# Supervised Learning with R

Luca Paoletti

June 2022


## Contents

# 1 Introduction

During my Statistical Learning course at UNIMI I have decided to take up this project regarding Supervised Learning Algorithms using R as a programming language.
The aim of the project is to use some of this supervised machine learning algorithms in order to make predictions / classifications and compare the results obtained with the different applications. Despite the fact that the aim declared before is to obtain results with those algorithms, a pre-processing and features engineering part was also carried out.

The chosen dataset has the aim of classifying people's obesity level on the basis of some lifestyle's features (eating habits and physical condition). In order to predict the seven different obesity levels I have tried to run two different learning algorithms: Decision Tree model (and Ensemble method using Decision Tree as base classifier), Multiclass Logistic Regression and $K$-NN ($K$ Nearest Neighbour).

# 2 State of the Art

## 2.1 Decision Tree classifier

The decision tree is a method of supervised learning used both for regression and classification. Each tree is made up of nodes and leaves. The nodes are the places where data, based on some conditions, are split; whereas the leaves are the places where data end after the splitting. In each tree, the nodes are tagged with tests and the leaves are tagged with labels.The aim is to obtain the features for which we obtain the best split, and to do so, we can use Gini function (it refers to Gini impurity) or Cross Entropy (it refers to Information Gain).

The *Gini Index* is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

measure of total variance across the $K$ classes, where $\hat{p}_{mk}$ represents the proportion of training observations in the $m$-th region that are from the $k$-th class. The Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one.
For this reason the Gini index is referred to as a measure of node *purity* - a small value indicates that a node contains predominantly observations from a single class.

An alternative to the Gini index is *Cross-Entropy*, given by

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

As far as Entropy refers to *Information Gain*, we need to introduce also this last metric. We can define information gain as a measure of how much information a feature provides about a class. Information gain helps to determine the order of attributes in the nodes of a decision tree.
The main node is referred to as the parent node, whereas sub-nodes are known as child nodes. We can use information gain to determine how good the splitting of nodes in a decision tree is.
It can help us determine the quality of splitting, as we shall soon see. The calculation of information gain should help us understand this concept better.

$$Gain = E_{parent} - E_{children}$$

The term Gain represents information gain. $E_{parent}$ is the entropy of the parent node and $E_{children}$ is the average entropy of the child nodes.

### 2.1.1 Ensemble method: Boosting

Ensemble methods are meta-algorithms that add randomization to learning algorithms so to have them generate multiple predictors on the same training data. These predictors are then aggregated, either linearly or by taking a majority vote, to form a single predictor. Adding randomization and then combining predictors has an averaging effect on the performance of the learning algorithm, which corresponds to reducing variance and increasing bias. For this reason, ensemble methods are often used with nonparametric learning algorithms (such as tree predictors) that have little bias.

Boosting is a general approach that can be applied to many statistical learning methods for regression or classification. Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly.
Intuitively, each new model focuses its efforts on the most difficult observations to fit up to now, so that, at the end of the process, we obtain a strong learner with lower bias.

## 2.2 Multiclass Logistic Regression

In order to understand the mutliclass version of the Logistic Regression, it is important to explain the binary logistic regression.
Let's write $p(X) = Pr(Y = 1|X)$ for short having only one covariate and a dicotomic response variable. Logistic Regression uses the form

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

It is easy to see that no matter what values $\beta_0, \beta_1$ or $X$ take, $p(X)$ will have values between 0 and 1.
A bit of rearrangement gives

$$\log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X.$$

This monotone transformation is called the *log odds* or *logit* transformation of $p(X)$.

It is easily generalized to more than two classes. One version has the symmetric form

$$Pr(Y = k|X) = \frac{e^{\beta_{0k}+\beta_{1k}X_1+...+\beta_{pk}X_p}}{\sum_{l=1}^{K} e^{\beta_{0l}+\beta_{1l}X_1+...+\beta_{pl}X_p}}$$

here there is a linear function for each class.

