Politecnico di Torino

HCV DATASET ANALYSIS

PROJECT FOR THE COURSE “MATHEMATICS IN MACHINE LEARNING”

A.A 2021/2022

CORRADO NAVILLI S291251

s291251@studenti.polito.it

PROFESSORS:

FRANCESCO VACCARINO

MAURO GASPARINI

CONTENTS

1 INTRODUCTION

Healthcare is a sector that can benefit from the use of machine learning as a tool for helping and reducing the burden of professionals and improving the patients' quality of life. A crucial step in a disease management is certainly the diagnostic step: diagnosis can be both difficult for clinicians and painful for the patients, especially for those cases in which invasive examinations are needed in order to reach a diagnosis.
HCV, acronym for *Hepatitis Virus – type C*, is a virus that specifically targets the liver of individuals and that can lead to chronic diseases (in around 85% of the cases [1]) with an increasing damage of the organ (leading at the very end to the death of the individual). Such virus has a parenteral transmission, meaning that can be transmitted through direct contact with infected blood or infected instrumentations (such as the ones commonly used for tattoos and aesthetical treatments, not properly sterilized). No vaccine is currently available for this virus.

Since if not properly treated with drugs the infection can progressively damage the liver, it is also important to reach a diagnosis as soon as possible. Machine Learning algorithms able to detect suspicious cases from blood tests may facilitate this task and speed up the process.

2 DATA DESCRIPTION

The HCV dataset (available on UCI Datasets at https://archive.ics.uci.edu/ml/datasets/HCV+data) contains 615 (not duplicated) records corresponding to as many different patients. For each patient, some personal data are provided along with the results from blood tests (features #5-14):

- **Unnamed feature:** (numeric) – progressive number, ID of the patient
- **Category:** (target class) – values: {'0=Blood Donor', '0s=suspect Blood Donor', '1=Hepatitis', '2=Fibrosis', '3=Cirrhosis'}
- **Age:** (numeric)
- **Sex:** (categorical) – {"Male", "Female"}
- **ALB:** (continuous) - amount of albumin in patient's blood
- **ALP:** (continuous) - amount of alkaline phosphatase in patient's blood
- **ALT:** (continuous) - amount of alanine transaminase in patient's blood
- **AST:** (continuous) - amount of aspartate aminotransferase in patient's blood
- **BIL:** (continuous) - amount of bilirubin in patient's blood
- **CHE:** (continuous) - amount of cholinesterase in patient's blood
- **CHOL:** (continuous) - amount of cholesterol in patient's blood
- **CREA:** (continuous) - amount of creatine in patient's blood
- **GGT:** (continuous) - amount of gamma-glutamyl transferase in patient's blood
- **PROT:** (continuous) - amount of protein in patient's blood

Note that the unit measure for features from #5 to #14 is not available.

There are few missing values that will need some processing afterwards:

Unnamed: 0 :0
Category : 0
Age : 0
Sex : 0
ALB : 1
ALP : 18
ALT : 1
AST : 0
BIL : 0
CHE : 0
CHOL : 10
CREA : 0
GGT : 0
PROT : 1

The target variable *Category* has 4 different values with a progressive numeration corresponding to an increasing degree of severity of the disease, leading to a multiclass classification problem. This works wants to approach a binary classification problem; therefore, such values will be remapped according to the following:

- Class “0=Blood Donor” → “Healthy”
- Classes “1=Hepatitis”, “2=Fibrosis”, “3=Cirrhosis” → “Disease”

The class “0s=suspect Blood Donor” has been intentionally left outside this mapping. This is because the “ground truth label” associated to these records has a high degree of uncertainty that could probably affect negatively the prediction models. These records will therefore be discarded from the analysis and will be eventually fed to the models to see the assigned label.

3 DATA EXPLORATION

First of all, the *Unnamed feature* represent a repetition of the index of the dataset (starting from 1 instead of from 0) and for this reason such column has been dropped.
As a second step, the 7 records (1.14% of the total) with label “0s=suspect Blood Donor” will be dropped from the analysis.

Class balance

The classes are highly unbalanced, even after grouping the data into only two macro-categories, the “Healthy” and the “Ill”. Indeed, the ratio Healthy:Ill is a little over 7:1 (533 versus 75 data points).

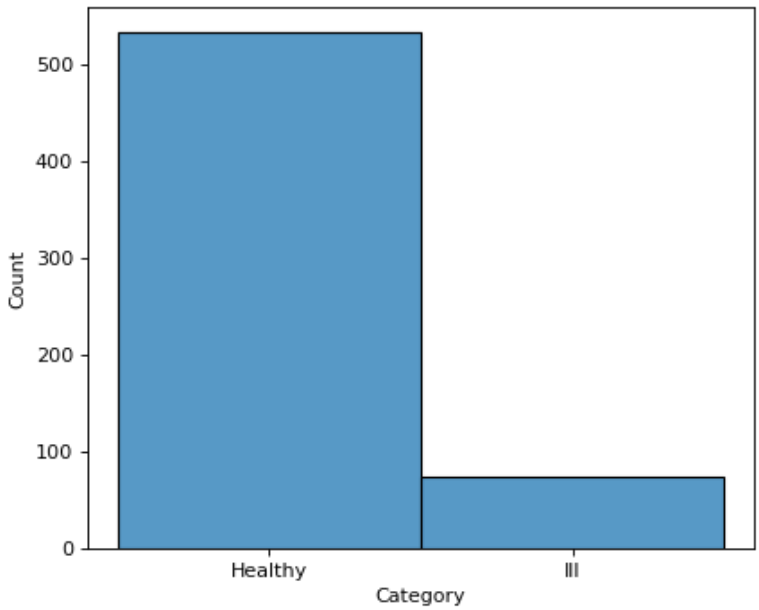


Fig 1. Representation of the imbalance of the classes in absolute values.

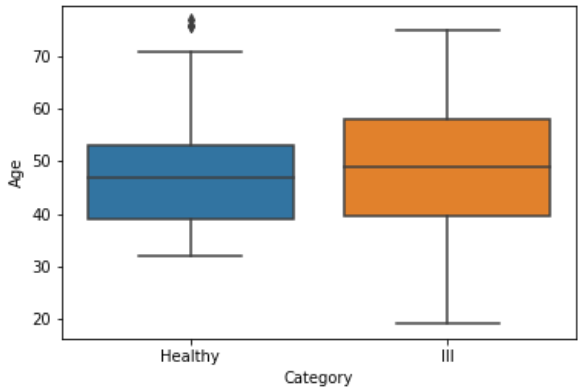
It is very important to keep trace of this imbalance because it may deeply affect the training of the model and the evaluation.
This is a typical situation when dealing with clinical data, since (hopefully) severe diseases do not have a high diffusion in the population and a dataset with too many data entries belonging to the ill class in proportion to the healthy class would not be representative of the real data distribution.
In the next phases this difference between the proportion will be handled and explained in details.

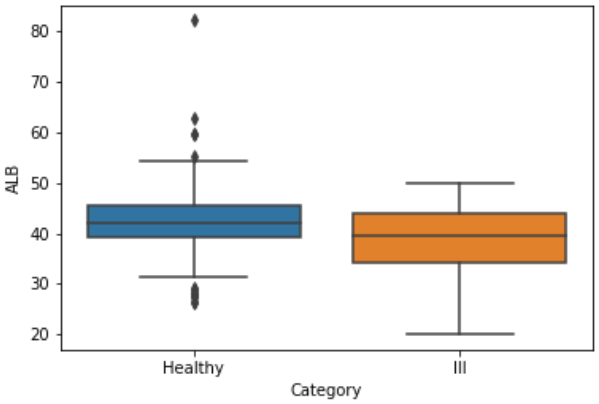
3.1 UNIVARIATE AND BIVARIATE ANALYSIS

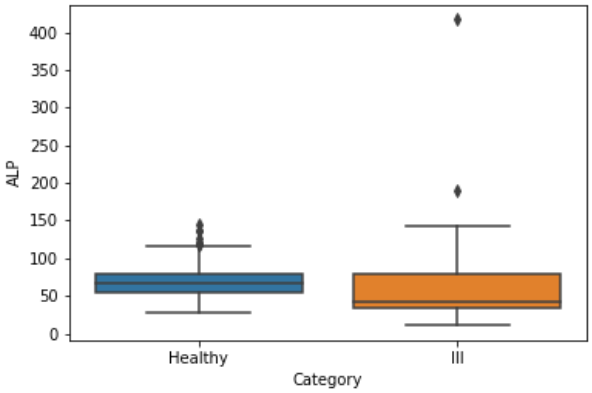
When dealing with multiple features it may be helpful to analyze the distribution of the data by considering one feature at time. It is impossible to visualize the distribution of data over a high multi-dimensional feature space, but focusing on a single dimension allows retrieving some useful information thanks to some easy plotting of the data.

