lin_mod)
check_heteroscedasticity(lin_mod)
check_autocorrelation(lin_mod)
check_collinearity(lin_mod)

coeftest(lin_mod, vcov. = vcovHC, type = "HC1")
lin_mod_performance = performance(lin_mod)

# STEPWISE REGRESSION
lin_mod_step = lm(formula = NObeyesdad ~ ., data = train_data)
lin_mod_step = select_parameters(lin_mod_step)
lin_step_sum = summary(lin_mod_step)

coeftest(lin_mod_step ,vcov. = vcovHC , type = "HC1")

check_model(lin_mod_step)
check_normality(lin_mod_step)
check_heteroscedasticity(lin_mod_step)
check_autocorrelation(lin_mod_step)
check_collinearity(lin_mod_step)

lin_step_performance = performance(lin_mod_step)

# LASSO REGRESSION
x = as.matrix(train_data[,-20])
y = train_data$NObeyesdad
cv_lasso = cv.glmnet(x,y)
plot(cv_lasso)

coef = coef(cv_lasso , s = cv_lasso$lambda.min)
coefname = coef@Dimnames [[1]][ -1]
coef = coefname[coef@i]
coef

fmla = as.formula(paste("y␣~␣", paste(coef, collapse = "+")))

lin_lasso = lm(fmla, data=train_data)
lin_lasso_sum = summary(lin_lasso)
check_model(lin_lasso)

check_normality(lin_lasso)
check_heteroscedasticity(lin_lasso)
check_autocorrelation(lin_lasso)
check_collinearity(lin_lasso)

coeftest(lin_lasso, vcov. = vcovHC, type='HC1')

lin_lasso_performance = performance(lin_lasso)

# compare linear regressions
compare_performance(lin_mod,lin_lasso,lin_mod_step,rank = T)
plot(compare_performance(lin_mod,lin_lasso,lin_mod_step,rank = T))

# Robust lasso estimator
lin_mod_lasso_robust = rlm(fmla,data=train_data, psi = psi.bisquare)
```

```r
hweights = data.frame(resid = lin_mod_lasso_robust$resid, weight =
    lin_mod_lasso_robust$w)
hweights2 = hweights[order(lin_mod_lasso_robust$w),]
hweights2 [1:10 ,]

rob_se_pan = list(sqrt(diag(vcovHC(lin_mod , type = "HC1"))),
                  sqrt(diag(vcovHC(lin_mod_step , type = "HC1"))),
                  sqrt(diag(vcovHC(lin_lasso , type = "HC1"))),
                  sqrt(diag(vcovHC(lin_mod_lasso_robust , type = "HC1"))))

#### 1) Part 1: MULTINOMIAL LOGISTIC REGRESSION ####
train_data_log = train_data
test_data_log = test_data
train_data_log$NObeyesdad = factor(train_data_log$NObeyesdad, levels = c(1:7),
    labels = c("Insufficient_Weight", "Normal_Weight", "Overweight_Level_I",
    "Overweight_Level_II", "Obesity_Type_I", "Obesity_Type_II",
    "Obesity_Type_III"))
test_data_log$NObeyesdad = factor(test_data_log$NObeyesdad, levels = c(1:7),
    labels = c("Insufficient_Weight", "Normal_Weight", "Overweight_Level_I",
    "Overweight_Level_II", "Obesity_Type_I", "Obesity_Type_II",
    "Obesity_Type_III"))

# Setting the 'reference' level
train_data_log$NObeyesdad = relevel(train_data_log$NObeyesdad, ref =
    "Insufficient_Weight")
multinom_model = multinom(NObeyesdad ~ ., data = train_data_log) #training the
    model
summary(multinom_model)
exp(coef(multinom_model)) # Convert the coefficients to odds by taking the
    exponential of the coefficients.
head(round(fitted(multinom_model), 2))# The predicted values are saved as
    fitted.values in the model object.

# Predicting & Validating the model: Predicting the values for train dataset
train_data_log$ClassPredicted = predict(multinom_model, newdata = train_data_log,
    "class")
tab = table(train_data_log$NObeyesdad, train_data_log$ClassPredicted) # Building
    classification table
tab
round((sum(diag(tab))/sum(tab))*100,2)

# Predicting the class on test dataset.
test_data_log$ClassPredicted = predict(multinom_model, newdata = test_data_log,
    "class")
tab_1 = table(test_data_log$NObeyesdad, test_data_log$ClassPredicted)# Building
    classification table
tab_1
round((sum(diag(tab_1))/sum(tab_1))*100,2)

#### 2) Part 1: REGRESSION DECISION TREES####
train_data_log = train_data
test_data_log = test_data
train_data_log$NObeyesdad = factor(train_data_log$NObeyesdad, levels = c(1:7),
    labels = c("Insufficient_Weight", "Normal_Weight", "Overweight_Level_I",
    "Overweight_Level_II", "Obesity_Type_I", "Obesity_Type_II",
    "Obesity_Type_III"))
test_data_log$NObeyesdad = factor(test_data_log$NObeyesdad, levels = c(1:7),
    labels = c("Insufficient_Weight", "Normal_Weight", "Overweight_Level_I",
    "Overweight_Level_II", "Obesity_Type_I", "Obesity_Type_II",
    "Obesity_Type_III"))
```