## 2.3 K Nearest Neighbour (KNN)

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. The algorithm is quite mechanical and can be summed up in the next points:

1. Load the data
2. Initialize $K$ to your chosen number of neighbour
3. For each example in the data
   (a) Calculate the distance between the query example and the current example from the data
   (b) Add the distance and the index of the example to an ordered collection

4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first $K$ entries from the sorted collection
6. Get the labels of the selected $K$ entries
7. If regression, return the mean of the $K$ labels
8. If classification, return the mode of the $K$ labels

KNN's main disadvantage of becoming significantly slower as the volume of data increases makes it an impractical choice in environments where predictions need to be made rapidly. However, provided you have sufficient computing resources to speedily handle the data you are using to make predictions, KNN can still be useful in solving problems that have solutions that depend on identifying similar objects.

# 3 The Data

## 3.1 Description of the Dataset

This dataset contains data for the estimation of obesity levels in people from the following countries: Mexico, Peru and Colombia, with ages between 14 and 61 and different eating habits and physical condition. Data were collected using a web platform with a survey where anonymous users answered each question, then the information was processed obtaining 17 attributes and 2111 records.
The attributes related with eating habits are: Frequent consumption of high caloric food (FAVC), Frequency of consumption of vegetables (FCVC), Number of main meals (NCP), Consumption of food between meals (CAEC), Consumption of water daily (CH20), and Consumption of alcohol (CALC). The attributes related with the physical condition are: Calories consumption monitoring (SCC), Physical activity frequency (FAF), Time using technology devices (TUE), Transportation used (MTRANS), other variables obtained were: Gender, Age, Height and Weight. Finally, all data were labelled and the class variable NObesity was created with the values of: Insufficient Weight, Normal Weight, Overweight Level I, Overweight Level II, Obesity Type I, Obesity Type II and Obesity Type III.

After all data were collected, then data were preprocessed, so they could be used for different techniques of data mining. The number of records was 485 records, and the data were labelled using equation

$$\text{Mass Body Index} = \frac{\text{Weight}}{\text{height} \times \text{height}}$$

The labels obtained are the following:

- Underweight $\rightarrow$ less than 18.5
- Normal $\rightarrow$ 18.5 to 24.9
- Overweight $\rightarrow$ 25.0 to 29.9
- Obesity I $\rightarrow$ 30.0 to 34.9
- Obesity II $\rightarrow$ 35.0 to 39.9
- Obesity III $\rightarrow$ Higher than 40

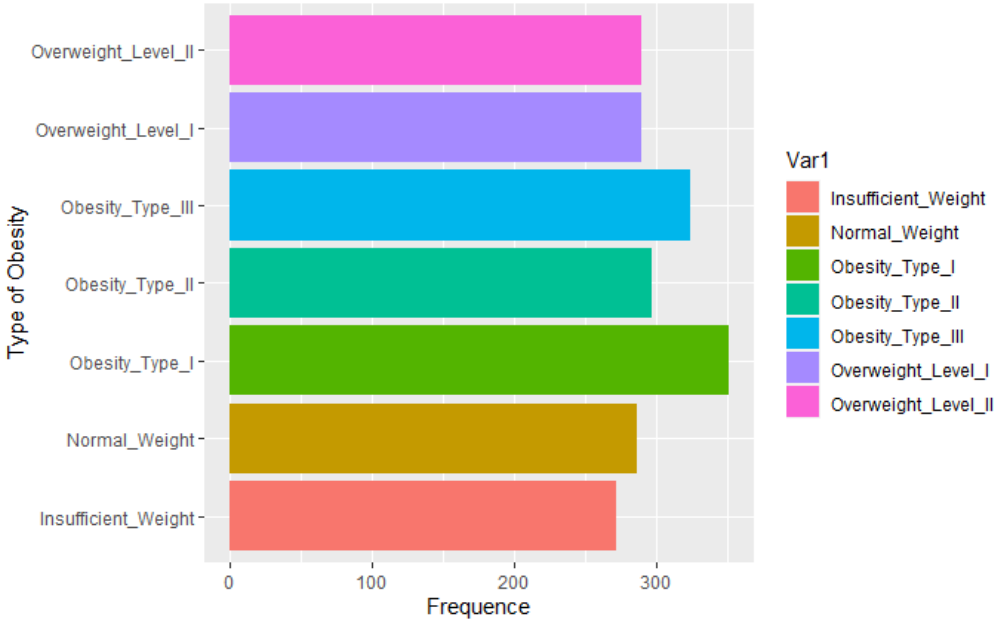


Figure 1: Distribution of the response variable

## 3.2 Pre-Processing

As far as this data have been collected through a multiple question questionnaire, I have decided to relabel most of the continuous features to factor variable. Of course, this new setup follows the questionnaire sent to the participants.
The changes are

