Statistical Learning report:
Detecting Heart Diseases

Doina Vasilev

March 2023

Abstract

The aim of the report is to present the main usefult steps to carry out a statistical analysis.
For this purpose, I've focused on a dataset provided by the University of California, containing several qualitative and quantitative variables relative to 918 patients, with the aim of predicting heart diseases.
To do so, I've first conducted an extensive data pre-processing, which resulted in a final training set where the missing values of **Cholesterol** have been replaced using the "random forest" algorithm, and where the classes of **Sex** and **FastingBS** were re-balanced.
Then, I've applied three supervised algorithms, such as decision trees, random trees, and bagging to provide predictions on the outcome variable **HeartDisease**.
For each, I've provided few performance metrics, such as accuracy, precision, recall, sensitivity and sensibility to evaluate their performance.
The results show that models trained on balanced sets obtain improved predictive accuracy, and overall better performances, even when tested on unbalanced data.
Moreover, decision trees and random forests highlight the importance of the variables ST-Slope and ChestPainType in detecting the presence of heart diseases.
Finally, I've applied hierarchical clustering on the unbalanced and the balanced datasets, obtaing for both two optimal number of clusters.

1 The data

The dataset is retrieved from the UCI Machine Learning Repository, i.e., a collection of databases, domain theories, and data generators provided by the University of California.

The original dataset contains 918 observations with 11 numerical and categorical features, and the final output, Heart Disease, which is the binary variable I wish to predict.
Age: (*numeric*) feature representing the age of the patient.
Sex: (*binary*) feature describing the sex of the patient. The two possible values are: **M** (male), and **F** (female).
ChestPainType: (*character*) feature describing the type of chest pain. It can take four possible values: **TA**(typical angina)[1], **ATA** (atypical angina), **NAP** (non-anginal pain), **ASY** (asymptomatic).
RestingBP: (*numeric*) feature measuring the level of blood pressure during rest. It's measured in mm/Hg.
Cholesterol: (*numeric*) feature measuring total serum cholesterol in mg/dl.
FastingBS: (*binary*) feature representing whether the value of blood sugar is higher or lower than 120 mg/dl. It can take two values only: **1** (if FastingBS is greater than 120 mg/dl), or **0** (otherwise).
RestingECG: (*character*) feature representing resting electrocardiogram results. It can

[1]Angina is a type of chest pain commonly associated to coronary heart disease, which cuts off blood flow to the heart.

take the values: **Normal**, **ST** when the patient presents ST-T wave abnormality (T wave inversions and/or ST elevation or depression greater than 0.05 mV), or **LVH** when the patient shows probable or definite left ventricular hypertrophy by Estes' criteria.
MaxHR: (*numeric*) feature representing maximum heart rate achieved.
ExerciseAngina: (*character*) feature determined whether Angina (ChestTypePain) is exercise-induced. It can take two values: **Y** (yes), and **N** (no).
Oldpeak: (*numeric*) feature representing the level of ST depression induced by exercise with respect to rest. Again, ST relates to the position on the ECG results.
ST-Slope: (*nominal*) feature representing the slope of the ST segment at peak exercise. It can take three possible values: **Up** (upsloping ST), **Flat** (flat ST), **Down** (downsloping ST).
HeartDisease: (*binary*) feature representing the output variable, i.e., what I wish to predict. It can take two possible values: **1**, if the patient has (or is predicted to have) heart disease, or **0**, if patient has not (or is not predicted to have) an heart disease.

Attribute	**Description**	**Data Type**	**Domain**
Age	Patient's age in years	Numerical	28 - 77
Sex	Patient's sex	Binary	[M, F]
ChestPainType	Type of chest pain	Nominal	[ASY, ATA, TA, NAP]
RestingBP	Blood pressure at rest	Numerical	0 - 200
Cholesterol	Total serum cholesterol	Numerical	0 - 603
FastingBS	Level of blood sugar higher or lower than 120 mg/dl	Binary	[Y,N]
RestingECG	ECG results	Nominal	[Normal, ST, LVH]
MaxHR	Maximum heart rate achieved	Numerical	60 - 202
ExerciseAngina	Exercise-induced angina	Binary	[0, 1]
Oldpeak	Exercise-induced level of ST depression	Numerical	(-2.6) - 6.2
ST-Slope	Peak-exercise ST slope	Nominal	[Up, Down, Flat]
HeartDisease	Output variable	Binary	[Normal, HD]

Table 1: Summary of the dataset.

1.1 Numerical Data

Table 2 provides a brief summary of some of the most important indices and values representing the distributions of the numerical variables.
From the start, we can observe anomalous values for the variables **RestingBP** and **Cholesterol**: in both cases, the dataset presents unattainable minimum values for a living human being, and extreme maximum values in the case of **Cholesterol**.

	Age	**RestingBP**	**Cholesterol**	**MaxHR**	**Oldpeak**
min	28	0.0	0.0	60.0	-2.6
1st Q.	47	120.0	173.2	120.0	0.0
median	54	130.0	223.0	138.0	0.6
mean	53.51	132.4	198.8	136.80	0.8874
3rd Q.	60.0	140.0	267.0	156.0	1.5
max	77	200.0	603.0	202.0	6.2

Table 2: Summary statics for numerical variables.

1.1.1 Distributions

1.1.2 Densities and Histograms

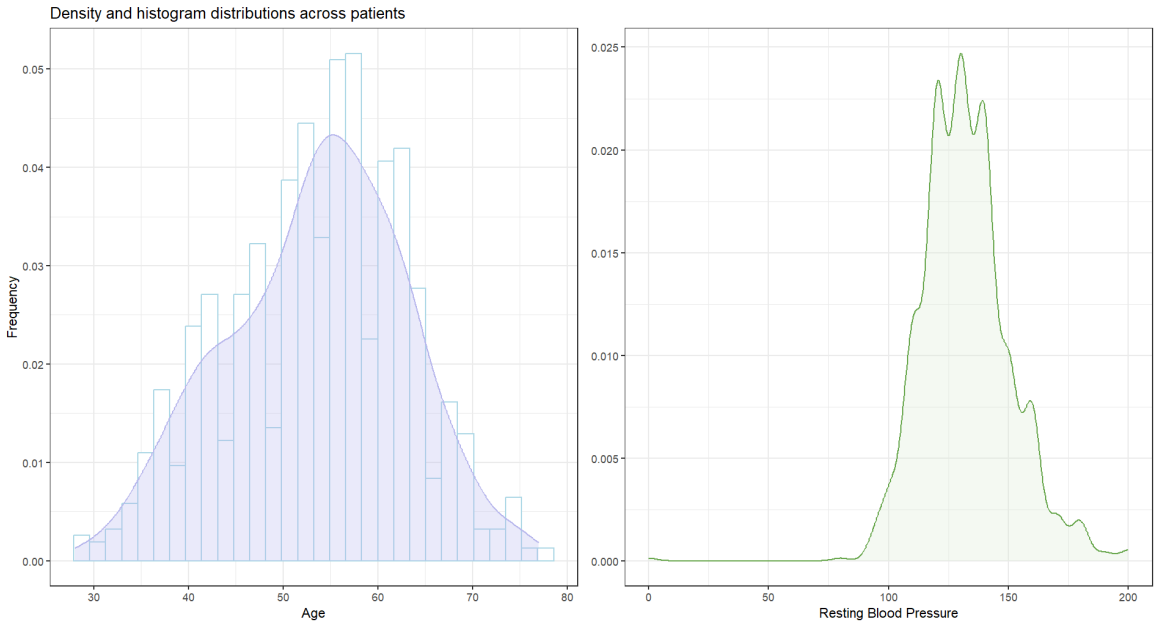

Figure 1: Distributions of Age and Resting Blood Pressure across the observations.

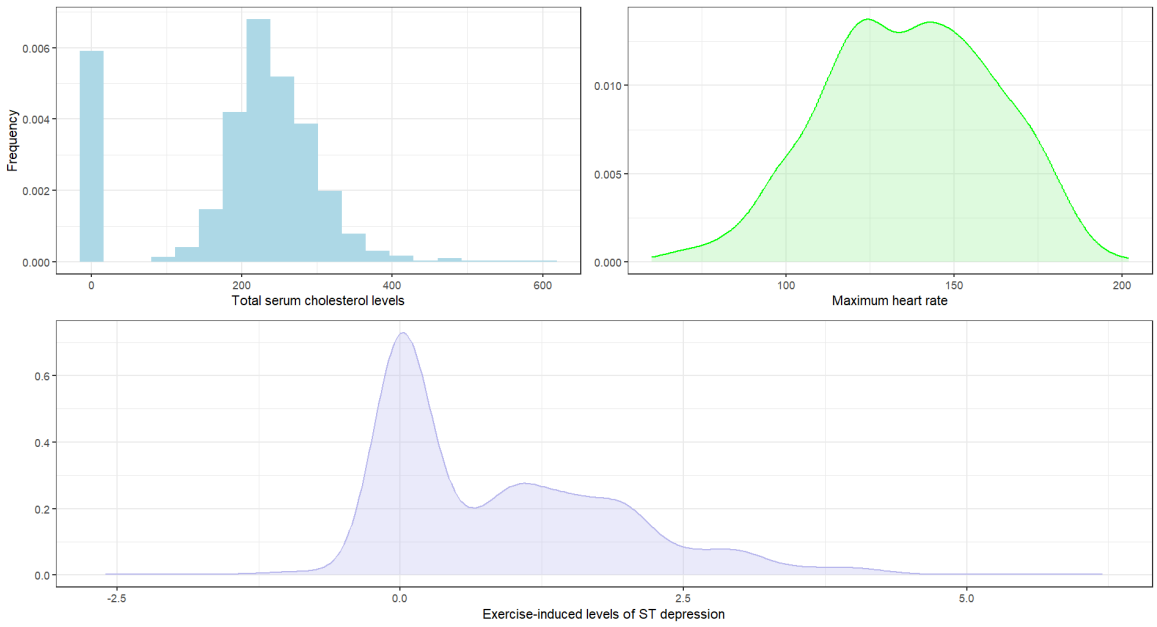

Figure 2: Distributions of Cholesterol, MaxHR and Exercise-induced levels of ST depression.

1.2 Categorical Data

The following Table provides the distributions over the classes of the categorical variables.

Variable		Count	Frequency
Sex	M	725	78.98 %
	F	193	21.02 %
	Total	918	100. 00 %
Fasting Blood Sugar	0	704	76.69 %
	1	114	23.31 %
	Total	918	100. 00 %
ChestPainType	ATA	173	18.85 %
	TA	46	5.01 %
	NAP	203	22.11 %
	ASY	496	54.03 %
	Total	918	100.00 %
ST_Slope	Flat	460	50.11 %
	Up	395	43.03 %
	Down	63	6. 86 %
	Total	918	100.00 %
ExerciseAngina	Y	371	40.41 %
	N	547	59. 59 %
	Total	918	100.00 %
RestingECG	Normal	552	60.13 %
	ST	178	19.39 %
	LVH	118	20.48 %
	Total	918	100.00 %

Figure 3: Distributions over the classes of categorical variables.