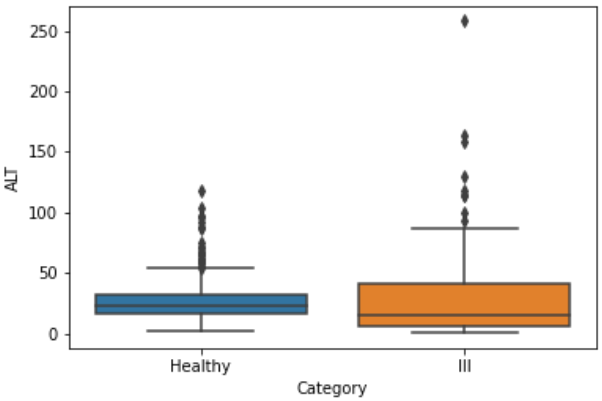
Univariate analysis through boxplots

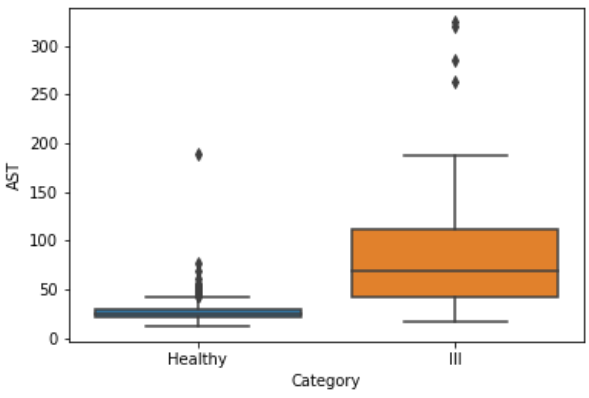
The boxplots are a good choice for the analysis of numerical features since they highlight the quantiles and the median for the data distribution allowing a first qualitative analysis, and help detects some anomalies that may represent interesting outliers or for example poor quality data points such as measurement mistakes.
The following images show such plots.

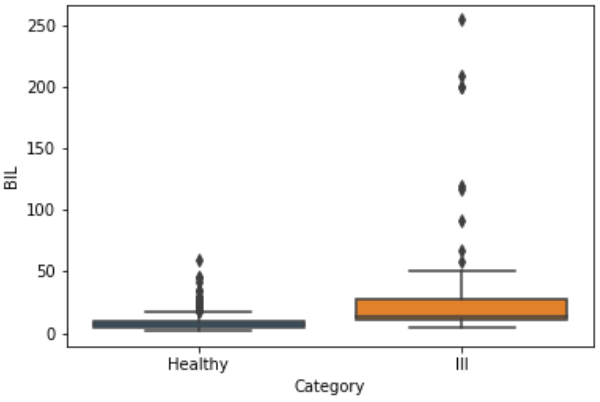












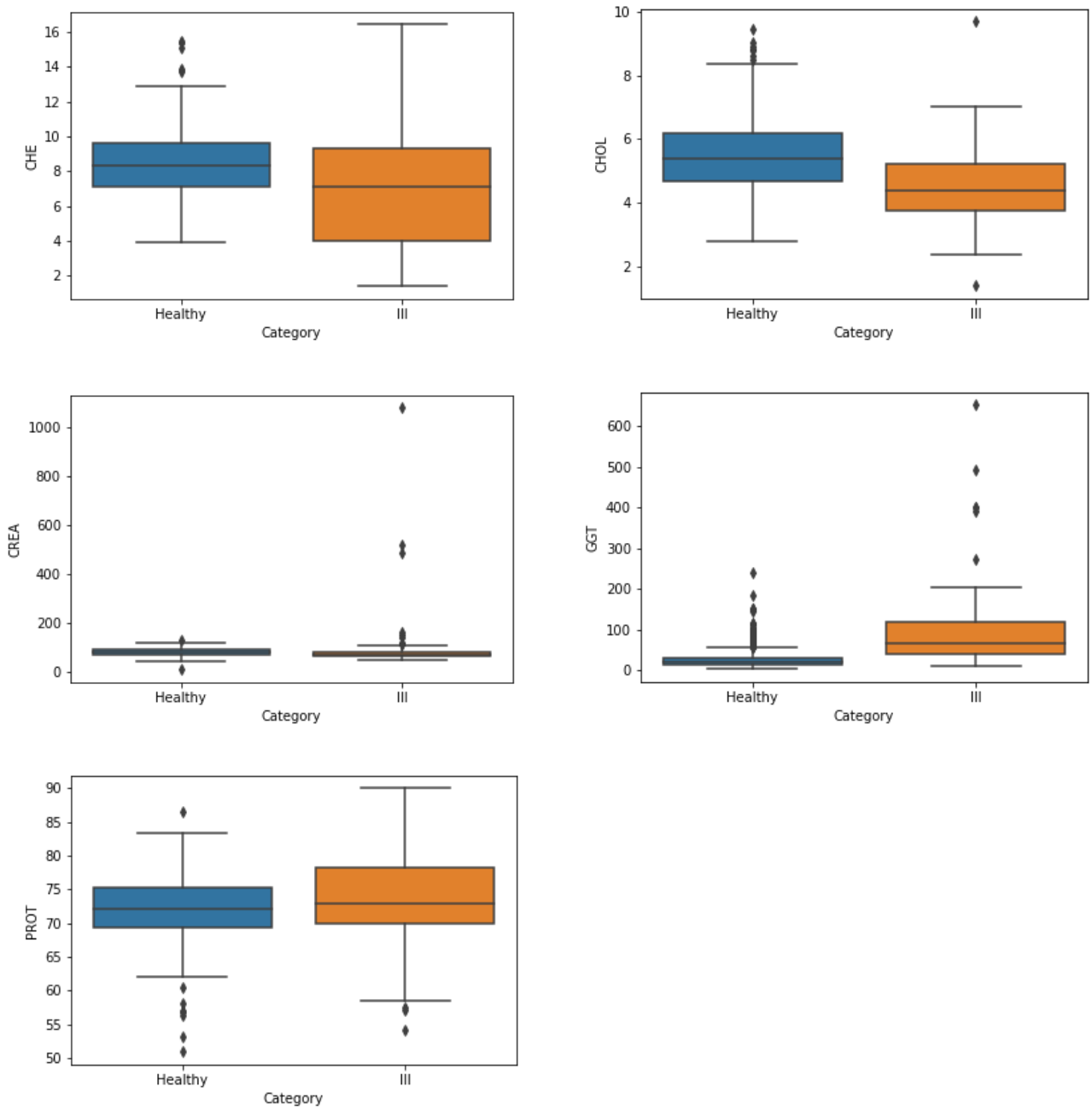


Fig 2. Boxplots of the numerical features.

Now that the boxplots are available, it is possible to make some qualitative analysis.
It looks like that the univariate distributions are similar between the two groups, for almost all the features: the two corresponding boxplots overlap and there is no a significant disjunction that would allow detecting immediately a discriminative feature. There are some peculiarities, though:

1. AST: The points belonging to class ill look drifted towards higher values. Also the amplitude of the distribution is higher (as can be retrieved by the height of the boxplot). Data belonging to Healthy class seem having values concentrated around 25, while the others lie on a larger range.
2. GGT: like AST, points belonging to Ill class seem drifted towards higher values, compared to the Healthy class.
3. CHOL and ALB: oppositely to 1) and 2), points belonging to Ill class seem slightly drifted towards lower values, compare to the Healthy class
4. CREA, GGT, ALP: in the Ill class some points seem to be very distant from the others and they might be outliers. Especially in the CREA case where a point with a value close to 1000 lead to the compression of

the plots. Such points are that far from the whiskers that they might actually be outliers, but a more quantitative analysis will follow in the next paragraphs.

Summarizing the previous remarks, AST and GGT are the two features that taken individually may help the in the diagnosis process. CREA, GGT, ALP values need instead a deeper analysis in order to detect the presence of outliers.

For the feature “Sex”, pie charts are used to see the proportions belonging to class “Healthy” and “Ill”. It can be seen that the proportions are not exactly the same, with the male one increasing in the latter class.