- FCVC feature answers to the question 'Do you usually eat vegetables with your meal?' and the multiple choices were:
  - Never
  - Sometimes
  - Always

  Looking at the boxplot of the continuous feature, I have decided to split the varibles in
  - $< 2$ never
  - $= 3$ always
  - *else* sometimes
- NCP feature answers to the question 'How many main meals do you have daily?' and the multiple choices were:
  - Between 1 and 2
  - 3
  - more than 3

  always looking at the boxplot I have decided to split them into
  - $= 3$ three
  - $< 3$ less than three
  - *else* more than three

- finally, the consumption of water feature, CH2O, split in
  - $< 1$ less than 1l
  - $> 2$ more than 2l
  - *else* between 1l and 2l

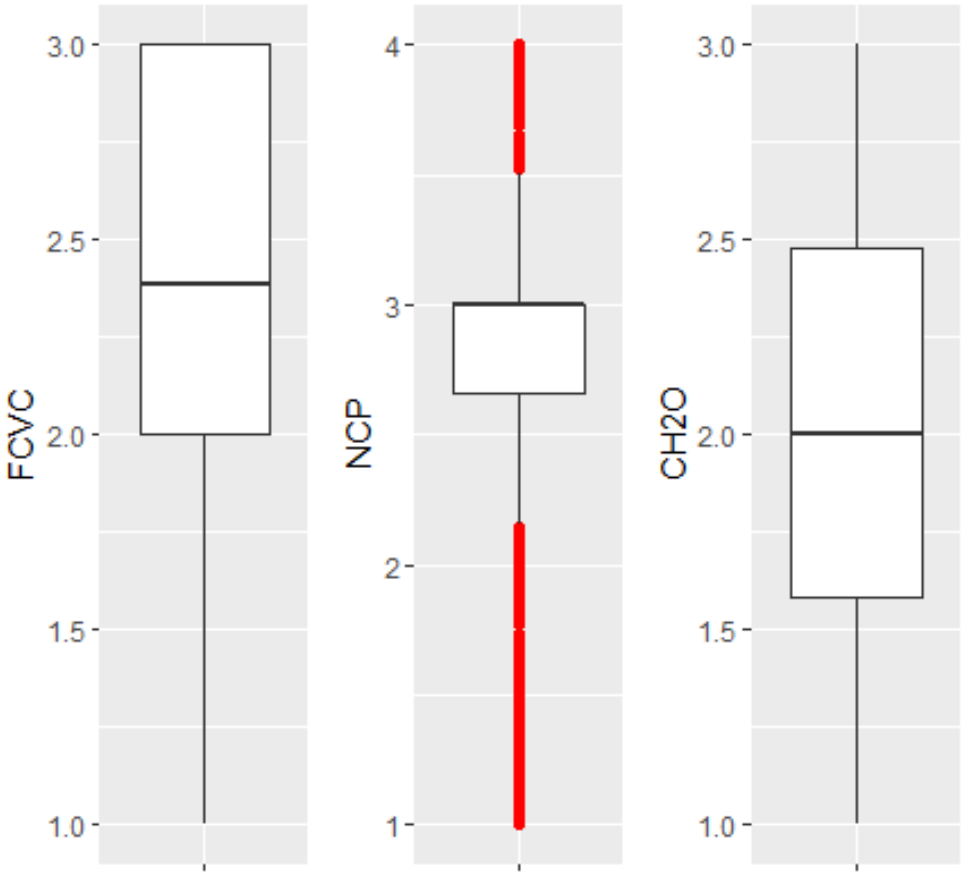


Figure 2: Continuous to categorical boxplot

My next pre-processing step was checking the presence of NA and fortunately, there are no NAs in the dataset. The next focus is on the correlation between the features, checking some possible sort of collinearity or multicollinearity. With these two definitions, I mean the presence of linear relation between 2 (collinearity) or more than 2 (multicollinearity) independent variable. Of course, this type of relation could be detected among continuous features (e.g. Age, Height, Weight, Physical activity frequency (FAF) and Time using technology devices (TUE)).
The best way to check for collinearity, in my opinion, is through the correlation matrix.