```
tree_1 = rpart(NObeyesdad~., data = train_data_log, method = "class")
printcp(tree_1)
summary(tree_1)

cv_tree_1 = plotcp(tree_1) #size of the tree = 10
rpart.plot(tree_1)

prune_tree_1 = prune(tree_1,cp=0.023)
rpart.plot(prune_tree_1)

#prediction on train data
dt_1_yhat = as.numeric(predict(tree_1, newdata=train_data_log, type= "class"))
dt_1_train = as.numeric(train_data_log[,"NObeyesdad"]) #output vector in test data
plot(dt_1_yhat,dt_1_train);abline(0,1, col = "red") #how is the distribution
    between the final units in the nodes

# Prediction on test data
dt_1_pred = as.numeric(predict(tree_1, test_data, type= "class"))

# Confusion Matrix
confMat = table(test_data[, "NObeyesdad"],dt_1_pred)
dt_1_accuracy = sum(diag(confMat))/sum(confMat)

TREE = c(rmse(test_data$NObeyesdad, dt_1_pred),
         mae(test_data$NObeyesdad, dt_1_pred),
         R2(train_data$NObeyesdad, dt_1_yhat))

tab_2 = table(test_data_log$NObeyesdad,dt_1_pred)
tab_accuracy = sum(diag(tab_2))/sum(tab_2)

#### 3) Part 1: RANDOM FOREST REGRESSION ####
rf_1 = randomForest(NObeyesdad ~ .,data=train_data, mtry = sqrt(19), type =
    "regression")
plot(rf_1, main = "Random␣Forest␣Error")
rf_1_y_hat = as.numeric(predict(rf_1 ,newdata=test_data))
test_data$NObeyesdad = as.numeric(test_data$NObeyesdad)
mean((rf_1_y_hat - test_data$NObeyesdad)^2)

ggplot() +
  geom_point(aes(x = test_data$NObeyesdad, y = rf_1_y_hat)) +
  geom_abline()

impo = round(importance(rf_1),2)
varImpPlot (rf_1)

oob.err=double(19)
test.err=double(19)
for(mtry in 1:19){
  fit=randomForest(NObeyesdad~.,data=train_data,mtry=mtry,ntree=400)
  oob.err[mtry]=fit$mse[400]
  pred=predict(fit,test_data)
  test.err[mtry]=with(test_data,mean((NObeyesdad-pred)^2))
  cat(mtry,"␣")
}
oob.err
test.err
matplot(1:mtry,cbind(test.err,oob.err),pch=19,col=c("red","blue"),type="b",ylab="Mean␣
    Squared␣Error");
    legend("topright",legend=c("Test","OOB"),pch=19,col=c("red","blue"))

RF = c(rmse(test_data$NObeyesdad,rf_1_y_hat),
       mae(test_data$NObeyesdad, rf_1_y_hat),
       R2(test_data$NObeyesdad, rf_1_y_hat))

#### Part 1: COMPARISONS ####
test_pred = predict(lin_mod, newdata = test_data)
train_pred = predict(lin_mod , newdata = train_data)
OLS =  c(rmse(test_data$NObeyesdad, test_pred),
         mae(test_data$NObeyesdad, test_pred),
         R2(train_data$NObeyesdad, train_pred))
test_pred = predict(lin_mod_step , newdata = test_data)
train_pred = predict(lin_mod_step , newdata = train_data)
Stepwise =  c(rmse(test_data$NObeyesdad, test_pred),
              mae(test_data$NObeyesdad, test_pred),
              R2(train_data$NObeyesdad, train_pred))
test_pred = predict(lin_lasso , newdata = test_data)
train_pred = predict(lin_lasso , newdata = train_data)
LASSO =  c(rmse(test_data$NObeyesdad, test_pred),
           mae(test_data$NObeyesdad, test_pred),
           R2(train_data$NObeyesdad, train_pred))
test_pred = predict(lin_mod_lasso_robust , newdata = test_data)
train_pred = predict(lin_mod_lasso_robust , newdata = train_data)
Robust =  c(rmse(test_data$NObeyesdad, test_pred),
            mae(test_data$NObeyesdad, test_pred),
            R2(train_data$NObeyesdad, train_pred))
metrics =  data.frame(OLS,Stepwise,LASSO,Robust,TREE,RF)
row.names(metrics) =  c("RMSE","MAE","R2")
metrics
#### 1) Part 2: INTRODUCTION ####
rm(list = ls())
dev.off()
data = read.csv(file = "~/Desktop/FantaCalcio_Project/Obesity/data/data.csv")

# Data cleaning
data$Age = round(data$Age, 0)
data$Height = round(data$Height, 2)
data$Weight = round(data$Weight, 2)
data$FCVC = round(data$FCVC, 0)
data$NCP = round(data$NCP, 0)
data = data %>%
  mutate(NPC=ifelse(NCP==4,3,NCP))
data$NCP = data$NPC
data = data[, -18]
data$CH2O = round(data$CH2O, 0)
data$FAF = round(data$FAF, 0)
data$TUE = round(data$TUE,0)