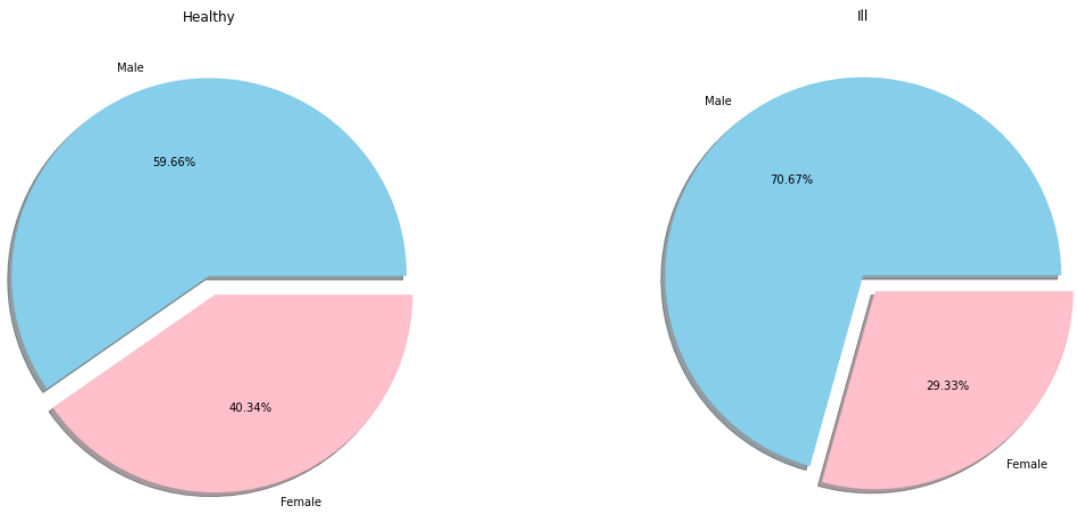


Fig 3. Pie charts with the proportions of the point according to Sex feature.

Bivariate analysis through scatterplots
The univariate analysis is not enough to make some inference, since the belonging to a class or another may depend (and typically it does) on more features at the same time. The qualitative analysis level may be increased by adding one dimension, since it is possible to easily visualize 2 dimensions.
For this reason, scatterplots representing the pairwise relationship between features are plotted on a grid plot, along with the univariate distribution represented on its diagonal.

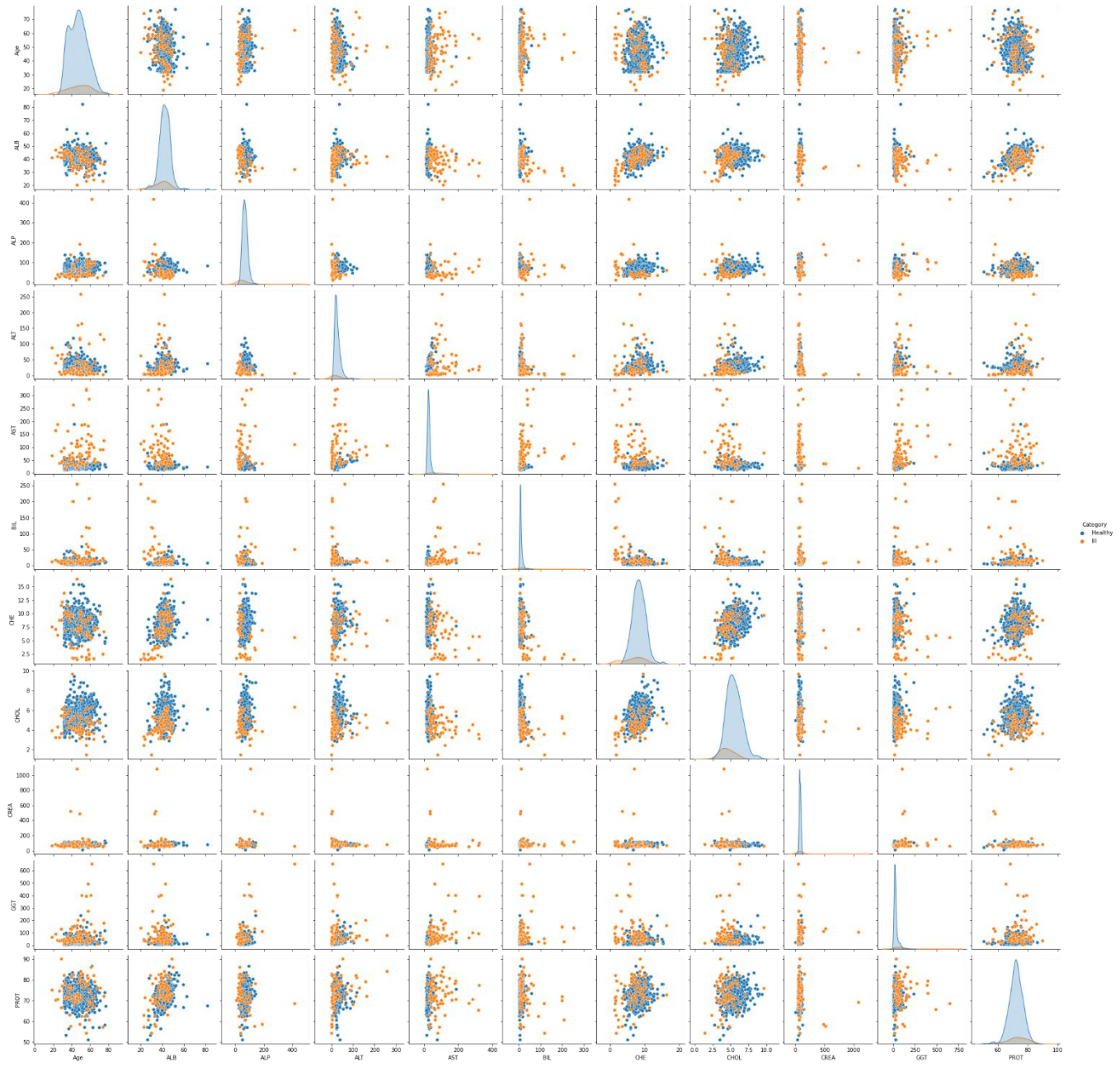


Fig 4. Scatterplots for the bivariate visive analysis and univariate distributions.

As a confirmation of what already observed thanks to the boxplots, the distributions for the classes follow a similar shape (see the plots on the diagonal). In addition, the presence of values very far from the median compress the visualization (not the position!) of the other data points: this is because they enlarge the area on which the data points falls.
From these scatterplots, it is difficult to understand if same distribution follow a cigar-like shape (typical clue of correlation) but the case ALB-PROT, where a similar pattern is visible and leads to the hypothesis of a positive (due to the positive slope of the cigar) correlation between such variables. Also CHE-ALB and CHE-CHOL shows something similar, although weaker.

Correlation matrix
Quantitative analysis may confirm and reveal correlation between features. In this work the analysis will focus on the **Pearson's correlation coefficient**, defined as:

$$\rho_{(X,Y)} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

where X, Y represent two random variables and σ_X, σ_y the respective standard deviations.
Given such formula, it is clear that $|\rho_{(X,Y)}| \leq 1$ and while the absolute values gives a measure of the relationship between the two variables (the closer to 1, the stronger), the sign gives an indication of the direction: direct relationship if positive, inverse if negative. Functionally speaking, if two variables are highly positive correlated ($\rho_{(X,Y)}$ close to 1) it means that an increase in the value of one will correspond to an increase in the value of the other, while if negative correlated the increase in one will correspond to a decrease in the other. The opposite situation no correlation, that is to say when $\rho_{(X,Y)} = 0$. Generally, a correlation coefficient is considered significant when $|\rho_{(X,Y)}| \geq 0.8$.
Detecting the correlation is important for the step of feature selection: if two variables are strongly correlated it means that by analyzing only one of them it is also possible to know the behavior of the other one and for this reason the latter may be ignored. This would reduce the feature space by one dimension, with the related consequence in terms of computational complexity and of geometrical implication.
The *Pandas* corr() function allows computing such correlation coefficients and store them in a correlation matrix (containing on the cell ρ_{ij} the values computed for features *i* and *j*) that can then be plotted as an heatmap:

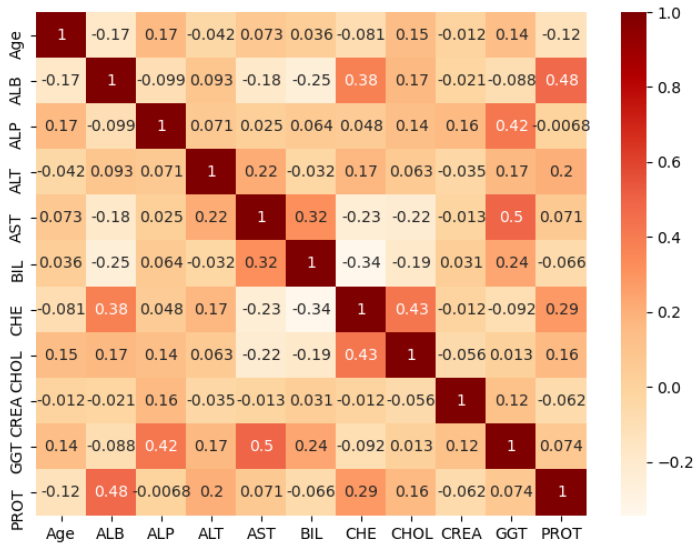


Fig 5. Correlation matrix heatmap.