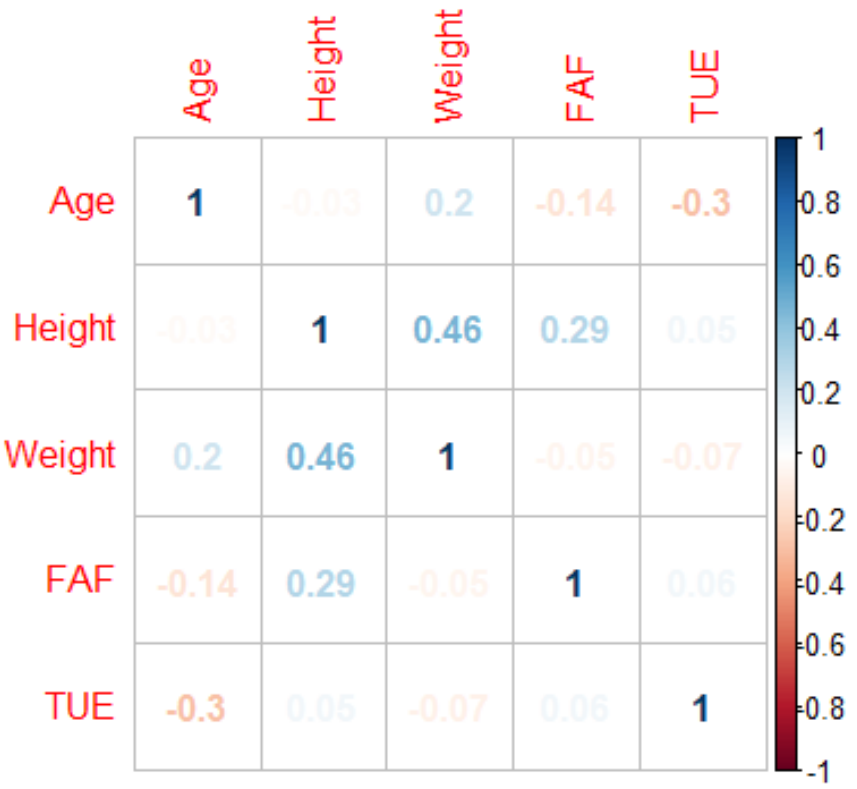


Figure 3: Correlation Matrix

Thanks to the plot displayed above, it is possible to notice that there is not high correlation. The highest one is between Height and Weight but, as far as those are one of the main attributes useful in order to check the obesity level of a person, I have decided not to remove them from the dataset.

Moreover, it is important to underline if there are some possible relations or different distributions between the answer's feature ($y$) and the covariates ($x_i$). In the next boxplots, it is also possible to see the outliers in red. As we can clearly see, the different

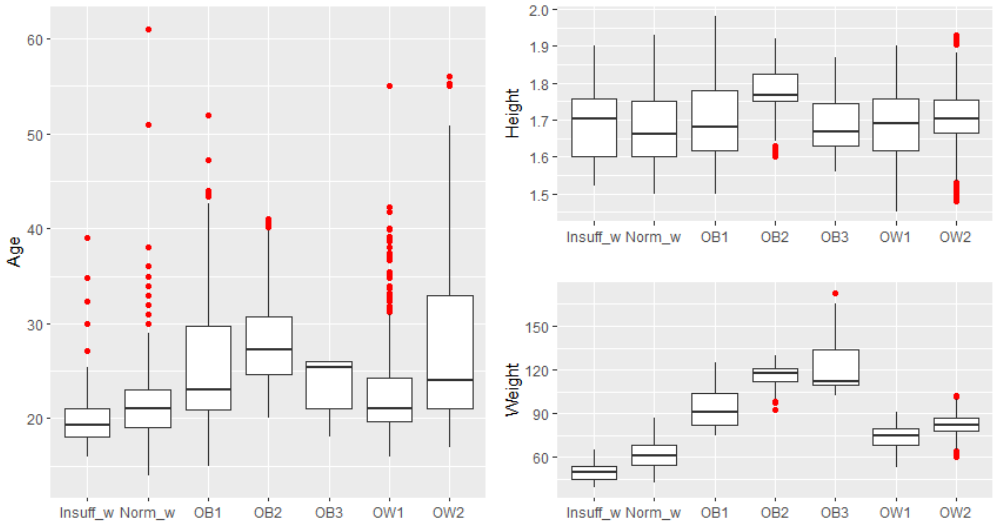


Figure 4: How will $y$ change related to $x_i$?

levels of $y$ change due to the changes in $x$ and due to changes in $x$ we can also underline different outliers. In this particular case, I have decided not to remove outliers because these dots are influential dots only for one ore two levels of the response variable. What does that mean? Basically, if a weight of 60 is an outliers for Obesity I, it doesn't mean that it is an outlier for Obesity II, too.