2 Data pre-processing

The original dataset does not contain null values, however the following issues arise:
1. The variables **Cholesterol** and **RestingBP** present unattainable values for a living human being. In particular, the first variable showcases extreme low and high values of total serum cholesterol, whilst the latter only extreme minimum levels of blood pressure. Both these issues highly influence the distribution of the variables, and are potentially introducing risky *bias*.
2. Among the categorical variables there are several imbalanced classes, such as **Sex**, where almost 80% of the observations are from male individuals.
Male individuals are more likely to be diagnosed with heart diseases, however it's important to fix these class imbalances so that the trained models are not biased towards them. In particular, I've decided not to eliminate the female patients altogether for two reasons: first, it would reduce the already quite small dataset I'm working on; secondly, for ethical reasons: medical history has for a long time neglected women's health in clinical research as argued by the US Public Health Service Task Force on Women's Health, which warned that "*the historical lack of research focus on women's health concerns has compromised the quality of health information available to women as well as the health care they receive.*"[2] However, running peculiar models on the sexes, rather than generic ones as in such case, could be an option and could result in improved overall performances.

The next variable I'm going to work on is **FastingBS**, for which 77% of the observations fit into the "0" class, i.e., 77% of the patients showcase lower values of blood sugar than 120 mg/dl after fasting.
Although both scenarios are good representations of the real world, it's best to train the models to be as balanced as possible, in order to avoid great bias deriving from class imbalance.

2.1 RestingBP

In the case of **RestingBP**, only one observation presents a value of **0.0**, therefore I simply removed the observation that introduced the bias, which did not cause much difference in the variable's distribution.
The results are shown in Table 3.

	RestingBP - Before	**RestingBP - After**
min	0.0	80.0
1st Q.	120.0	120.0
median	130.0	130.0
mean	132.4	132.5
3rd Q.	140.0	140.0
max	200.0	200.0

Table 3: Changes in RestingBP before and after removing the observation related to min values = 0.

[2] *Ethical and Legal Issues Relating to the Inclusion of Women in Clinical Studies*, US Public Health Service Task Force (1995).

2.2 Cholesterol

As seen previously, the distribution of Cholesterol is greatly skewed by the presence of 0s. Indeed, **171** patients have cholesterol levels equal to 0, which amounts to the 18.65% of the total sample.
Since the number of observations having anomalous values is quite high with respect to the total, I've decided not to delete the related observations. However, I've attempted several strategies to substitute the zeros.

2.2.1 Strategies

1. The first strategy I've exploited was indeed to remove the observations related to the anomalous values of Cholesterol, obtaining a dataset of **746** data points.
2. The second strategy implemented is to replace the zeros with the median. There's no loss of information with respect to the first strategy; however, the values are heavily skewed towards the central value, distorting the distribution of the variable.
3. The third strategy has required the implementation of the **Predictive Mean Matching** (pmm) algorithm, from the Mice package.
The algorithm was first introduced by D.B. Rubin in 1986 (**D. B. Rubin**; **1986**), and then by R.J.A.Little in 1988 (**R. J. A. Little**; **1988**).
The algorithm works in following way:
– First, it selects the variable for which we're performing the imputation.
– Second, it creates a regression of the variable to be imputed (y_i) on the other variables, and returns $\hat{y}$.
– Finally, the algorithm picks the model that minimizes the loss:

$$\hat{y}_{j,final} = argmin|\hat{y}_{j,miss} - \hat{y}_{i,obs}|; i \in observed \tag{1}$$

Where $\hat{y}_{i,obs}$ is the vector space of the observed y_i, and $\hat{y}_{j,miss}$ is the vector space of the estimates $\hat{y}_j$.
4. The fourth, and winning, strategy requires to replace the zeros with values estimated using **Random Forests**.
The algorithm was first proposed by **L.Breiman (2001)**, and it is now implemented by the R package *randomForest*, as it was originally defined.
The desirable characteristics of RF allow them to: [1] handle mixed types of missing data; [2] address interactions and nonlinearity; [3] scale to high-dimensions while avoiding overfitting; and [4] yield measures of variable importance useful for variable selection (**Tang. F & Ishwaran H.**; **2017**)
The algorithm's approach for imputing missing values is the following: (1) pre-impute the data, (2) grow the forest, (3) update the original missing values using proximity of the data. (4) Iterate for improved results. (**Tang. F & Ishwaran H.**; **2017**). Thus, it's a reiterative process, where the data is repetitively imputed until it reaches best results.
The results are illustrated in the Table 4, and Figure 4.

	n. observations	**min**	**1st Q.**	**median**	**mean**	**3rd Q.**	**max**
Original dataset	918	0.0	174.0	223.0	199.0	267.0	603.0
Strat. 1	746	85.0	174.0	223.0	199.0	267.0	603.0
Strat. 1	746	85.0	207.2	237.0	244.6	275.0	603.0
Strat. 2	917	85.0	214.0	223.0	240.6	267.0	603.0
Strat. 3	917	85.0	208.0	239.0	246.1	277.0	603.0
Strat. 4	917	85.0	207.0	236.0	234.8	274.0	603.0

Table 4: Different imputation strategies for the variable Cholesterol.

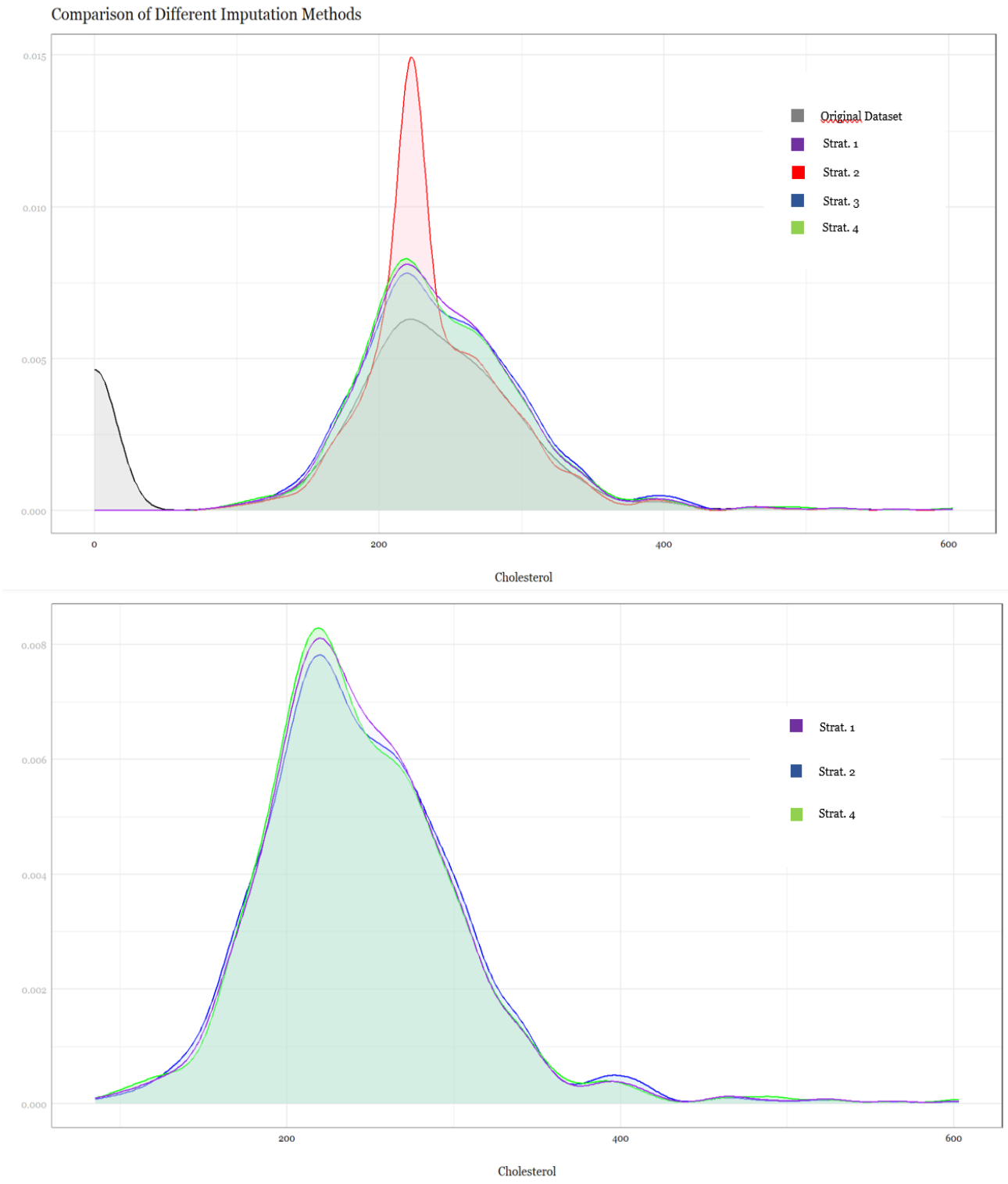

Figure 4: Distributions of Cholesterol based on different imputation strategies.

2.3 Outliers detection

Table 5 showcases the distributions of the observations for the 5 numerical variables.
We can observe a high number of outliers in the cases of **Cholesterol** (29 outliers), **RestingBP**, (27) and **Oldpeak** (16).
I've only focused on the outliers observed in **Cholesterol**, since it presents unattainable lower and higher values for human beings.
On the other hand, the outliers of RestingBP and Oldpeak lie inside a reasonable range of values, thus I've decided not to remove them.
On the other hand, the models I'm going to employ for prediction of the output variable (*decision trees* and *random trees*) are robust to outliers, thus I won't observe great differences in the performance of the models.
Tables 5 and 7 depict the new properties of the clean dataset.
As shown in Figure 6, several outliers have been left in the dataset: since decision and random trees are robust to outliers I've decided to remove the extreme and unattainable values only. Moreover, since the discussion is centered around medical data, I believe that the outliers represent important information in the detection of diseases.

	Age	**RestingBP**	**Cholesterol**	**MaxHR**	**Oldpeak**
min	28	80.0	**100.0**	60.0	-2.6
1st Q.	47	120.0	**207.5**	120.0	0.0
median	54	130.0	**236.0**	138.0	0.6
mean	53.55	132.6	**241.0**	136.7	0.883
3rd Q.	60.0	140.0	**274.0**	156.0	1.5
max	77	200.0	**394.0**	202.0	6.2
n.obs	**899**	**899**	**899**	**899**	**899**

Table 5: Summary statics for numerical variables in the *Clean.csv* dataset.