For the available data, the heatmap shows some positive correlation between GGT-AST, ALB-PROT, CHOL-CHE, GGT-ALP although such values are not high enough to be considered significant (definitely under the threshold of 0.8). In any case the values of the correlation coefficient themselves would not be enough to establish the actual presence of a correlation: hypothesis tests for significance testing would allow detecting the real correlations.
This concludes the exploratory data analysis section, but before going on to the next phase, here some notes about the detected correlations:

- The correlation coefficients confirm what observed on the scatterplots about the pairs ALB-PROT and CHOL-CHE (and for CHE-ALB too);
- The GGT-ALP correlation was not qualitative detected in the previous plots.

4 DATA PRE-PROCESSING

The completion of the exploratory data analysis is followed by the work of pre-processing, consisting in preparing data for putting them in a proper format for the algorithms.

4.1 FACTOR ENCONDING AND FILLING MISSING VALUES

First of all the factor variables are encoded into binary variable. Since here both the factors can only have two different levels, it is enough to transform the columns themselves, and one-hot-encoding is not needed.

- Sex: “m” is encoded as 0, “f” as 1.
- Category: “Healthy” is encoded as 0, “Ill” as 1.

Regarding the missing values they have been filled with mean values, since the shape of univariate distributions resemble normal distributions.

4.2 FEATURE SELECTION AND ENGINEERING

Feature selection means choosing only some of the available features to be included in the computations (the other ones are therefore discarded): this operation reduces the feature space dimension. For example as already explained before when dealing with strongly correlated features it may be sufficient to include in the algorithms only one of them. Another way of performing the selection is through domain knowledge: some features may be already known to be irrelevant for some domains and therefore would only represent a “cost”.
In this case all the original features were selected (excluded the column “Unnamed: 0” that has already been dropped at the beginning).

Feature engineering on the other side is the action of creating new features from the available ones, for example by applying some transformation to one or more variables and substitute them or adding them to the list: this operation may change the feature space in both direction (addition/removal of dimension) or keeping it unchanged in terms of dimension cardinality. For example when dealing with dates, it may be useful to apply some trigonometrical functions in order to keep the information of cyclicality instead of a misleading ordinal feature.
In this case, standardization and principal component analysis were performed on the original data and will be discussed later on.

4.3 STANDARDIZATION

Standardization is the act of scaling the values of the individual features in order to obtain their distribution as close as possible to a Standard Normal distribution.
Scaling is an operation of reducing the window on which the values falls and standardization is only a specific type of scaling. Standardization works particularly well when the original data distribution follows a normal distribution and in our case it looks from the univariate distribution plots that this characteristics holds, therefore no other transformation on the data are necessary before trying to standardize.
The formula for standardize a variable is the following:

$$z = \frac{x - \mu_X}{\sigma_X}$$

Where μ_X is the mean of the variable distribution and σ_X the standard deviation. Note that we do not know the real data distribution, but we have an estimate given by the sample.
Therefore, such quantities are computed on the sample as follows:

$$\mu_X = \frac{1}{N}\sum_{i=0}^{N} x_i$$

$$\sigma_X = \sqrt{\frac{1}{N}(x_i - \mu_x)^2}$$

Standardization is useful in all those situations that require the computation of distances and when different variables need to be compared, either because they become more easily comparable and also because for example when some weights have been computed it does not happen that a larger-valued variable “dominates” another one, reducing its significance.
In this work, standardization has been put inside a pipeline and applied before all the different algorithms discussed, but also used for the task of outlier detection.

4.4 OUTLIERS DETECTION

Outliers are data points that look particularly different from all the other points in terms of distribution. They might have either a “negative” meaning in terms of confounding elements or mistakes that affect the modeling or a “positive” meaning in terms of particular/ interesting points. Depending on the problem under examination, the goal may be detect and exclude them from the analysis or on the opposite detect and focus the analysis on them.
In this case, the interest is in detecting and excluding such points, with a focus on the single features (one-dimensional outliers) since some strange values emerged during the exploratory data analysis phase.

4.4.1 IQR

The first approach is based on the quantiles. The boxplots used before allowed to have an immediate visualization of the main quantities. The below images summarize these elements:

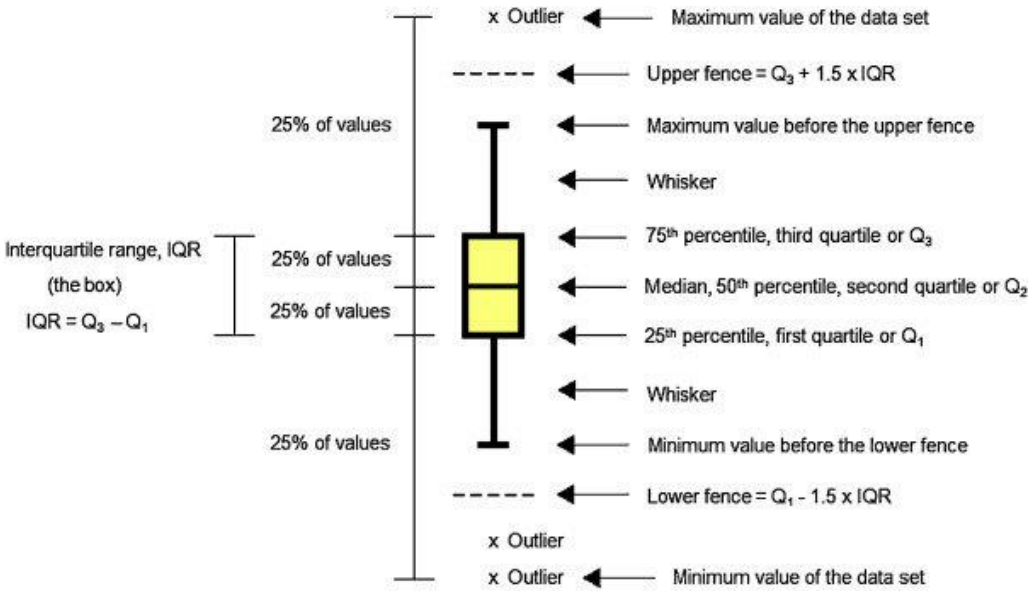


Fig 6. Legend of the boxplot charts and some quantities definition. Image from the web [2].

The Inter-quartile range approach identifies as outliers the point falling beyond two threshold, the upper and the lower fence (whose values are determined by the formulas presented in the image).
The application of such method led to a drastic decrease in the data points available, especially those belonging to class “Ill”: after the removal, the proportion of data left belonging to class “Healthy” and “Ill” were respectively 80,68% and 17,33%.
For this reason, the method was not adopted.

4.4.2 Z-SCORE

The z-score approach is based on the standardization of the data. After this transformation, all data falling beyond a threshold are discarded. In this case, the threshold has been fixed to ± 3.

Also this method led to losing a lots of points, especially belonging to class “Ill”: after the removal, the proportion of data left belonging to class “Healthy” and “Ill” were respectively 95,31% and 44%.
For this reason, the method was not adopted.

4.4.3 MANUAL SELECTION

This method does not have solid mathematical or statistical foundation: it relies on domain knowledge and personal choices. No clinician has been involved in this analysis, then it is impossible to understand which values may represent some outliers (also because the lack of unit measure in the original dataset does not help with searching for answer on the web or other resources).
As a simulation of this type of action, the following criteria have been used in order to exclude some points identified as outliers:

- ALB > 80

- ALP > 300
- CREA > 600
- GGT > 500

This led to the reduction of the dataset size in the following proportions: after the removal, the proportion of data left belonging to class “Healthy” and “Ill” were respectively 99,81% and 97,33%.
These data points have been actually removed.

Technical note: these three different approaches have been included into a specific function created with this work, “*remove_outliers”.*

5 SETTINGS

5.1 DATASET SPLIT

Once the pre-processing has been concluded, the values of the dataframes are collected into a data matrix X and a label vector y.
These two are then divided into two subsets: the training and the test set.
The test set plays the role of the real data and therefore must not be manipulated at all, but only used at the end of the process to evaluate the performance of the models.