## 3.3 Descriptive Analysis

With this last subsection of the Data chapter, I just want to show some descriptive analysis that I made in order to better understand the meaning of the dataset.
For example, the following plot represents how the response variable is distributed respectively to Weight and Height.

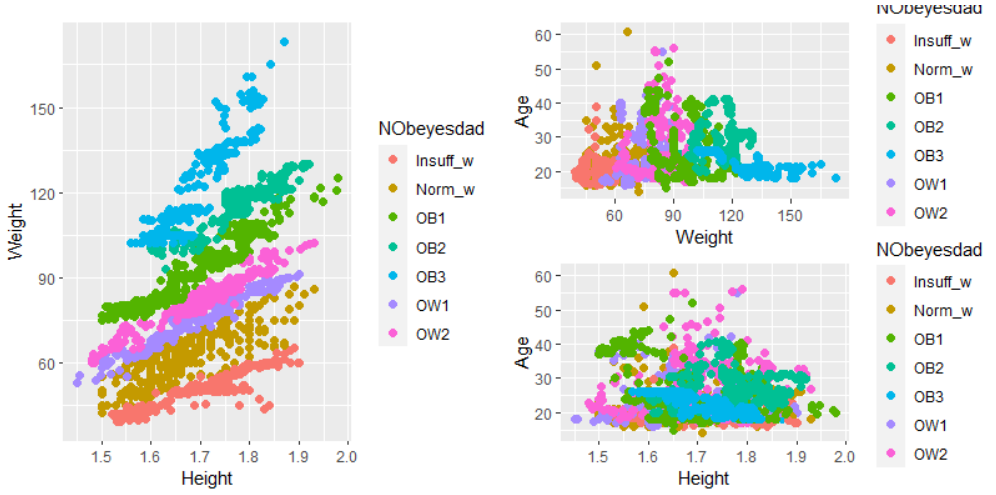


Figure 5: Distribution effects on $y$

It is possible to see 7 different clusters, that is the 7 different classes of $y$.

Another example, not so clear the previous one, is the position of the clusters with respect to Weight and Age or Height and Age. With these analysis I have considered the preprocessing and feature engineering part completed and I have started the explanation of the different models.

# 4 Models

## 4.1 Decision tree and Boosting method

First of all, I have split the dataset into train and test set, in particular the train is 2/3 of the test set. Then I have fitted the Decision Tree giving as input all the covariates except for 'family history with overweight', 'SMOKE', 'MTRANS' and 'CALC' and Entropy as splitting method. The excluded features have been removed because of collinearity with other more important features and because of the risk of overfitting. A model with so many independent variables could be intended as too precise and too complex.
The result is a tree with max depth = 6 and using Weight, Height, FAVC and Gender as splits. Of course the most used one is the weight but, through a deeper analysis, more specific features such as Gender and FAVC can helps too.
Predictions made with this fitted tree, have returned not completely satisfactory results; for example I can see from the next plot how the prediction is really good for some classes (OB1, OB2, OB3) but not so good for other ones (OW1, Normal Weight).