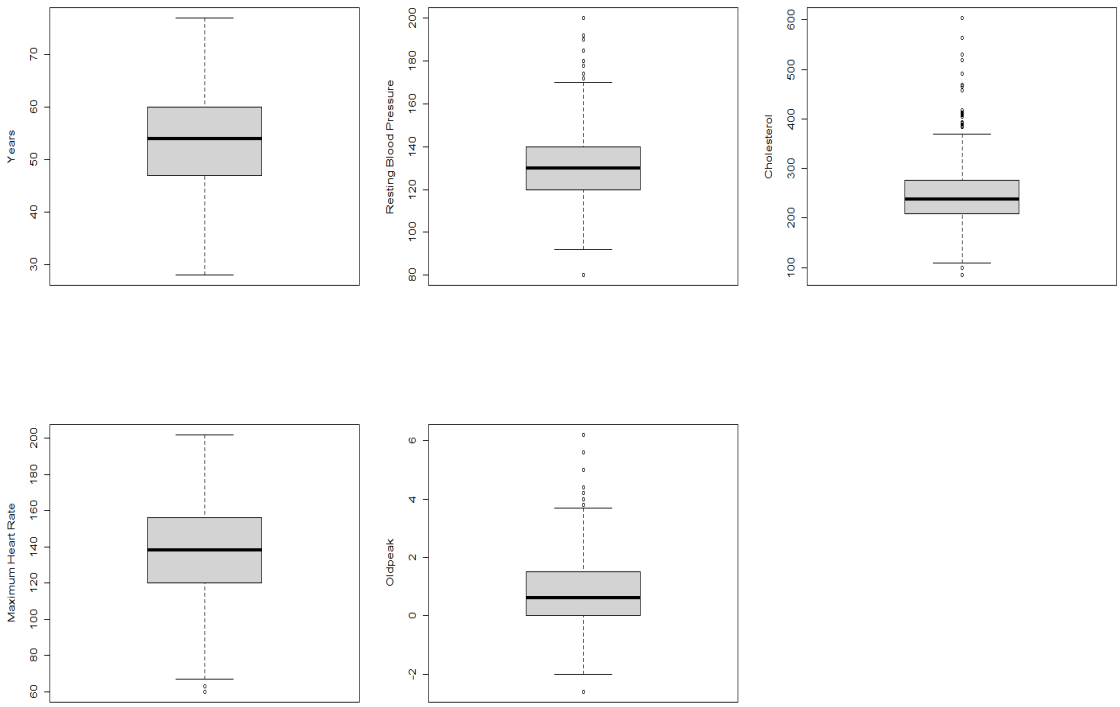

Figure 5: Boxplots of the original numerical variables.

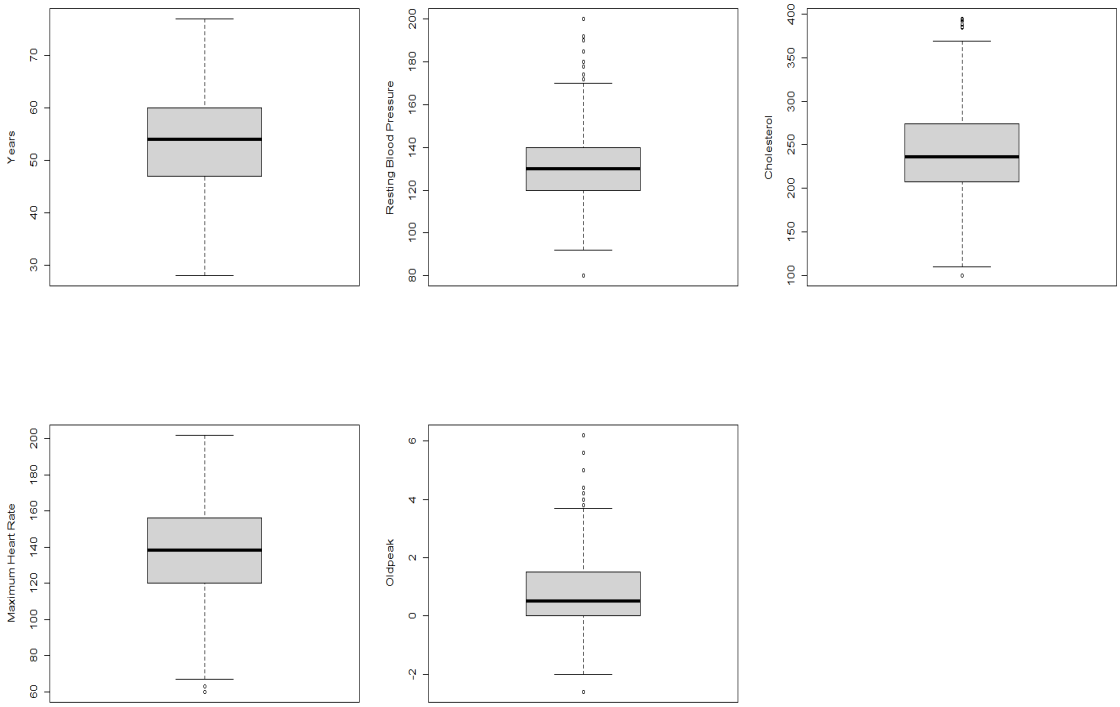

Figure 6: Boxplots of the numerical variables in the *Clean.csv* dataset.

Variable		Count	Frequency
Sex	M	712	79.20 %
	F	187	20.80 %
	Total	**899**	**100. 00 %**
Fasting Blood Sugar	0	692	76.97 %
	1	207	23.03 %
	Total	**899**	**100. 00 %**
ChestPainType	ATA	173	19.13 %
	TA	46	5.12 %
	NAP	203	22.14 %
	ASY	496	53.61 %
	Total	**899**	**100.00 %**
ST_Slope	Flat	448	49.83 %
	Up	389	43.27 %
	Down	62	6.90 %
	Total	**899**	**100.00 %**
ExerciseAngina	1	365	40.60 %
	0	534	59.40 %
	Total	**899**	**100.00 %**
RestingECG	Normal	538	59.84 %
	ST	177	19.69 %
	LVH	184	20.46 %
	Total	**899**	**100.00 %**

Figure 7: Distribution over the classes in the *Clean.csv* dataset.

2.4 Sex

In order to re-balance the observations across the two classes, M and F, I've exploited an **undersampling** technique.
Undersampling refers to a set of techniques designed to balance the class distribution within a classification dataset.

"... undersampling, that consists of reducing the data by eliminating examples belonging to the majority class with the objective of equalizing the number of examples of each class".[3]

Undersampling techniques remove instances from the training set in the majority class (M), in order to re-balance the classes and create an even distribution of the observations across the categories.

To perform undersampling I've used the function "*undersampling*" in the "*caret*" package. This particular function can be used with binary and categorical data, which is not the case for the functions in the "*ROSE*" or "*imbalance*" packages, which require numeric data only. The resulting dataset contains **374** observations, and the two classes are made up by **197** instances each.

2.5 FastingBS

Again, the class imbalance across the two categories can lead to biased results, and thus poor performances.
In this case, the class is biased towards the value "0", which implies that almost 77% of the patients have blood sugar levels lower than 120 mg/dl.
As mentioned before, this is might be a good representation of reality; still, in order to have unbiased results, the algorithm must be trained on sets that are as balanced as possible.

Since the training set has been reduced in the previous step, I've exploited an **oversampling** technique to re-balance the two categories of FastingBS.
Oversampling techniques perform the opposite operation of undersampling: they introduce random duplications of the instances from the minority class[4].

The combination of the two techniques creates a new training set of **610** observations, where the observations over the classes of Sex and FastingBS are evenly distributed[5].
The results of the transformations are shown in Tables 6 and 8.

[3] From: *Learning from Imbalanced Datasets*, 2018.

[4] One must be careful when using such techniques because they introduce some bias as well. Better techniques, such as SMOTE could be implemented, but I've been constrained by the fact that my dataset is mixed, and not numerical.

[5] The distribution over Sex is no longer even after the second transformation, but still balanced.

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak
min	29	94.0	100.0	60.0	-2.6
1st Q.	48.0	120.0	207.0	117.0	0.0
median	55.0	132.0	246.0	139.0	0.7
mean	54.12	135.5	244.0	136.9	0.8462
3rd Q.	60.0	145.0	288.0	157.0	1.5
max	76	200.0	394.0	202.0	6.2
n.obs	**610**	**610**	**610**	**610**	**610**

Table 6: Summary statics for numerical variables in the *train.csv* dataset.

Variable		Count	Frequency
Sex	M	344	56.39 %
	F	266	43.60 %
	Total	**610**	**100. 00 %**
Fasting Blood Sugar	0	305	50.0 %
	1	305	50.0 %
	Total	**610**	**100. 00 %**
ChestPainType	ATA	111	18.20 %
	TA	31	5.08 %
	NAP	134	21.97 %
	ASY	334	54.75 %
	Total	**610**	**100.00 %**
ST_Slope	Flat	314	51.48 %
	Up	247	40.49 %
	Down	49	8.04 %
	Total	**610**	**100.00 %**
ExerciseAngina	1	386	36.72 %
	0	224	63.28 %
	Total	**610**	**100.00 %**
RestingECG	Normal	337	55.24 %
	ST	120	19.67 %
	LVH	153	25.08 %
	Total	**610**	**100.00 %**

Figure 8: Distribution over the classes in the *train.csv* dataset.

3 Supervised Learning

3.1 Decision Trees

Decision trees are classifiers representing partitions of the instance space. Decision trees consist of nodes that form a rooted tree, meaning it is a directed tree with a node called "root" that has no incoming edges. All other nodes have exactly one incoming edge.
A node with outgoing edges is called an internal or test node. All other nodes are called leaves.
In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a certain discrete function of the input attributes values.[6]
In the case of classification trees, there are several criteria to determine the splits, such as *classification error rate*, which is determined by the fraction of the training observations that do not belong to the most common class of the region:

$$E = 1 - \max_k(\hat{p}_{mk}) \tag{2}$$

where p_{mk} is the proportion of observations in the *mth* region, but belong to the *kth* class. However, the classification error rate is not sufficiently sensitive for the intermediate steps, i.e., growing the tree.

Gini index and *cross-entropy* are better indeces for growing trees, since they measure node *purity*, i.e., a small value that indicates whether a node contains predominantly observations from a single class.
I've trained a decision tree on each of the samples I've created: (1) the training sample from the *rf.csv* dataset, on which I've only imputed the missing values of **Cholesterol**; (2) the training sample from the *clean.csv* dataset, on which I've removed the extreme outliers; finally, (3) the *train.csv*, on which I've performed re-sampling over the imbalanced classes.
I've measured the performance of the trees using several performance metrics:
(1) **Accuracy** for binary classification can be defined as:

$$\frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

where TP is the n° of true positives (where TP is HD, in our case), TN is the n° of true negatives (Normal), FP is the n° of false positive, and FN is the n° false negatives.
(2) **Sensitivity** can be defined as:

$$\frac{TP}{TP + TN} \tag{4}$$

i.e., as the fraction of rightfully-predicted HDs, over the total predictions of HD.
(3) **Sensibility**:

$$\frac{TN}{TN + TP} \tag{5}$$

i.e., as the fraction of rightfully-predicted absence of HDs, over the total predictions of "Normal".
In case of imbalanced datasets, it's desirable to check *precision* and *recall*.
(4) **Precision** can be defined as:

$$\frac{TP}{TP + FP} \tag{6}$$

i.e., as the fraction of rightfully-predicted HDs, when HD is true.
(5) **Recall**:

$$\frac{TN}{TN + FN} \tag{7}$$

[6] *Decision trees*, Lior Rokach & Oded Maimon.

i.e., as the fraction of rightfully-predicted absence of HDs, when lack of HD is true.
(6) **F1** score is a performance metrics that averages the results of precision and recall. Also known as *harmonic mean*.
The results on the training data are represented by Figure 9.
In particular, we can observe that St-Slope and ChestPainType determine the biggest splits across the observations.
Overall, the best performing tree is the one built on the **clean[train,]** set, which is represented by Figure 10.
Among the 11 variables, 5 only have been used to build the tree: ChestPainType, Cholesterol, MaxHR, Oldpeak, and ST-Slope.