When dealing with highly imbalanced classes such as in this case, random splitting is not a good choice because it may end up in a division not representing the real data. For example it may end in a more balanced training set and in a test set without any point belonging to the minority class. The stratified sampling prevent this situation, by maintaining the proportion between the two classes in all the subsets.

The original dataset has been split into a typical 70% Training – 30% Test setting, with shuffling of the data (important, since the records were ordered according to the Category level) and stratification.
The table summarize the obtained split, that satisfies the requirements:

	TRAINING SET	TEST SET
HEALTHY (0)	372	160
ILL (1)	51	22
RATIO ILL/HEALTHY	0,1371	0,1375

5.2 DIMENSIONALITY REDUCTION: PCA

Curse of dimensionality
The dimension of the feature space may lead to severe problems for the algorithm. On one side, in terms of computations: large matrices are difficult to handle for operations such as inversion, multiplication and others.
On the other side, the problem belongs to the feature space itself in terms of mathematical propriety: this is the so called “curse of dimensionality”.
The addition of new dimensions to the feature space stretches it, and the effect on the points is that these are more and more isolated. The higher the number of dimension, the higher the percentage of the space that must be considered in order to get the data points: this is because the high-dimensional spaces are mostly empty. In other words, the density of the data decreases.
Another important consideration is that the points tend to become equally distant from each other, consequently affecting the efficacy of the algorithms. For example algorithms based on distance measures such as K-Nearest Neighbors will fail since it will not possible to build a neighborhood, that is to say finding the “nearest” points (they are equally distant!). On the contrary, an algorithm such as SVM (that will be described later on) will benefit from the fact that the points are well separated and finding the hyperplanes that separates the different classes become easier.
The image below shows these concepts of increased distance and emptiness on the first three dimensions, the ones that we are able to visualize (while it is not possible to visualize the concept of stretched space).

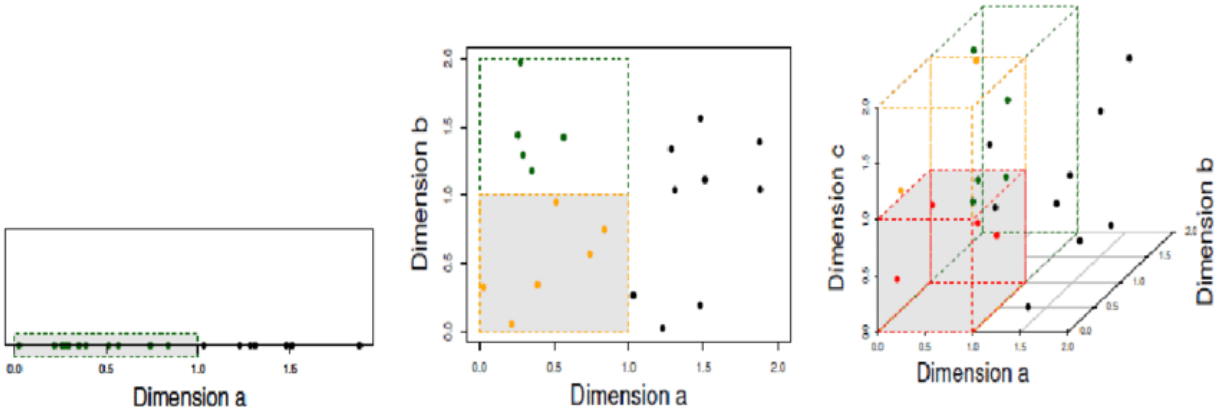


Fig 7. Visual explanation of the curse of dimensionality. Image from the web [3].

Note that the grey area is a n-dimensional hypercube with side equal to 1, and in 1-D is an interval containing all the points, in 2-D is a square containing only partially the points (the previous ones, now projected onto 2 dimensions) and in 3-D is a cube containing even less points. These proceeds towards the infinity, meaning that we should need hypercubes with a longer edge when we want to collects all the points in a higher dimensional space, although the same length was enough in a lower dimensional space.

PCA: Principal Component Analysis

One way to overcome the curse of dimensionality is to rearrange the features such that with less of them (reduced feature space) it is still possible to retrieve the important information contained, without using all of them.
The PCA is a method of performing this rearrangement in such a way that by taking the components one by one in order it is possible to build a space made from the directions that maximize the variance contained in the data.
The PCA is a basic form of autoencoder, consisting in a first function f that projects the data belonging to a d-dimensional space into a lower dimensional space (k-dimensional, k<d), combined then with another function g that send these projected data into the original d-dimensional feature space. The objective of the autoencoder is to find f and g such that their combination coincide to the identity function, that is to say: $g(f(X)) = Id$.

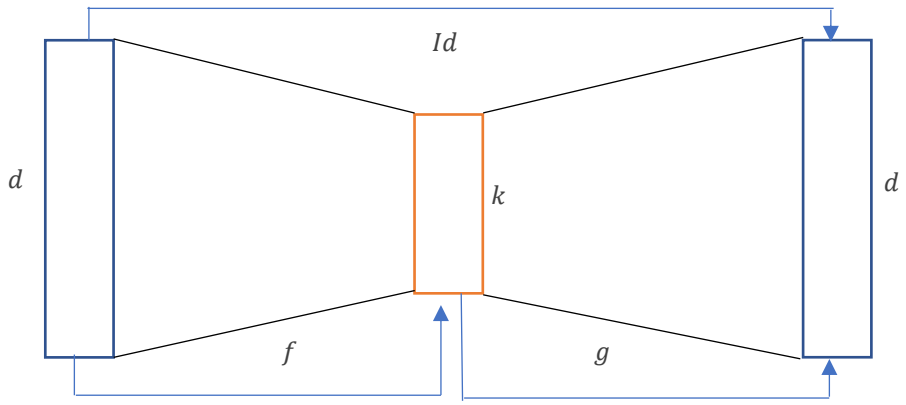


Fig 8. A representation of an autoencoder.

Basically, the objective is to minimize the reconstruction error that can be anecdotally explained with the following example. If there is a cardboard box (3d object) in the middle of a road and a car passes over it, the box is then projected onto the road and becomes a 2d object. When trying to "re-open" it and transform into the original 3d shape, it will most likely not be the exactly same box as before: comparing it to the original one, there will be some difference. Such differences are equivalent to the reconstruction error.
Since the goal is to find two particular linear functions such that their composition leads to the Identity function, this is a **supervised** method (the target is known: the identity function).
The formulation of the problem is the following:

$$argmin_{f,g} \frac{1}{m} \sum_{i=1}^{m} \|x_i - g(f(x_i))\|_2^2$$

Another formulation of the problem uses instead matrices as representation of linear functions and is the following:

$$min_{U \in R^{d,k}, W \in R^{k,d}} \sum_{i=1}^{m} \|x_i - UWx_i\|_2^2$$

The objective here is to find the two matrices U,W such that the sum of the quadratic errors is minimal. The solution to this second formulation are the matrices U and $W = U^T$, where U is constituted from the columns $u_1, \dots, u_k$ corresponding to the *k* leading eigenvectors of the sample covariance matrix.
These vectors represents the principal components, and are also the directions along which the variance is maximal. They generate then new subspace on which the original points are projected.
Taken the principal components in order, each one is orthogonal to all the previous ones. Furthermore, the associated eigenvalues represent the quantity of explained variability.
It is important to add that the operation required as the first step of the process is the standardization: in this way the new data distribution (cloud of points) will be centered in the origin of the axes and the calculation will be easier.

When reasoning with the autoencoder the focus is on the reconstruction error and the problem had therefore a geometrical meaning.
When reasoning on the components retrieved as eigenvector of the sample covariance the focus is instead on finding the best linear combinations of features such that the **new** variables found (that is to say such linear combinations) one after another have the maximal variance and are uncorrelated to the previous ones (orthogonality of the vectors). Two different reasoning that have the same solution.

There is a third problem that has the same solution and it is the following:

$$argmin_V \sum_{i=1}^{m} d(x_i, V)^2$$

This means finding a subspace such that the sum of the distances between the points and the subspace is minimal. In this case the solution is the subspace generated by the principal components: $V_k = L\{PC_1, PC_{2,} \dots, PC_k\}$.
The meaning this time is a multi-dimensional regression.

All that said, the most challenging part is deciding the proper value of *k*. The solution is transforming the data so that all the feature are rearranged into the principal components and then plotting the individual and cumulated explained variance. Then, after deciding the minimal amount of explained variance (tipically, at least 90%) looking for elbow points from the plots: they mean that the following PCs only have a very small contribution to the total variance explanation, compared to the previous ones.

In order to perform PCA the training set needs to be transformed into a matrix X and standardized.