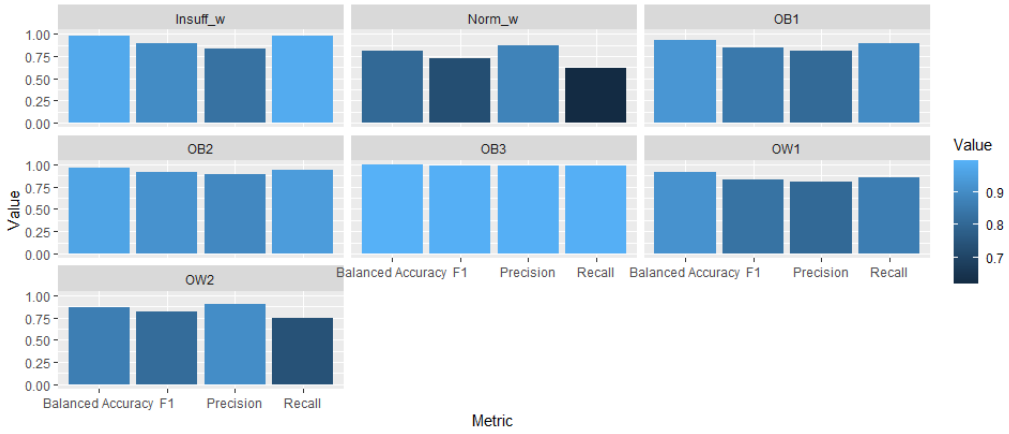


Figure 6: Evaluation metrics for Decision Tree

In order to fix this issue, I have tried to run a boosting ensemble method in order to get higher evaluation metrics results; in particular the focus is to reduce variance and increase bias. It could be that some of the really well predicted classes obtain these results thanks to the low variance in them.
In fact, thanks to the Boosting ensemble method, I get better results in all the classes, not just the ones that had been mis-classified when using the decision tree.


Figure 7: Evaluation metrics for Boosting ensemble method

In addition, the features importance changes, too. Weight and height are no more the most used features in the model, and this could also improve our model thanks to the fact that, as we have previously seen in the graph (Figure 5), Weight and Height could easily target the obesity level of a person.

## 4.2 Logistic Regression

For this second model too, I have decided to split the dataset in train (2/3) and test (1/3). The model used is a Multinomial Logistic Regression with response variable equal to the levels of obesity and as independent variables the same ones used for the Decision Tree model.
In this case, the performances of the regression are good even for the classes that the base Decision Tree without ensemble method could not predict accurately.

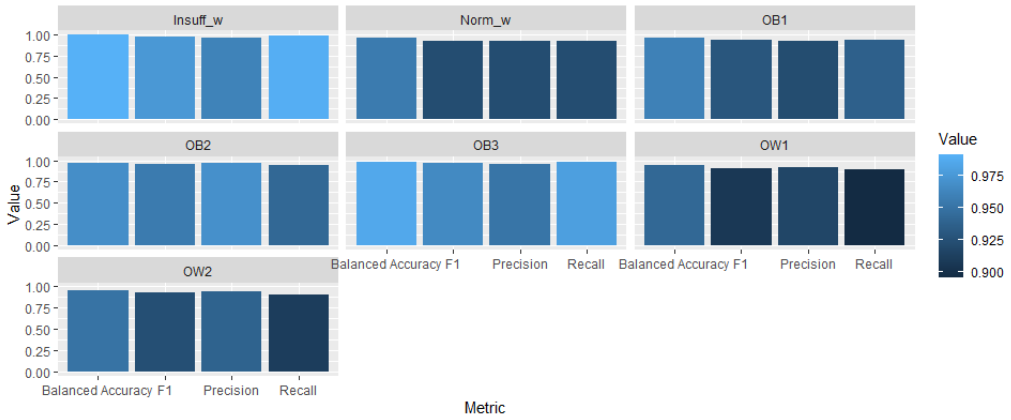


Figure 8: Evaluation metrics for Logistic Regression

In this graph it is possible to see how the Logistic regression seems more accurate with respect to the Decision Tree model. The reason could be that, a decision tree with so many features, will always use the same independent variables as split and consequently it could become a poor model; in this case we have seen how weight and height are basically the most important features in order to let the tree grow.

## 4.3 K Nearest Neighbour

The last model used is the K Nearest Neighbour. Until now I had not made any type of hyperparamethers tuning. Why? Basically, the models used till now were auto tuned by

the R function (depth of the decision tree) or, easily, the model did not need it (in the logistic regression the removal of some features was enough to get good results).
With the KNN it is very important to tune the correct value of $K$. Choosing a too high $K$ will result in an underfitting model (too many neighbours to consider), on the other hand, a really small $K$ will create overfitting in the model (too complex). In order to use the right $K$ I have decided to run the KNN model multiple times and, each time, I have stored and then plotted the accuracy level. The result is the following plot.