3.2 Random Forests

Although decision trees have several advantages, i.e., they have accommodating conditions, can be used for regression and classification indistinctly, and they are easily interpretable, they also lack predictive accuracy.
Therefore, other techniques have been created in order to improve their performance, such as random forests (and bagging, which can be considered as a peculiar case of random forests).
Random forests are ensemble methods which improve the predictive accuracy of the decision trees by modelling several trees (in our examination 500 trees have been created for each of the six models we'll discuss later) on repeated samples from the training sample, and average their results in one single, final, prediction.
With respect to bagging, in which all variables are used to grow the trees, each time there's a split, random trees will randomly select a subset of m of the p predictors.
That's why bagging can be considered a peculiar type of random forests.

As I did previously, I've trained random forests and bagging trees on all three datasets, and I've obtained the results showcased in Figure 11 and 14.

The overall performances have been improved with both classifiers; in particular, the models trained on *train.csv* have obtained highest scores across all performance measures.
To the previous performance metrics I have added the Out-Of-Bag error rate. Differently from the others, this metric is not observed on the test sample, but it is obtained directly from the splits within the training sample.
The OOB error is the estimated on the *out-of-bag* observations, i.e., the ones that have not been used to fit the random tree.
We can observe that for the *train* dataset, the error is quite small.
Finally, Figure 17 is portraying the variable importance resulting from running a random forest on the *train* dataset. Variable importance is defined by means of mean decrease in **Gini coefficient**.
This coefficient measures the contribution of each variable to the purity of nodes and leaves in the resulting random forest. The higher the value of mean decrease Gini score, the higher the importance of the variable in the model.
As observed previously with decision trees, ST-Slope and ChestPainType are important in determining the splits of the trees; however, this could result due to the relative class imbalance in the two variables.

Training Set	Test Set	Accuracy	Sensitivity	Specificity	Precision	Recall	F1
train	rf	0.8451	0.8600	0.8268	0.8789	0.8304	0.8550
clean[train,]	clean[-train,]	0.8333	0.9175	0.7349	0.8018	0.9175	0.856
rf[train,]	rf[-train,]	0.8207	0.7921	0.8554	0.8696	0.7921	0.8296

Figure 9: Measuring performance for the three decision trees.

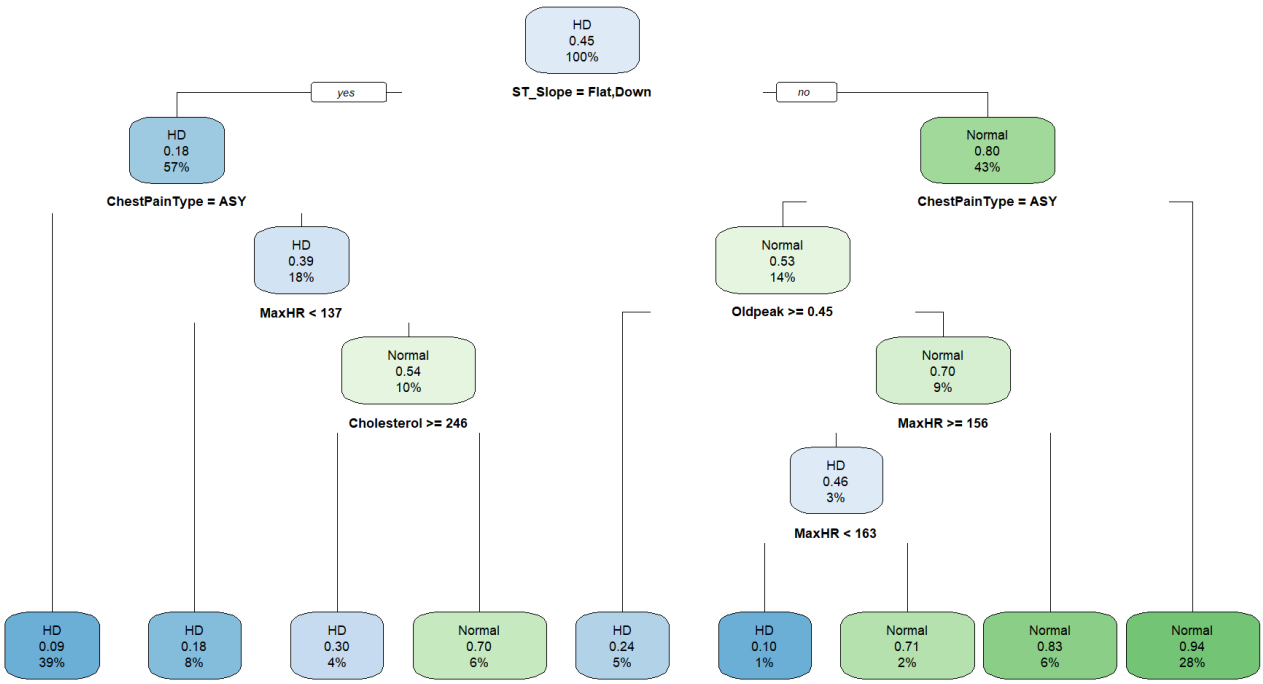

Figure 10: Best performing decision tree.

Training Set	Test Set	Accuracy	Sensitivity	Specificity	Precision	Recall	F1
train	rf	0.8975	0.8836	0.9146	0.9275	0.8836	0.9051
clean[train1,]	clean[-train1,]	0.8389	0.8557	0.8193	0.8389	0.8469	0.8513
rf[train1,]	rf[-train1,]	0.8478	0.8714	0.8193	0.8544	0.8713	0.8627

Figure 11: Performance metrics using bagging, for each training set.

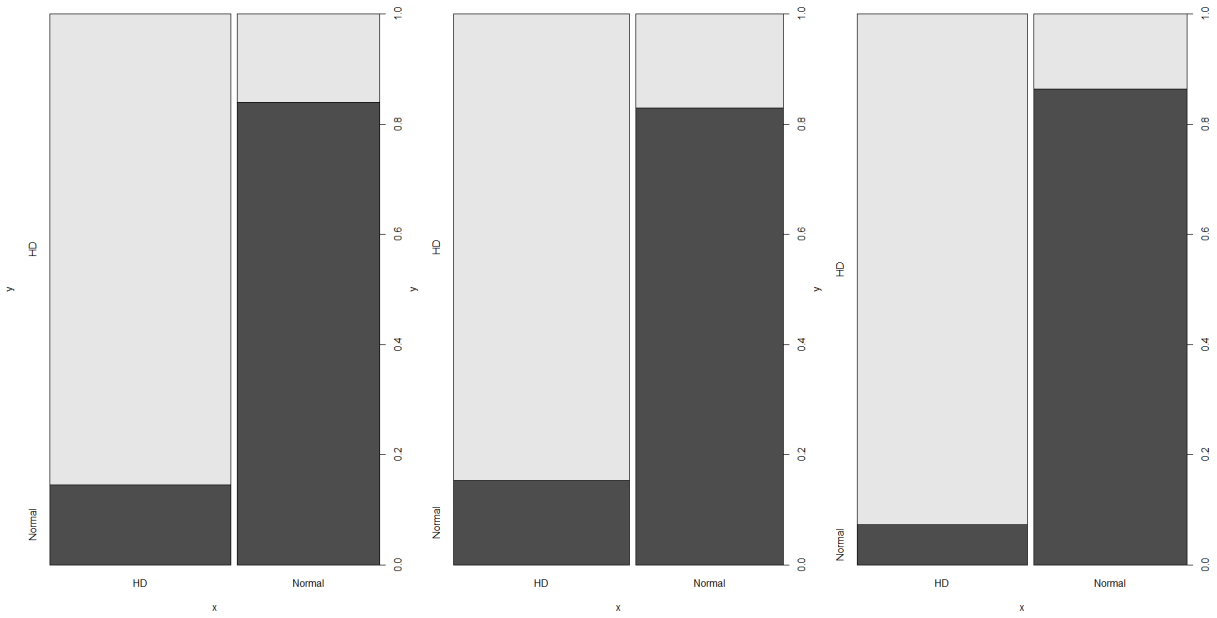

Figure 12: Accuracy for each of the three bagging trees: (1) on *rf* dataset, 0.8478; (2) on *clean*, 0.8389; (3) on *train*, 0.8975.

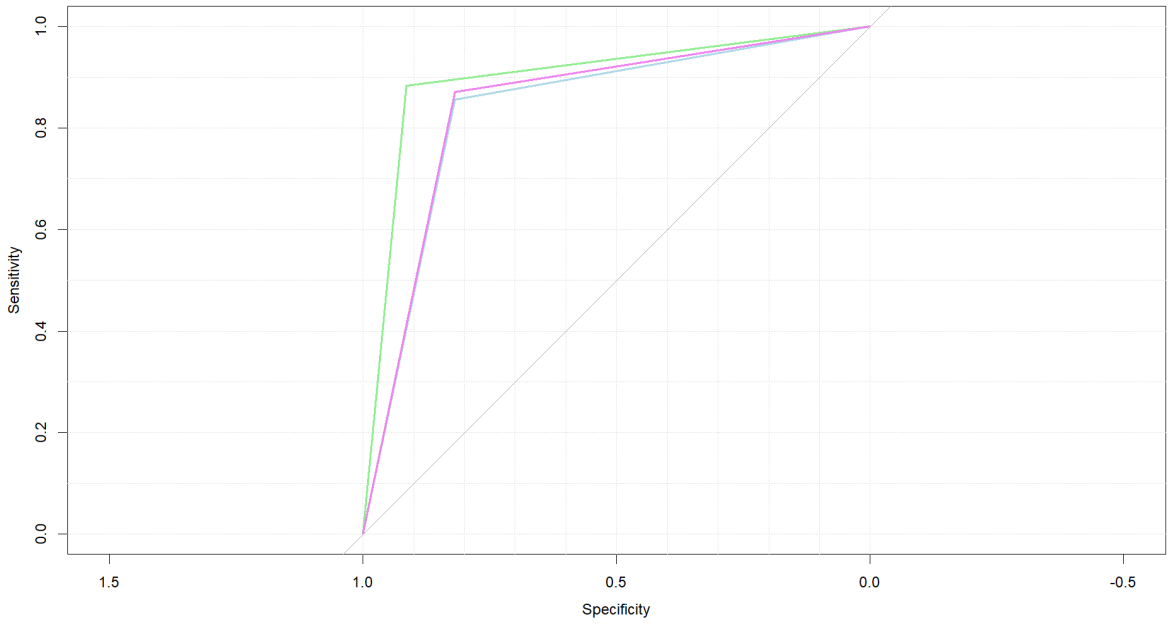

Figure 13: ROC curves for bagging trees. The best performance is obtained on the *train* dataset.