Here below the PCA plot for the dataset HCV:

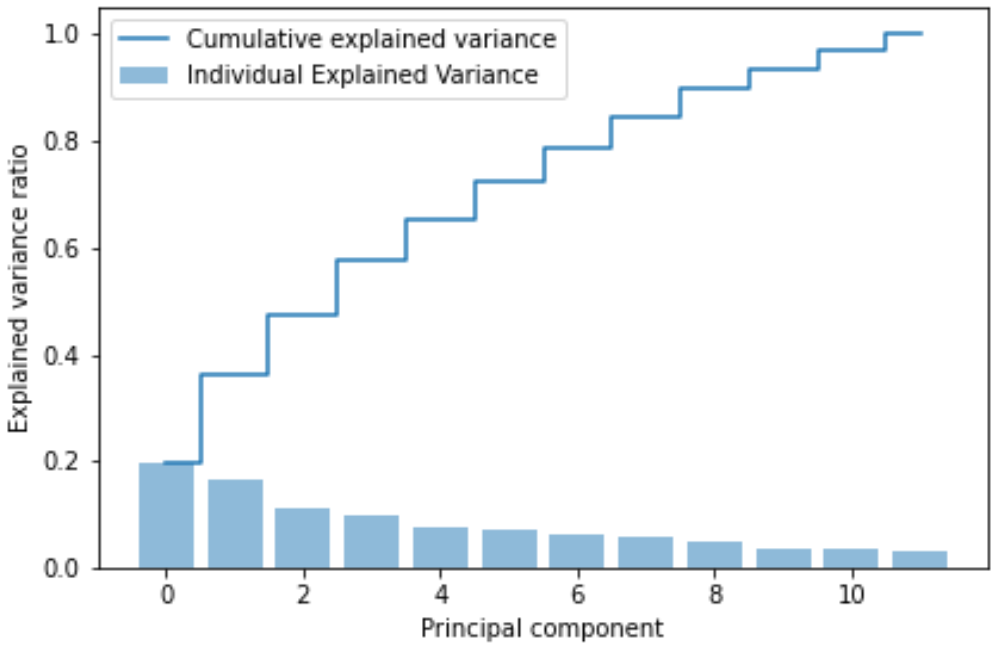


Fig 9. PCA for HCV Dataset.

This dataset does not have a high number of features, hence the typical shape with an elbow point is not visible (and also knowing that, a graph of type “steps” has been chosen).
In order to keep at least 90% of the explained variance 10 principal components have to be kept (values corresponding to bar labeled “9”) over the 12 available. The PCA has been applied in order to study this method, and the selection of the number of components will be assigned to the cross-validation phase.

Downsides of the PCA:
The main downside is that the rearrangement of the original features into linear combinations leads to the loss of interpretability of the models, since the new values are not referring to the “meaning-known” variables. Therefore if applied before decision trees it cancel the main advantage of the latter. On the other side, this property may be used in order to anonymize data.
A limit is instead that data are projected over the directions that maximize the variance, not the one that maximize the interclass separability.

5.3 CLASS BALANCING: OVERSAMPLING WITH SMOTE

The high imbalance between the two target classes may affects negatively the training of the models. The best way to overcome such problem is to create some sort of balance, and the possible ways are two: downsampling the majority class or upsampling the minority one.
The HCV dataset is relatively small (423 data points for the training set) so it is more reasonable to follow the second path, otherwise the result would mean training the models on very few instances, leading almost for sure to bad prediction models. In particular, the SMOTE oversampling technique will be used.

SMOTE: Synthetic Minority Oversampling TEchnique
As the name suggests, the oversampling with SMOTE consists in generating synthetic data belonging to the distribution of the minority class. The method is based on the neighborhoods: for each point of the minority class x_i (intended as vectors, not to be confused with the previous notation where it used to indicate the variable *x* value of a point *i*) the *k* neighbors are considered and one at random among them is selected, x_{zi}. Then a coefficient λ is randomly selected uniformly from an interval [0,1] and such proportion of the distance between the two points is added to the starting point, on the direction towards the other one.

$$x_{new} = x_i + \lambda(x_{zi} - x_i)$$

The component $(x_{zi} - x_i)$ determines the direction along which the new point will be generated, the coefficient λ the distance from the original one.

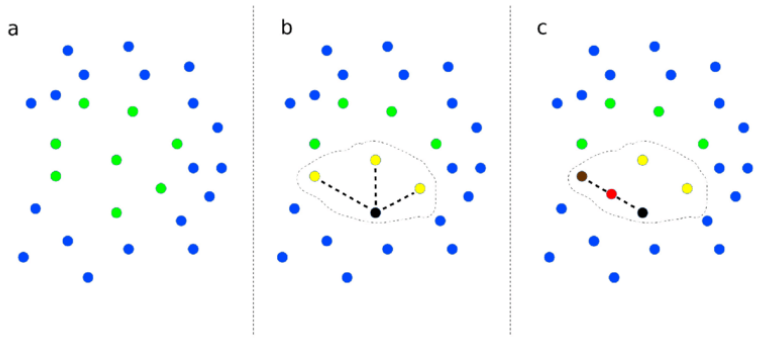


Fig 10: SMOTE algorithm. a) original data, b) neighborhood identification, c) synthetic data generation (in red). Image from the web [4].

Oversampling can be applied only on training set, the test set and the validation set must not be touched.
The advantage compared to a naïve random oversampling where the oversampling is given by the sampling with replacement of already existing point is that here the information gain is higher, since the distribution is forced to assume a more generalized shape.

5.4 EVALUATION METRICS

The last step of the setting for the training is the decision of quantifiable evaluation metrics to be used for the hyperparameters tuning and the model selections. The choice depends first of all on the type of task, in this case Classification.

The most used metrics in classification tasks are based on the confusion matrix, that in a binary classification context looks like this:

		Actual Class	
		Positive (**P**)	Negative (**N**)
Predicted Class	Positive (**P**)	True Positive (**TP**)	False Positive (**FP**)
	Negative (**N**)	False Negative (**FN**)	True Negative (**TN**)

TP, FP, FN and TN are positive integer numbers corresponding to the count of the occurrences of that specific case (for example False Positive counts the number of occurrences predicted as “Positive” but that were actually belonging to “Negative” class).
The typical measures used in classification are:

- **Accuracy**: ratio between the count of all the correct predictions and the count of all the predictions. It answer to the question: “Considering all the predictions made, how many were actually correct?”

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**: ratio between all the correct positive predictions and the count of all that were predicted as positive. It answer to the question: “Considering all the ones that you predicted as positive, how many were actually positive?”

$$P = \frac{TP}{TP + FP}$$

- **Recall**: ration between all the correct positive predictions and the count of all the data actually being positive. It answer to the question: “How good were you in detecting all the positive cases?”
  This measure is also called **sensitivity**, especially in the medical and statistical field.

$$R = \frac{TP}{TP + FN}$$

- **F1 score**: this is more peculiar compared to the previous ones, since it is able to capture the effects of class imbalance. It is a harmonic mean of Precision and Recall and it works as a version of Accuracy for imbalanced sets.

$$F1 = 2\frac{P * R}{P + R}$$

When dealing with imbalanced classes, the accuracy is not a good choice because it is not able to detect the level of performance with respect to the underrepresented class.

The F1-score has been selected as the measure to be maximized in the cross validation.

6 MODEL SELECTION

One of the most important concept in machine learning is that the perfect algorithm that learns all the data distribution does not exist. Some algorithms will perform well on some distribution but maybe poorly on some others, where others will outperform them.
Fixed thresholds of probability and error (<1/2) and given any learner belonging to a learner class and a training set size m, there exists a distribution such that with the previously fixed minimal probability the failure (measured through loss functions) of the learner trained over a training set S of m samples data will exceed the fixed error threshold.

The choice of the better model is then guided by previous experience on the type of the problem that must be faced, but the proper approach is to try different models and evaluate through some measures.

6.1 CROSS-VALIDATION

Cross-Validation is used for testing the performance level of a model. It uses the training set as it was the starting dataset and splits it into sub-portions in order to replicate the “training set – test set” availability, but in this case the test set is called **validation set**.
The model is trained on the training set and then evaluated through the results over the validation test, that simulates real data and must be therefore untouched. This anticipated testing phase allow to the detect whether the model is able to generalize the data distribution or is affected by underfitting (by being too simple) or overfitting (by being too complex and fitting to the training data).
Actually, in order to give the model the opportunity to work on different data, such split in training-validation sets (and the corresponding training-test phases) is repeated several times (each time on a new copy of the original training set) and the results in terms of loss are then averaged.
This averaging approach is done with the scope of reducing the variability of the validation results: indeed, maybe with a specific split the model looks very performing and then after training it on the whole training set and testing on the original test set the results are very different from what expected. This maybe because a “lucky” split occurred, while on a different one the results would have been very bad.
In short, cross-validation consists in repeating this training-validation split and the corresponding trainining-evaluation phases several times and then averaging the results.