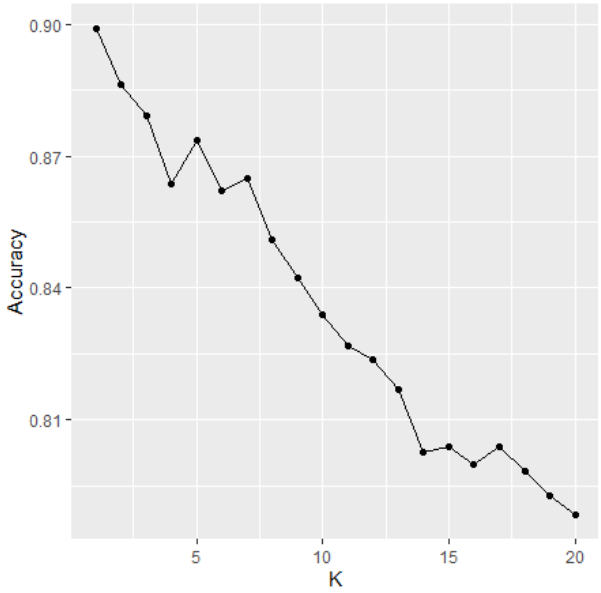


Figure 9: How $K$ changes accuracy level

It seems that the higher accuracy is obtained with $K = 1$ but, as I wrote before, too small$K$ will create overfitting. I have chosen $K = 3$ the following knee-elbow method (increase after a huge drop in the value of the accuracy).
Once I have fitted the model with right value of $K$ ($K = 3$), I have predicted the test $y$ and got these results.

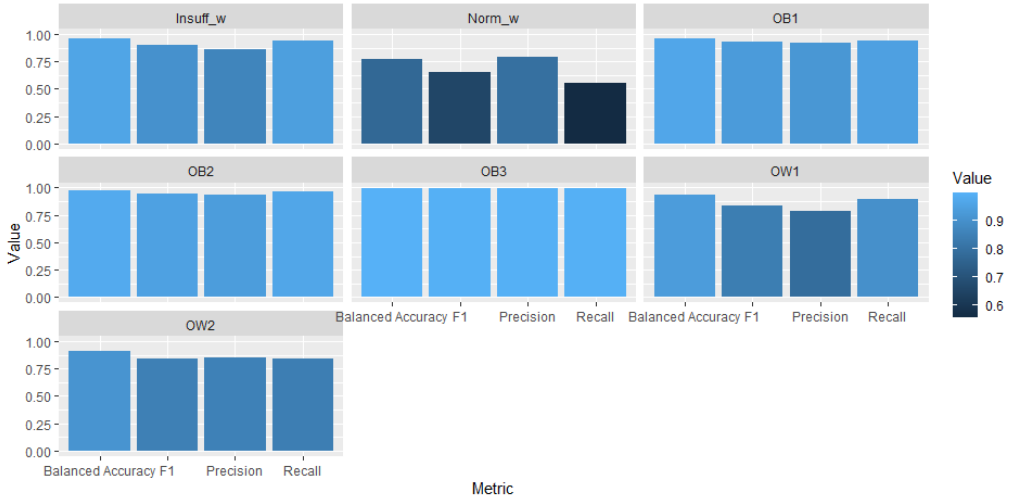


Figure 10: Evaluation metrics for 3 - NN

# 5 Findings and Conclusion

Results and findings of this supervised learning projects were already mentioned with the evaluation plots right above. The dataset chosen suits perfectly the models run. I have already mentioned the evaluation metrics for each model and I am going to display a summary of the 4 right below.