Training Set	Test Set	n. of variables at each split	Accuracy	Sensitivity	Specificity	Precision	Recall	F1	OOB error
train	rf	3	0.9138	0.9172	0.9098	0.9263	0.9172	0.9217	7.38%
clean[train2,]	clean[-train2,]	3	0.8778	0.9263	0.8235	0.8544	0.9263	0.8889	15.44%
rf[train2,]	rf[-train2,]	3	0.8578	0.9109	0.7952	0.8440	0.9109	0.8762	14.19%

Figure 14: Metrics to evaluate the performance of random trees on the three datasets.

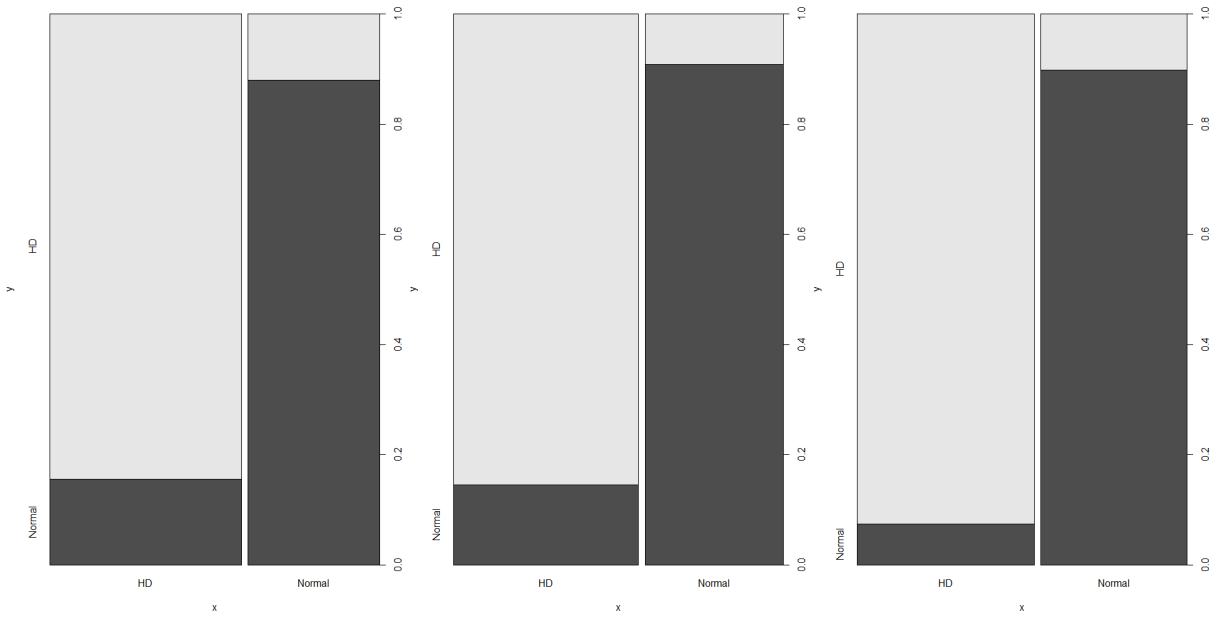

Figure 15: Accuracy of the three random forests trained. (1) on *rf*, 0.8578; (2) on *clean*, 0.8778; (3) on *train*, 0.9138.

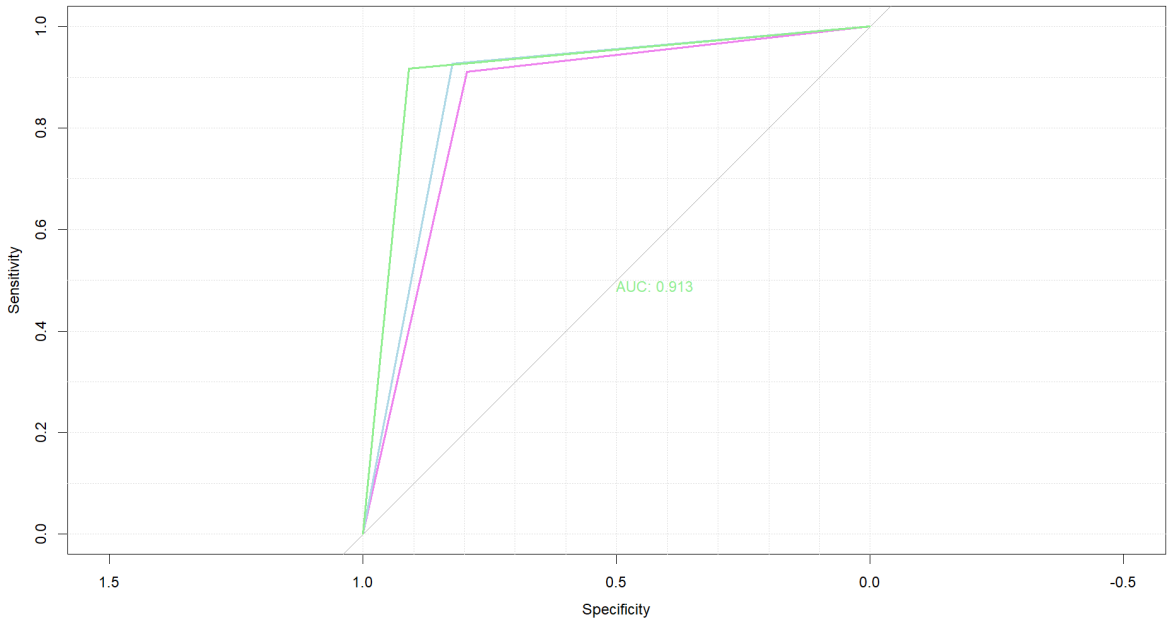

Figure 16: ROC curves for random trees. The best performance is obtained on the *train* dataset.

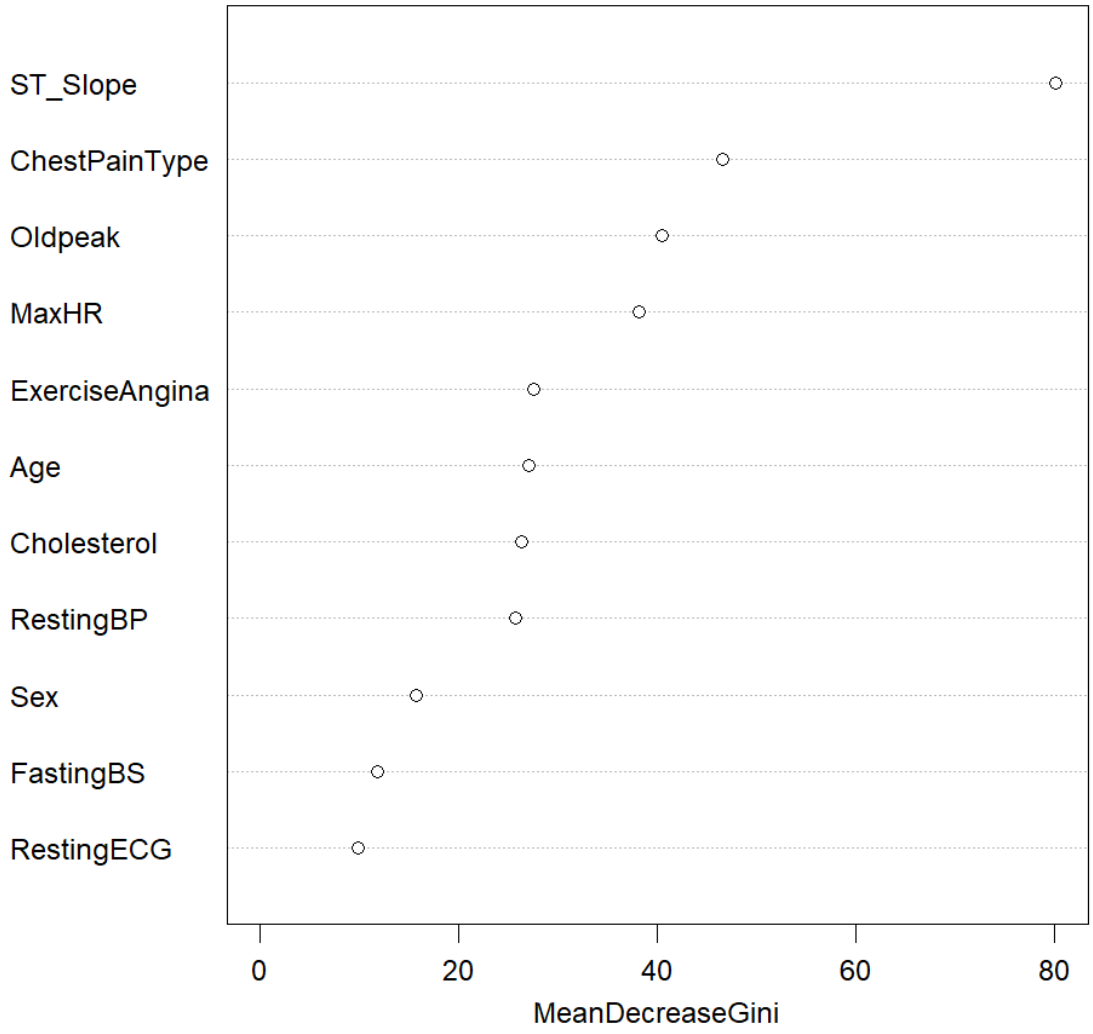

Figure 17: Feature importance for random forest on *train*.

4 Unsupervised Learning

4.1 Data pre-processing

Clustering algorithms, such as **Hierarchical Clustering**, exploit several types of *distances* to group data. When the variables are on different measure units, it's best to scale the dataset, to avoid distortions caused by the different scales.
In this dataset, the numeric variables are all expressed in different measure units, or on completely different scales. Therefore, before applying the clustering algorithm, I shall standardize my data.
The resulting variables have **mean** = 0 and **standard deviation** = 1:

	Age	**RestingBP**	**Cholesterol**	**MaxHR**	**Oldpeak**
min	-2.713	-2.910	-2.816	-3.01	-3.268
1st Q.	-0.696	-0.697	-0.672	-0.656	-0.828
median	0.05	-0.144	-0.104	-0.656	-0.828
mean	0.00	0.00	0.00	0.00	0.00
3rd Q.	0.684	0.409	0.654	0.755	0579
max	2.489	3.728	0.654	2.576	4.989

Table 7: Example of summary statics for numerical variables after scaling, for *clean.csv* dataset.