#Replace nominal variables with categorical variables
data["FCVC"][data["FCVC"] == 1] = "Never"
data["FCVC"][data["FCVC"] == 2] = "Sometimes"
data["FCVC"][data["FCVC"] == 3] = "Always"

data["NCP"][data["NCP"] == 1] = "1_or␣_2_meals"
data["NCP"][data["NCP"] == 2] = "3_meals"
data["NCP"][data["NCP"] == 3] = "More_than_3_meals"

data["CH2O"][data["CH2O"] == 1] = "Less_than_1_liter"
data["CH2O"][data["CH2O"] == 2] = "1_or_2_liters"
data["CH2O"][data["CH2O"] == 3] = "More_than_2_liters"

data["FAF"][data["FAF"] == 0] = "No_activity"
data["FAF"][data["FAF"] == 1] = "1_or_2_days"
data["FAF"][data["FAF"] == 2] = "2_or_3_days"
data["FAF"][data["FAF"] == 3] = "4_or_5_days"
```

```
data["TUE"][data["TUE"] == 0] = "0-to_2_hours"
data["TUE"][data["TUE"] == 1] = "3_to_5_hours"
data["TUE"][data["TUE"] == 2] = "More_than_5_hours"

quanti = data[,c(2,3,4)]
quali = data[, c(1, 5:16)]
quali = as.data.frame(unclass(quali),stringsAsFactors = TRUE)
data = cbind(quanti, quali)
rm(quali, quanti)
#### 2) Part 2: MCA ####
#1) Model
data_active = data[, c(6:9, 11, 12, 15)]
par(mfrow = c(2, 4))
for (i in 1:7) {
  plot(data_active[,i], main=colnames(data_active)[i],
       ylab = "Count", col="beige", las = 2)
}

res.mca = MCA(data_active, graph = TRUE, ncp = 3)

#Outliers detection with Factoinvestigate and removale
outliers = FactoInvestigate::outliers(res.mca)
outliers$ID

data_active = data_active[-outliers$ID, ]

res.mca = MCA(data_active, graph = TRUE, ncp = 3)


#2) Visualization and Interpretation: Eigenvalues-Variance
eig.val <- get_eigenvalue(res.mca)
fviz_screeplot(res.mca, addlabels = TRUE, ylim = c(0, 45))

#3) Bi-plot
fviz_mca_biplot(res.mca,
                repel = TRUE, # Avoid text overlapping (slow if many point)
                ggtheme = theme_minimal())

#4) Correlation between variables and principal dimensions
fviz_mca_var(res.mca, choice = "mca.cor",
             repel = TRUE, # Avoid text overlapping (slow)
             ggtheme = theme_minimal())

#5) extract the results for variable categories
var = get_mca_var(res.mca)

#6) coordinates of variables to create a scatter plot
round(var$coord, 2)

#7) contains the contributions (in percentage) of the variables to the definition
    of the dimensions.
round(var$contrib,2)

#8) represents the quality of the representation for variables on the factor map.
round(var$cos2,2)

#9) Color by cos2 values: quality on the factor map:
fviz_mca_var(res.mca, col.var = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, # Avoid text overlapping
             ggtheme = theme_minimal())
```

```
#10) Graph of individuals
ind = get_mca_ind(res.mca)

#11) Coordinates of column points
head(ind$coord)

#12) Quality of representation
head(ind$cos2)

#13) Contributions
head(ind$contrib)

#14) Color individuals by groups
fviz_mca_ind(res.mca,
             label = "none", # hide individual labels
             habillage = "FAVC", # color by groups
             palette = display.brewer.pal(n = 7, name = 'RdBu'),
             addEllipses = TRUE, ellipse.type = "confidence",
             ggtheme = theme_minimal())

#15) Dimension description
res.desc = dimdesc(res.mca, axes = c(1,2))

#16) Description of dimension 1
res.desc[[1]]

#17) Description of dimension 2
res.desc[[2]]

#### 4) Part 2: HCPC ####
res.HCPC=HCPC(res.mca, graph = FALSE)

#1) Dendrogram
fviz_dend(res.HCPC, show_labels = FALSE)

#2) Individuals factor map
fviz_cluster(res.HCPC, geom = "point", main = "Factor␣map")

#3) Description by variables
head(res.HCPC$desc.var$test.chi2,10)

#4) Description by variable categories
head(res.HCPC$desc.var$category,10)

#5) Description by principal components
res.HCPC$desc.axes

#6) Individuals closest to the centre
res.HCPC$desc.ind$para

#7) Individuals farest from the centre
res.HCPC$desc.ind$dist

#8) Plots
plot.HCPC(res.HCPC,choice='tree',title='Hierarchical␣tree')
plot.HCPC(res.HCPC,choice='map',draw.tree=FALSE,title='Factor␣map')
plot.HCPC(res.HCPC,choice='3D.map',ind.names=FALSE,centers.plot=FALSE,angle=60,title='Hierarchic
    tree␣on␣the␣factor␣map')
```