There are different way to perform such splitting, and two of them will be briefly described.

Leave-one-out cross validation
In this case the split is repeated as many times as the cardinality of the training set, here denoted as *m*. At each round, *m-1* points are used to form a training set and the one left out is used for the testing. It is obvious that this technique gives a deeper evaluation of the model, but it is also computationally demanding when facing large datasets.

K-fold cross validation
In this case the split is repeated *k* times. The training set is equally partitioned into *k* folds and at each round *k-1* of them are used for the training and the remaining is used for validation. It is clear that the leave-one-out is a special case of k-fold with *k=m*. Typical choices for *k* are 5,10.

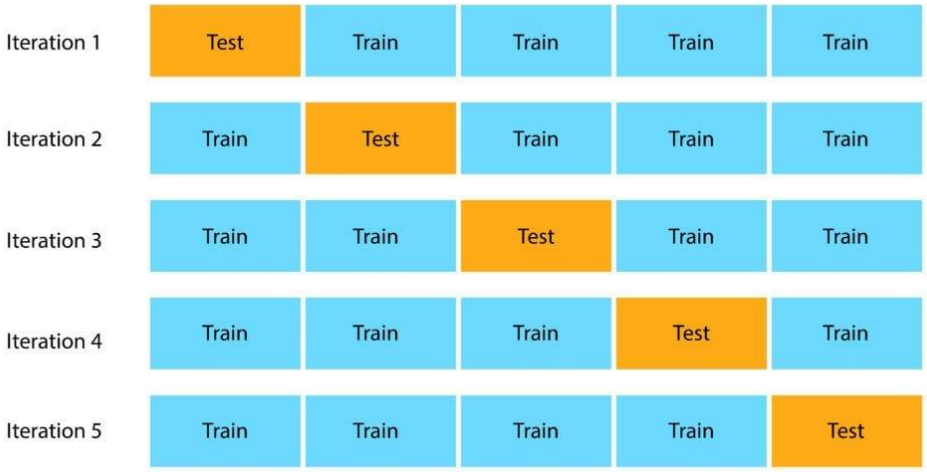


Fig 11. Visual explanation of 5-fold cross validation.

The following is the formula for computing the cross-validation loss (Loss indicating a generic loss function):

$$Loss_{k-folds} = \frac{1}{k}\sum_{i=1}^{k} Loss_i$$

A normal k-fold cross validation randomly picks points for partitioning, without taking into account the potential unbalance of the classes. Consequently there might be partitions not representative of the true distribution: in particular the validation set that as already written should represent the real situation may not be adequate. The solution is performing a **stratified k-fold cross validation**, where the partitions maintains the proportions of the points belonging to different classes.
In addition the training phase may benefit from oversampling in case of highly imbalanced classes, but it is extremely important that this resampling is not applied to the validation set. The reason has already been explained with respect to the test set and the validation set is the same thing but in the context of cross validation.

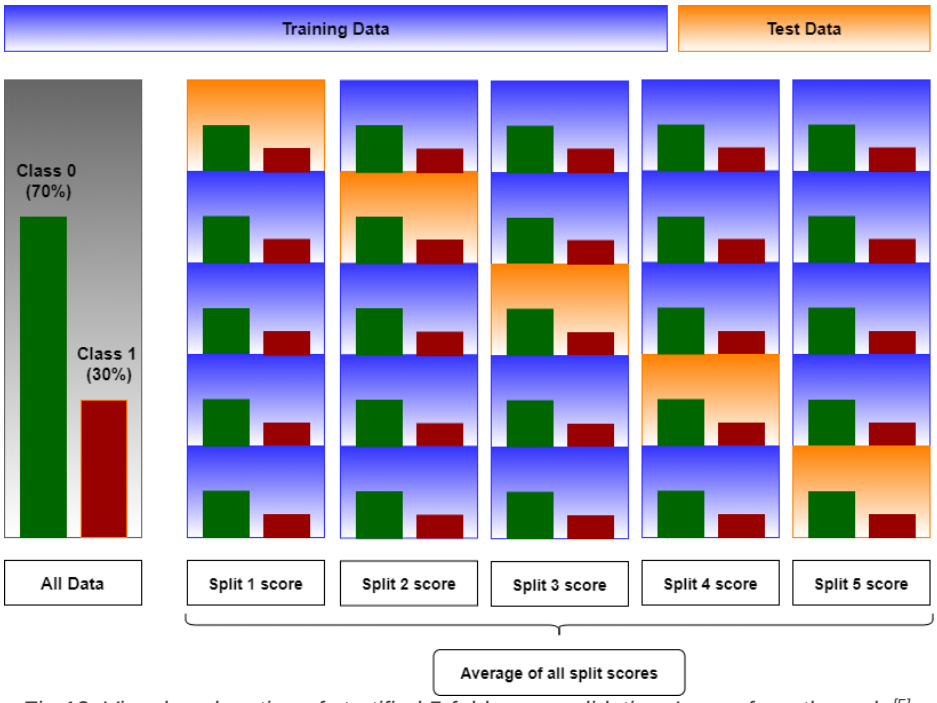


Fig 12. Visual explanation of stratified 5-fold cross validation. Image from the web [5].

In this work a stratified 5-fold cross validation has been used.

6.2 ALGORITHMS

The next paragraphs briefly explain some popular algorithms used for classification tasks under a mathematical perspective. Each type of the following models has been included into pipelines then trained and fine-tuned using grid search with cross-validation.

***Technical note**: the pipelines used for the training also contain some pre-processing operations, such as Standardization and Oversampling. They have been created using the python package imblearn (based on scikit-learn) that is specific for imbalanced datasets. It contains dedicated Pipeline functions that handle operations such SMOTE in a way they are not applied to the validation sets.*

6.2.1 DECISION TREE

The Decision Tree algorithms is a greedy algorithm that consists in splitting the data space in regions such that the majority of the points lying in that region belongs to the same class.
The division of the points is based on a function that is locally constant on **distinct** and **not overlapping** regions R_i having boundaries parallel to the axis of the space. Since it is not feasible to handle all the possible "shapes" of R_i, a first simplification is to consider them as high-dimensional rectangles, or boxes.
The class assigned to each box is the result of the majority vote of the points lying in such box.

This algorithm has a top-down approach: starting from the top, each split of the predictors generates two branches. This is perpetuated until the reach of the leaves, the node where no split can be done because all the points belong to the same class. Of course, the leaves may also be nodes at whose level the set of the points shows some characteristics (cardinality or proportions) according to some pruning criteria. In any case, each leaf corresponds to a high-dimensional box.
Furthermore this is called a greedy algorithm because it is unfeasible to consider all the possible splits in the chain at once and therefore in order to select a feature for the split currently examined, the best at that moment is selected. This does not guarantee that the final solution represent a global optimum and it may be only a local optimum.

Regarding the split criteria, the two more used are the Gini index and the Information gain based on cross-entropy, defined as it follows:

- Gini index: is a measure of the "purity" of the split. If the value is very low, it means that the points are homogeneous (hence, only a class is represented). If the value in 0.5 (max achievable) it means that the points proportions is half-half.

$$G = \sum_{k=1}^{K} p_{mk}(1 - p_{mk})$$

- Information gain: is a measure of how much information we gain thanks to that split and therefore the diminution of the entropy.

$$D = -\sum_{k=1}^{K} p_{mk} * \log(p_{mk})$$

The main advantage of using Decision Trees as model is that they are easily interpretable and the (detected) most important features are used at the higher levels of the tree (towards the root node).
The important hyperparameters to consider when building the model are the pruning criteria, for example the maximal amount of leaves (and therefore high-dimensional boxes of the space) and depth of the tree. These represent a form of regularization.

Experiment
In the code, the Decision Tree model has been added into a pipeline after standardization and oversampling. PCA has not been applied in order not to lose the interpretability associated to the model. The grid search gave the following results for the choice of the hyperparameters (max F1-score as training objective):

{'classifier__criterion': 'gini', 'classifier__max_features': 'sqrt'}

The following image displays instead a portion of the decision tree (trained on the whole training set) in a graphical representation, where the color of the boxes is related to the class most represented at that level (white mean perfect balance between the two and is indeed associated to a gini index 0.5).