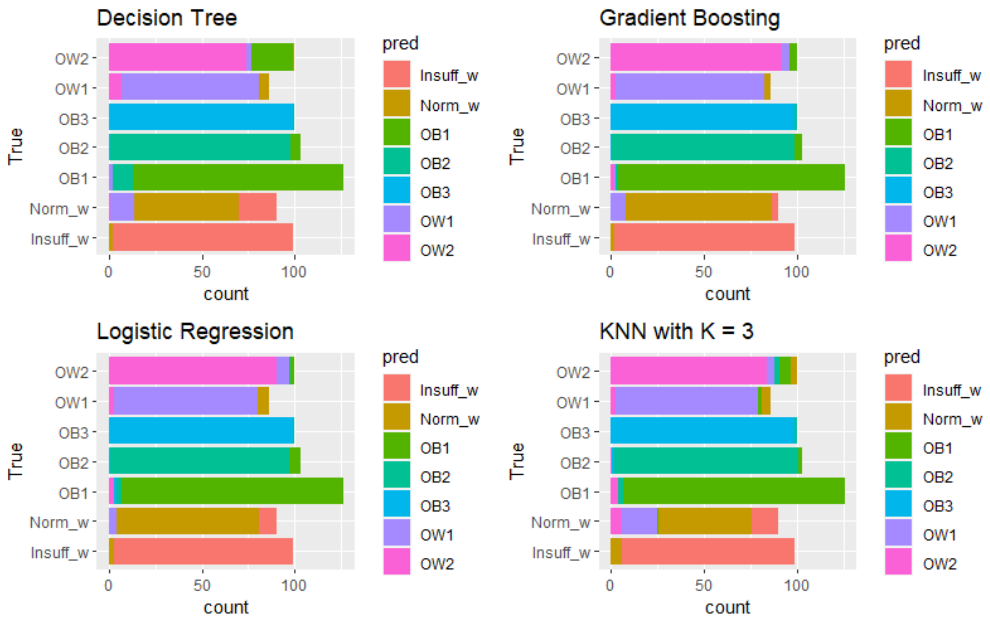


Figure 11: Predictions

The first finding that I can underline, is the precision of the models. Of course I do not have a 100% accuracy but, most of the test set records are correctly classified. In addition, it is easily notable that when the record is miss classified, most of the time the level of obesity assigned to the person is near to the correct one (e.g. TRUE = OB1, PREDICTED = OB2). That means that the model, despite the fact that it missclassifes the person, also gets really really close to the correct level.

The second result concerns features importance. Variables like weight and height are not the only ones useful in order to classify people's level of obesity, in fact, as it is clearly showed by the next plots, other features are often more important.

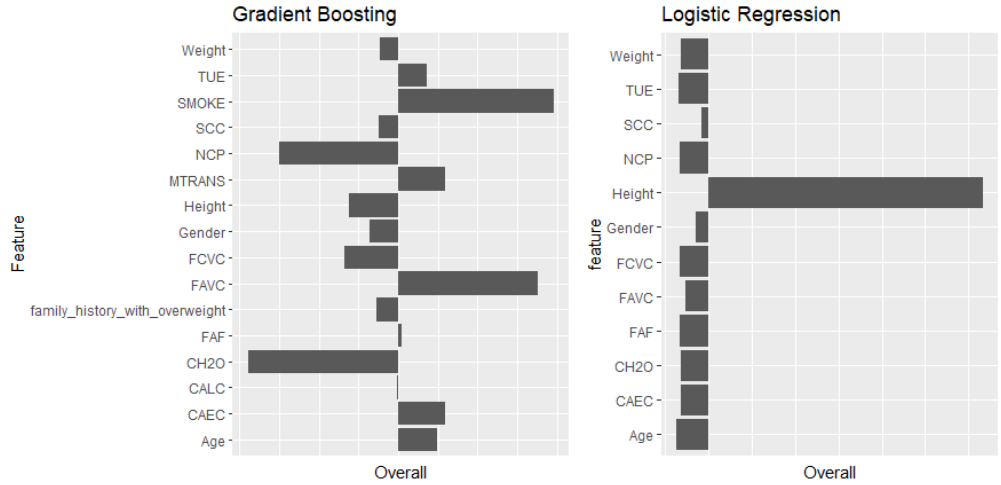


Figure 12: Feature Importance

As we can see, for the Boosting ensemble method the most important features are SMOKE, FAVC, MTRANS and many others. Of course, as far as ensemble method add randomization to the model, it could happen that running several times the algorithm, the ranking could change. On the other hand, looking at the feature importance graph of Logistic regression, Height followed by Gendere and SCC are the most important independent variable in the model.

Last but not least, the importance of a survey as method for collecting data. In the pre processing part I did not have to work a lot thanks to the goodness of the dataset. When you ask a person to answer some 'personal' questions, the answer will always be true. No outliers, no NAs and most of the feature were not linear correlated and clean from possible distribution noise.

In conclusion, I would say that the effectiveness of the project (based on the evaluation metrics), is mostly due to the cleanliness of the dataset. Sometimes, working with missing values or NA you end up with classification problems sch as underfitting, substitute NA with the wrong value or removing too many outliers. In this particular case I didn't have to work with them and consequently the models work fairly well every time.