4.2 Hierarchical Clustering

Hierarchical clustering is an alternative approach to K-mean clustering, which has the advantage of not requiring the choice of the number of clusters in advance.
In hierarchical clustering, the "tree" (*dendogram*) is generally built "bottom-up", i.e., starting from the leaves, it reaches the root node by clustering leaves, i.e., single observations, and groups of them.
The algorithm exploits the concept of "*distances*" in order to group the clusters.

Generally, computing the distances across **11** dimensions would require the observations to be only numerical (e.g., when computing the *Euclidean distance*); however, with hierarchical clustering, the concept of distances has been generalized to qualitative variables, too.
To apply such algorithm, we can exploit the so-called **Gower distance**.
The Gower distance is computed as the average of partial dissimilarities across instances.
The Gower index can be computed in the following way:

$$S_{i,j} = \frac{\sum_{s=1}^{p} z_{i,j_s}}{\sum_{s=1}^{p} w_{i,j_s}} \tag{8}$$

where $w_{i,j_s} = 1$ when the comparison between two observations is feasible; $w_{i,j_s} = 0$, otherwise.
For qualitative variables, $z_{i,j_s} = 1$ when the observations fall into the same class.
For quantitative variables, or ordinal degrees,

$$z_{i,j_s} = 1 - \frac{|x_{i,s} - x_{j,s}|}{K_s} \tag{9}$$

where K_s is the variation change of the variable X_s.
The Gower distance has domain between [0,1]:

(1) $S_{i,j} = 1$, when all the observations fit into the same classes, and have equal quantitative value (**perfect similarity**).
(2) $S_{i,j} = 0$, when no pair of observations fits into the same classes, and for each quantitative variables the observations have opposite values (**maximum diversity**).

For both *clean.csv* and *train.csv* datasets, I've created two matrices of Gower distances. Once the index has been computed (using the *dice* function, from the *cluster* package), a matrix has been created for each pair of values, and for each dataset.

(1) The first resulting matrix has 403 651 dissimilarities, with minimum and maximum values of 0.003501, and 0.775970 respectively.
To check the results, we can output the least and most dissimilar pairs in the matrix:

```
    Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR ExerciseAngina Oldpeak
157  42   M           ATA       120         196         0     Normal   150              0       0
141  42   M           ATA       120         198         0     Normal   155              0       0
    ST_Slope HeartDisease
157       Up       Normal
141       Up       Normal
```

Figure 18: Least dissimilar pair of observations from *clean.csv*.

```
    Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR ExerciseAngina Oldpeak
715  56   F           ASY       200         288         1         LVH   133              1       4
281  36   M           ATA       120         166         0     Normal   180              0       0
    ST_Slope HeartDisease
715     Down           HD
281       Up       Normal
```

Figure 19: Most dissimilar pair of observations from *clean.csv*.

(2) The second resulting matrix has 185 745 dissimilarities, with minimum and maximum values of 0.0000 (maximum diversity) and 0.7864, respectively. Again, we can check the results by computing the least:

```
    Age Sex ChestPainType RestingBP Cholesterol FastingBS RestingECG MaxHR ExerciseAngina Oldpeak
142  46   F           ASY       138         243         0         LVH   152              1     0.0
78   45   F           ASY       138         236         0         LVH   152              1     0.2
    ST_Slope HeartDisease
142     Flat       Normal
78      Flat       Normal
```

Figure 20: Least dissimilar pairs of obs from *train.csv*.

Before implementing the clustering algorithm, we shall select the *linkage* method.
The linkage method defines the way the model determine the splits among clusters.
Among the several methods one could implement, I chose the *complete* and *average* linkage methods.
The first computes all pairwise dissimilarities and records the largest one; the latter computes all pairwise dissimilarities and records their average.
The resulting partitions are showcased by Figures 22 and 23 for the *clean.csv* dataset, and by Figure 24 and 25 for the *train.csv* dataset.
First, we can notice that balanced datasets result in more evenly-defined and better-separated clusters.
Then, the heights of the partitions are greater with complete linkage, with respect to average linkage, and the first clusters are less shrinked.

Moverover, I use the *average silhouette method*, with *pam* function, to obtain the optimal number of clusters, which is in both cases equal to **two**.

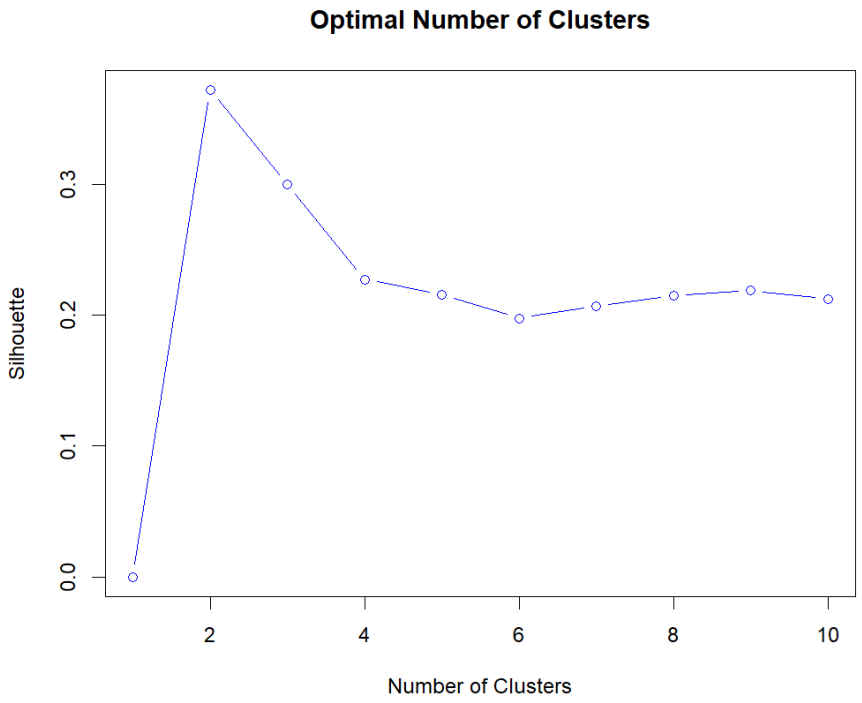

Figure 21: Optimal number of clusters.

Finally, we can plot the dendograms by cluster, and observe the partition of the instances in Table 8.
The clusters in Figures 28 and 29, show how the data is better-split into the balanced dataset, with respect to the unbalanced one.
Moreover, there's a slight improvement of the total variance explained by the first two principal components.

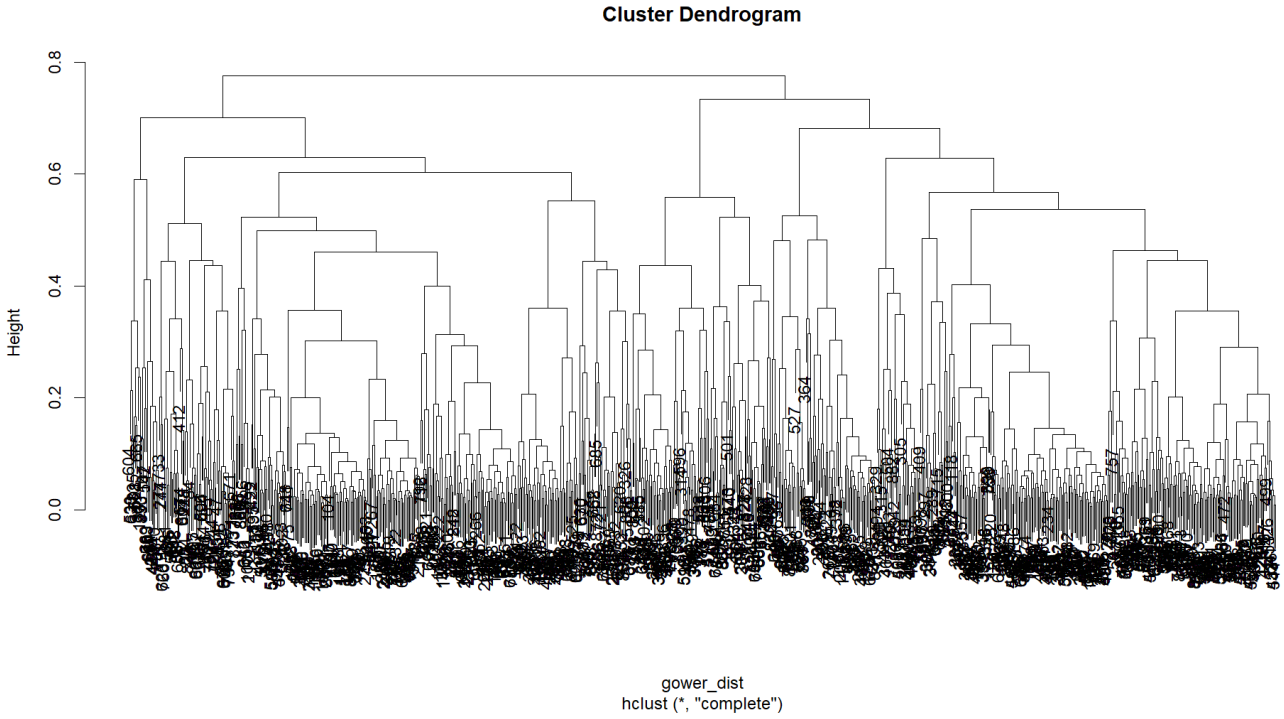

Figure 22: Hierarchical clustering grown with *complete* linkage, on *clean.csv* data.

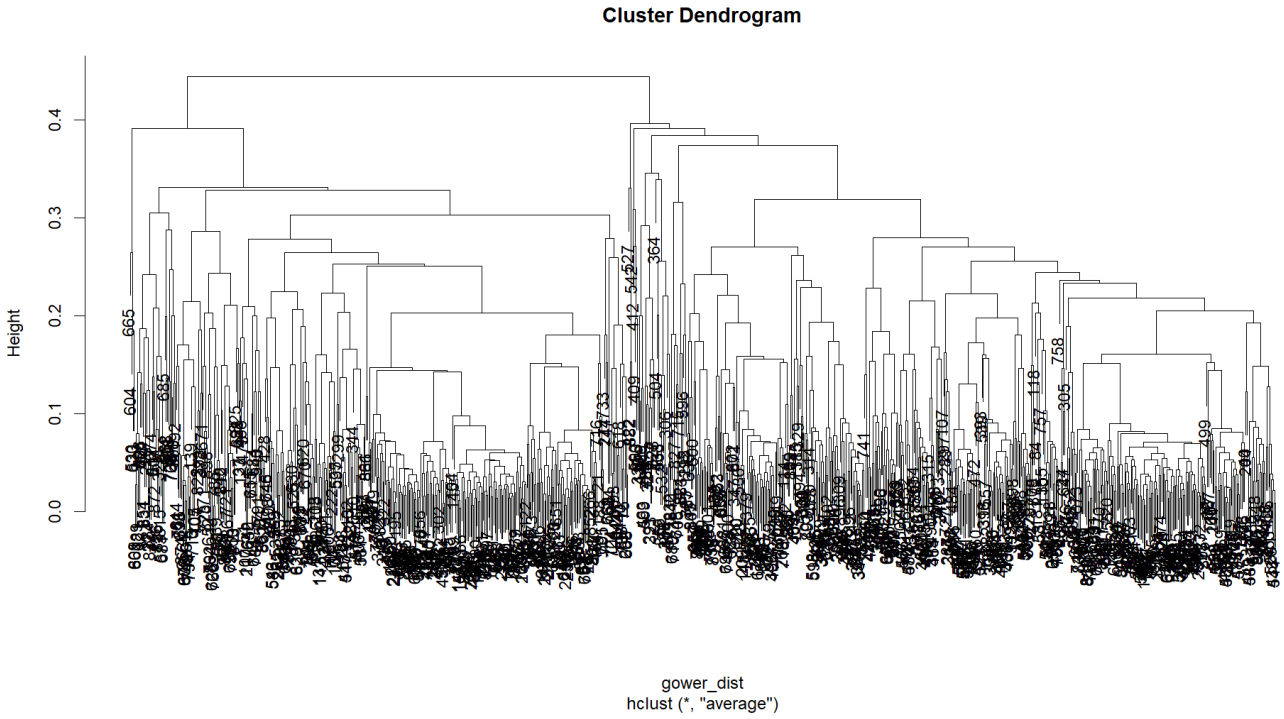

Figure 23: Hierarchical clustering grown with *average* linkage, on *clean.csv*.

groups	HD	Normal
1	12	382
2	483	22

Table 8: *Clean.csv* data partitioning in the two clusters.

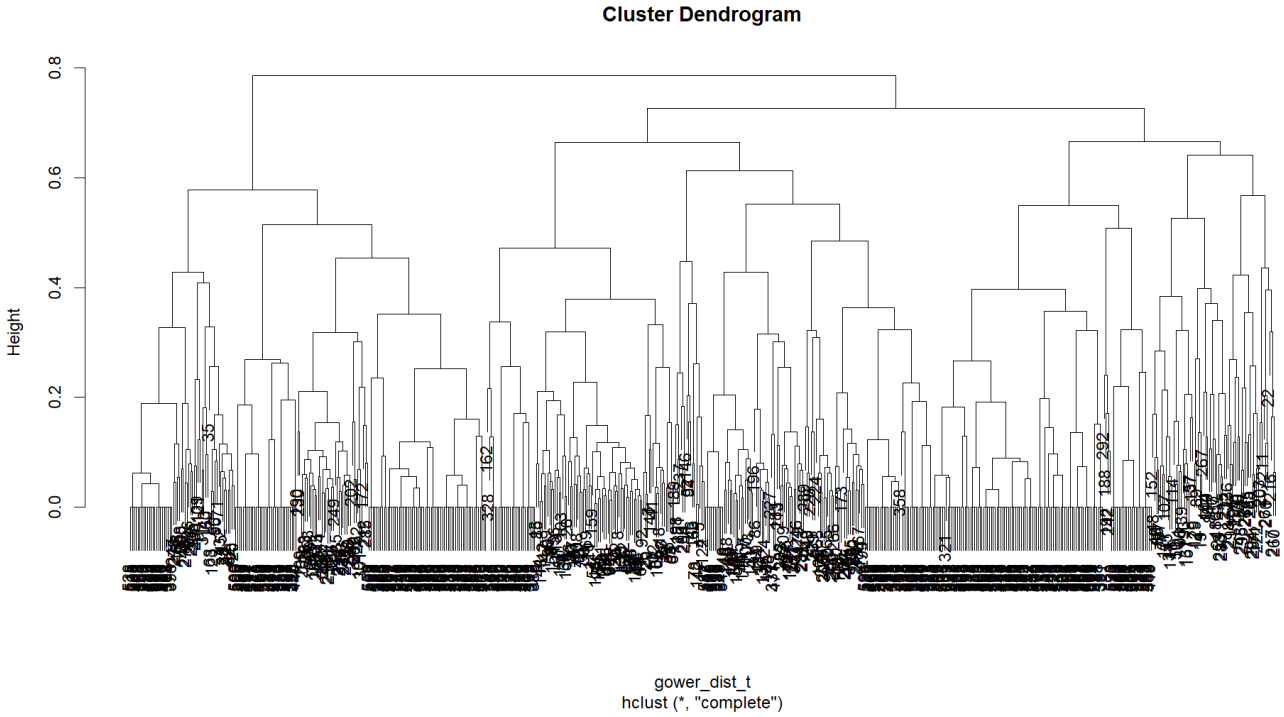

Figure 24: Hierarchical clustering grown with *complete* linkage, on *train.csv* data.

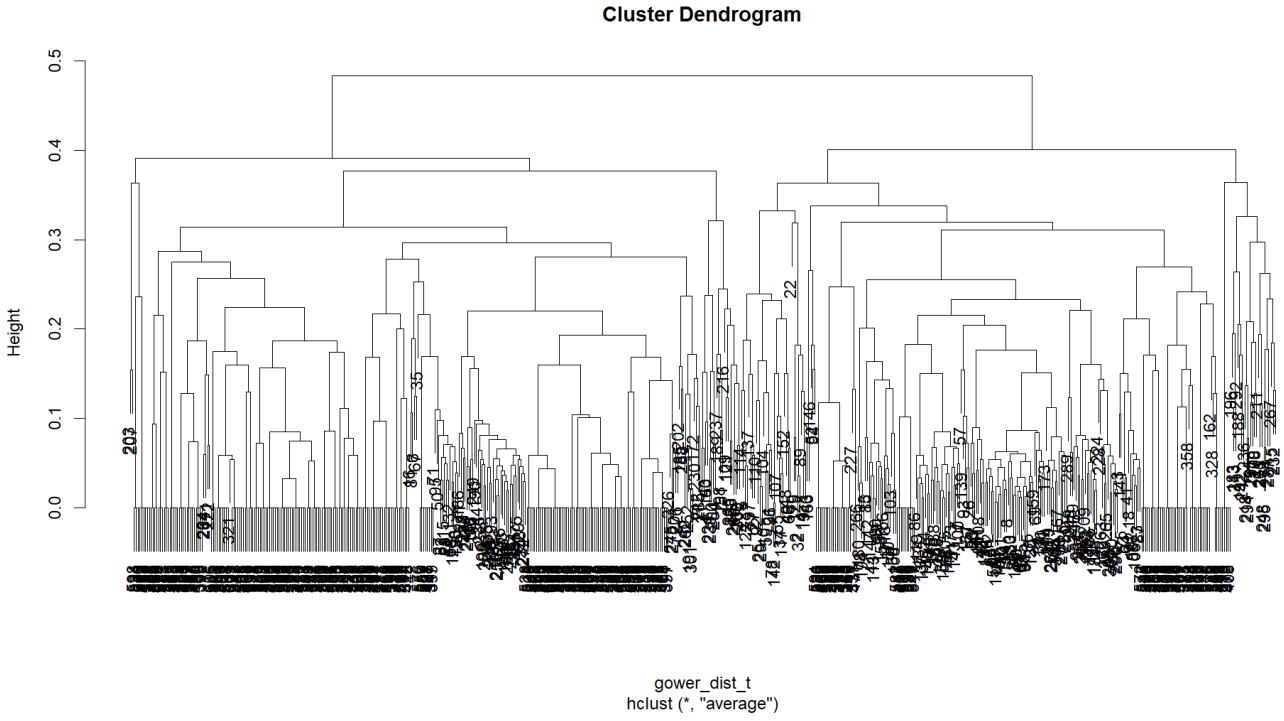

Figure 25: Hierarchical clustering grown with *average* linkage, on *train.csv*.

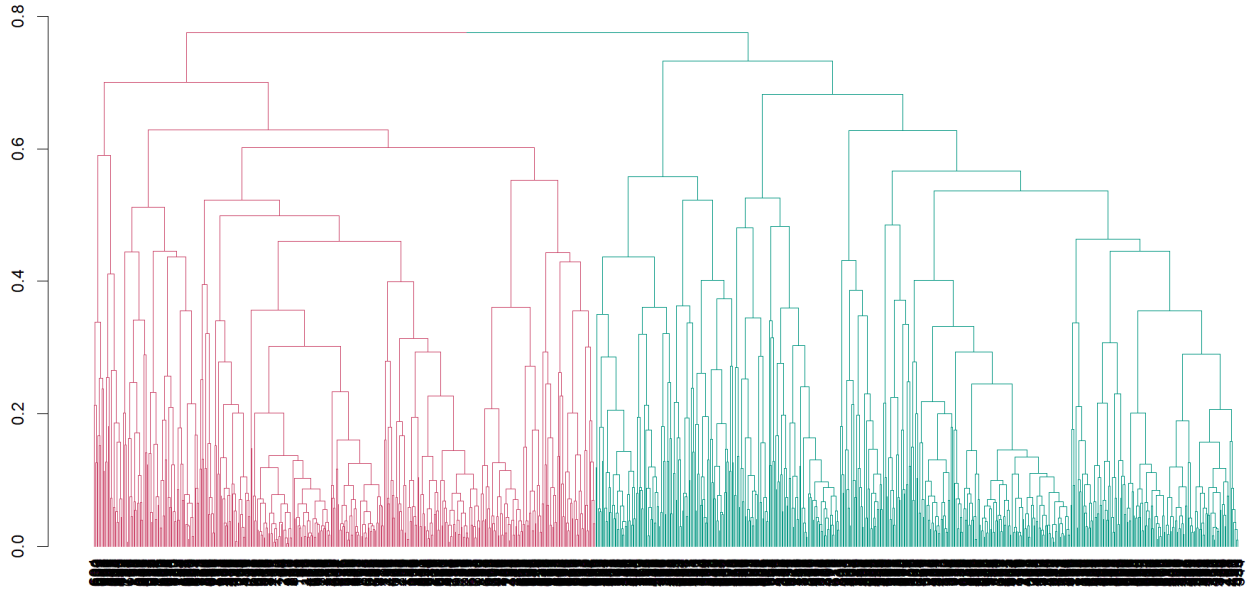

Figure 26: Alternative representation of the dendograms, split into two main clusters, using complete linkage.

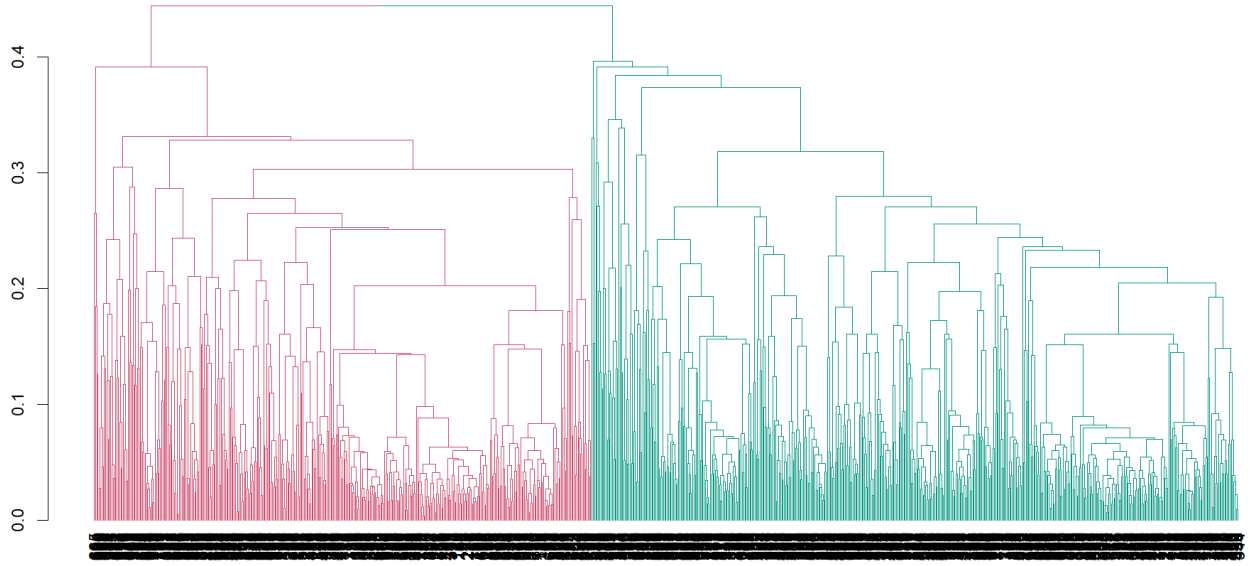

Figure 27: Alternative representation of the dendograms, split into two main clusters, using complete average.

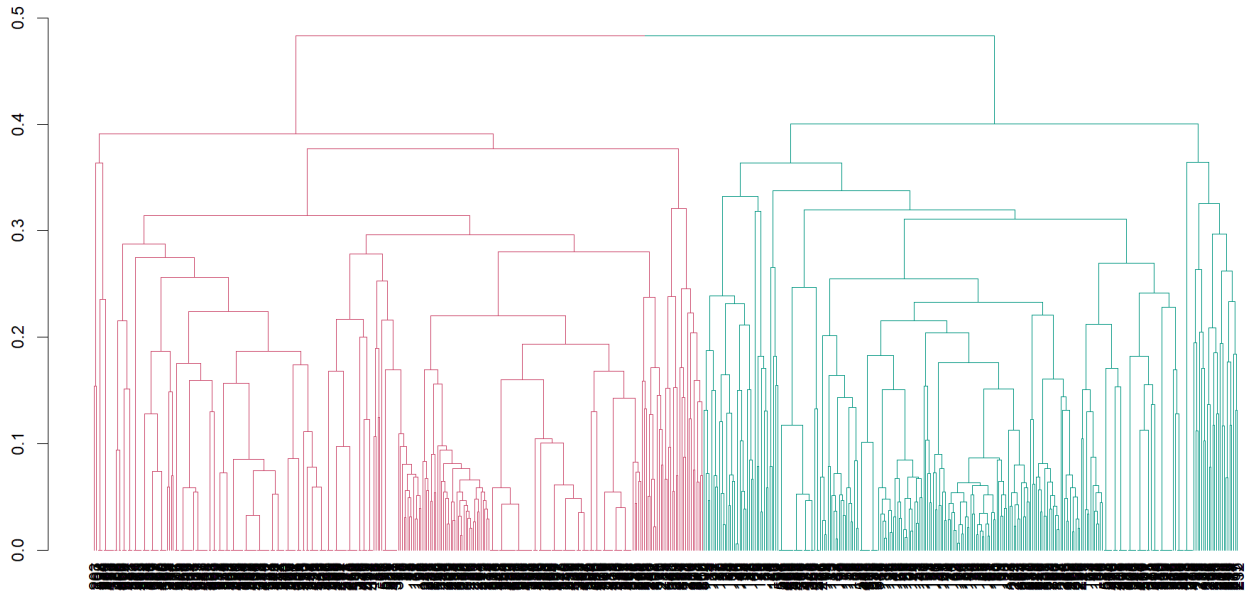

Figure 28: Alternative representation of the dendograms, split into the two main clusters, using *complete* linkage, on *train.csv*

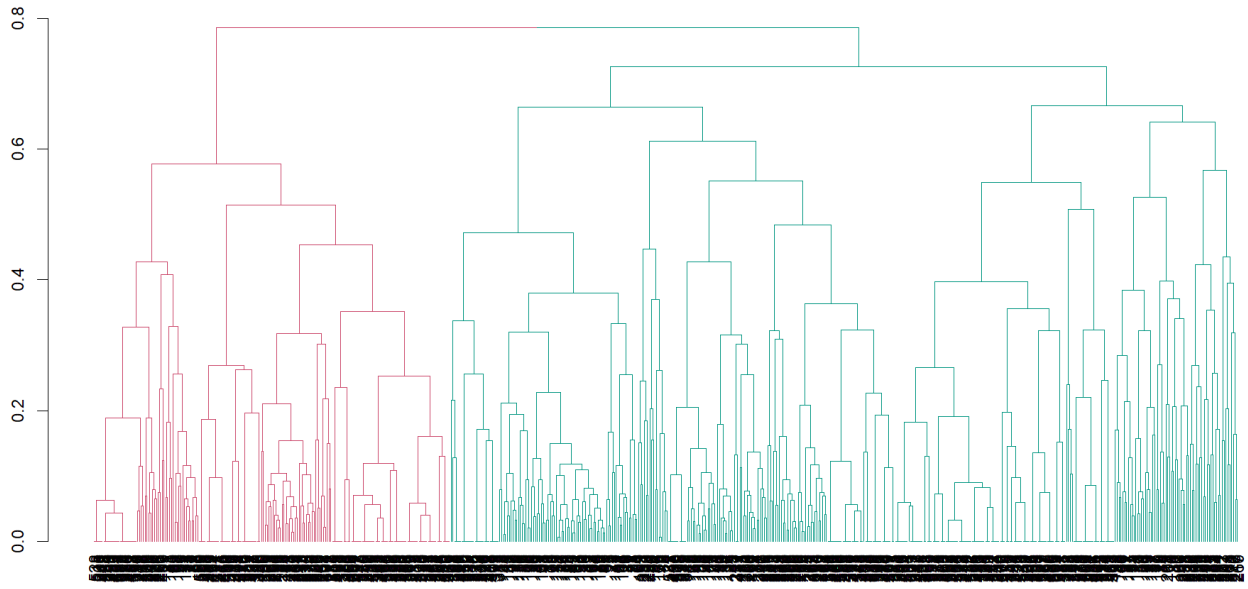

Figure 29: Alternative representation of the dendograms, split into the two main clusters, using *average* linkage, on *train.csv*.

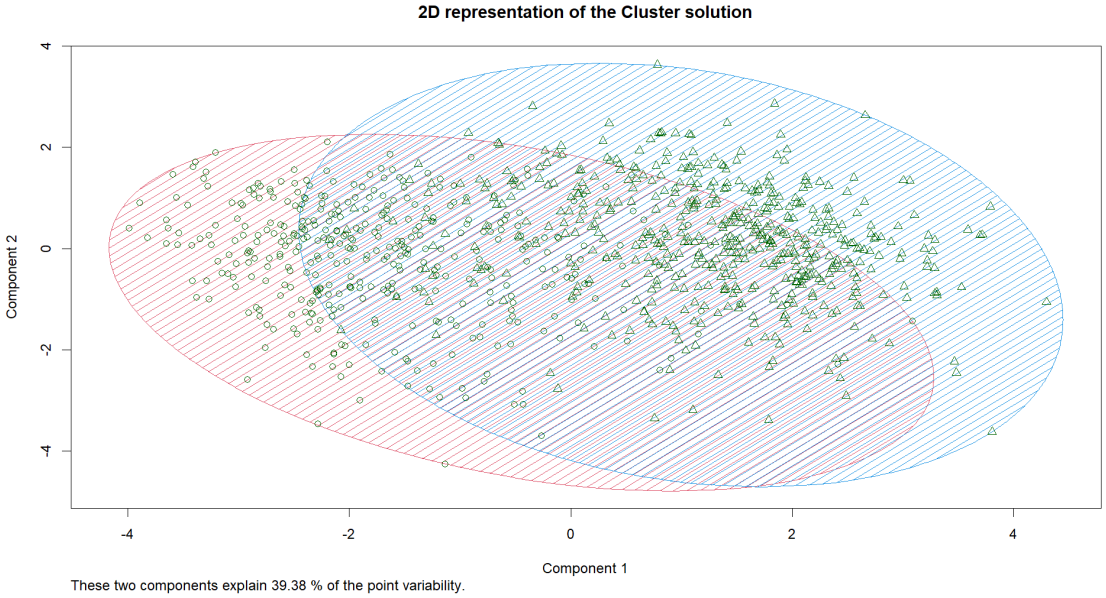

Figure 30: Representation on 2 dimensions of the clusters.

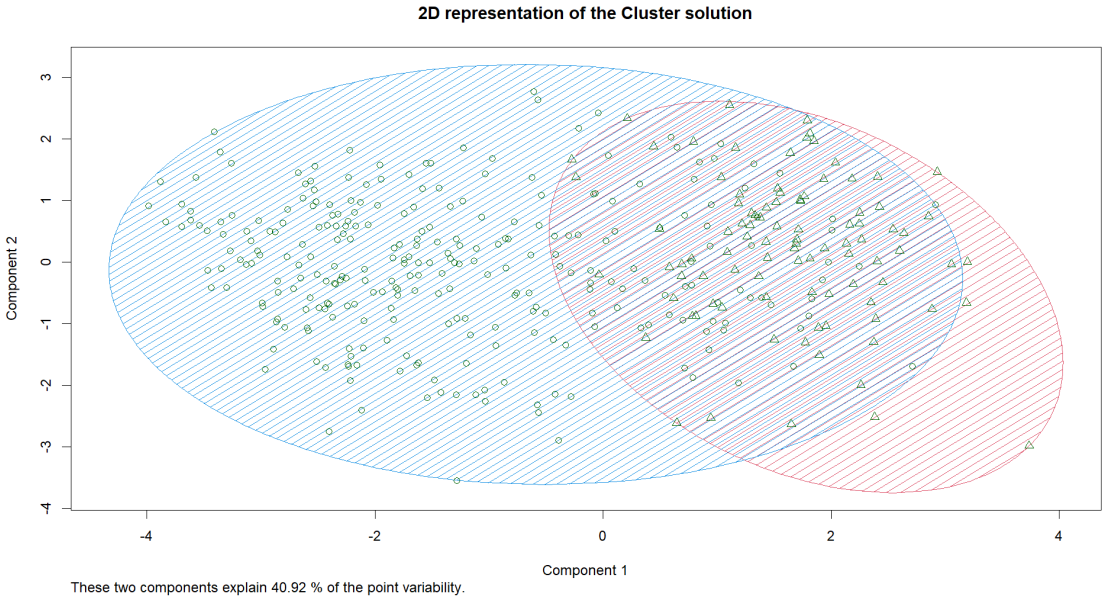

Figure 31: Representation on 2 dimensions of the clusters, on *train.csv*.