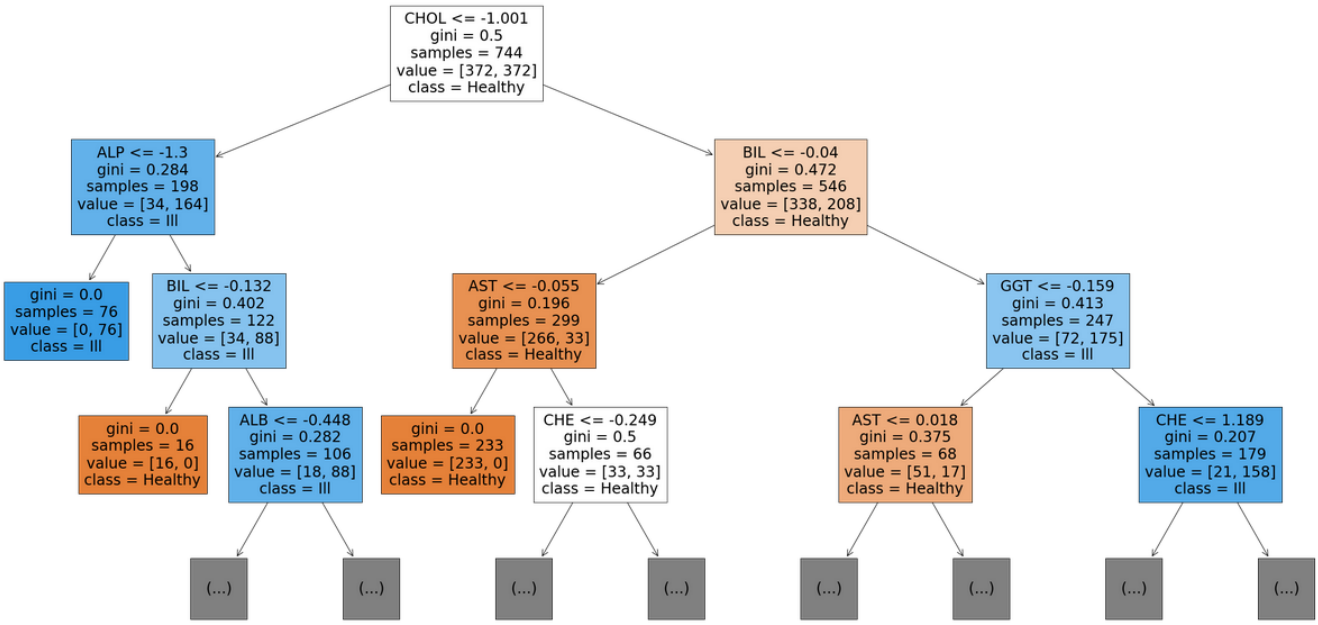


Fig 13. Partial representation of the Decision Tree. As it can be see, it is easy interpretable.

Finally, the confusion matrix after the testing phase on the starting test set.

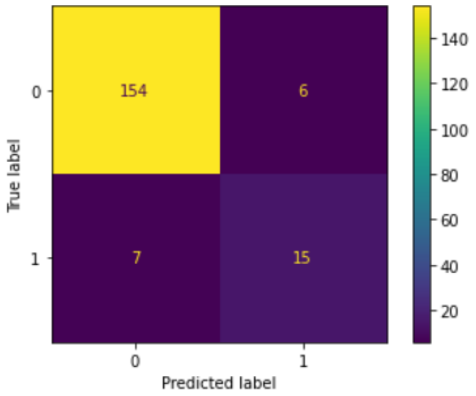


Fig 14. Confusion matrix for Decision Tree

6.2.2 RANDOM FOREST

Random Forest is an ensemble model based on decision tree, a sort of wisdom of the crowds: the predictions given by several decision trees are aggregated and a final prediction is given by the majority vote.
Behind this easy explanation there are two further concepts.

Bagging
The term “Bagging” refers to the combination of “Bootstrap” and “Aggregation”.
Bootstrapping is a resampling method, where random sampling (all the points have equal probability of being picked) from the original one with replacement produces different copies of a dataset. The bootstrapped sets have the same size of the original one and are independent and identically distributed and each tree is trained using one of these.
In the aggregation phase, the results of the training on the different dataset are aggregated to produce a majority vote.

Feature bagging

Even if the training of the different trees is performed on different dataset they would happen to be somehow correlated, because the higher splits will be typically performed on the same features. This correlation would lead to the fail of the law of the large numbers, therefore denying the validity of the majority vote as the "true" prediction.
In order to prevent this to happen, a solution is considering at each node only a subsets of the available features for evaluating the best split. Given p the number of available features, a typical choice for the subset dimension is $\sqrt{p}$.

Random Forest consists then of these two concept: Bagging + Feature bagging, the latter as an attempt to reduce the correlation among the decision trees.
The model has the benefits of being effective and stable, but the main advantage of decision trees that is to say interpretability is lost: this because the results are given on majority votes, therefore on the outputs of the trees.
Still, some functions of the scikit-learn libraries allow to retrieve an indication of the most important features, based on some computation made on each decision tree.

Experiment
In the code, the Decision Tree model has been added into a pipeline after standardization and oversampling. PCA has not been applied in order not to lose the information related to the feature importance that can be retrieved. The grid search gave the following results for the choice of the hyperparameters (max F1-score as training objective):
{'classifier__criterion': 'entropy','classifier__max_features': 'sqrt','classifier__n_estimators': 200}

Finally, the confusion matrix after the testing phase on the starting test set.

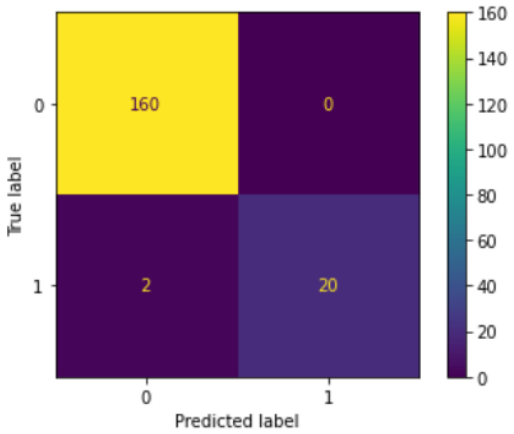


Fig 15. Confusion matrix for Random Forest.

6.2.3 SVM CLASSIFIER

Support Vector Machine is an algorithm based on distances computation. It is based on the assumptions that the data are linearly separable, that is to say that is possible to find a demarcation line that divides the classes. This also means that SVM Classifier is naturally a binary classification algorithm, that can then me adapted to a multi classification problem.
Assumed that the classes are actually linearly separable, SVM consists in finding the hyperplane that best separates the points belonging to them.
In order to do so, not all the points are relevant in the task but only the ones closer to the hypothetical hyperplane: these are the so-called "*support*" points. And since a point may be described by more features, they are called "*vectors*": hence, the definition of *support vectors*.
By means of "best separating hyperplane" it is referred to the one hyperplane such that the distance between the support vectors and it is maximal: this distance is called *margin*, and for this reason this algorithm is also called "margin classifier".
SVM works better in high dimensional spaces, since as already explained the points are more and more distant up to becoming equally distant when the number of dimensions goes to infinity.

Hard margin SVM
The problem of the margin maximization is formulated as follows:

$$argmax_{(w,b):||w||=1}\ min_{i\in[m]}\ |< w, x_i > +b|$$

$$\text{s.t. } y_i(< w, x_i > +b) > 0\ \forall i$$

where $< w, x > +b$ represents the hyperplane. The constraint means the all the points must be correctly classified (with class labels {-1,+1}).
By encompassing the constraint in the objective function it can be rewritten as follows:

$$argmax_{(w,b):||w||=1}\ min_{i\in[m]}\ y_i(< w, x_i > +b)$$

In these two formulations there is the constraint on $\left|\left|w\right|\right| = 1$. Another equivalent problem is instead focused on the minimization of the quantity $\left|\left|w\right|\right|^2$ after removing the constraint of its norm equaling 1. Hence, since the distance between the support vectors and the hyperplane is computed as

$$d = \frac{y_i(< w, x_i > +b)}{\left|\left|w\right|\right|}$$

minimizing $\left|\left|w\right|\right|^2$ would result in maximizing the distance, and the corresponding formulation of the problem becomes:

$$(w_0, b_0) = argmin_{w,b}\ ||w||^2$$

$$\text{s.t. } y_i(< w, x_i > +b) \geq 1 \qquad \forall i$$

, where the constraint is necessary in order to exclude the trivial solution $W = 0$ (capital W representing the vector with all the components w).
The previous formulations are also known as “hard margin” SVM because there is no tolerance for the mistakes: the constraints state that all the point should be correctly classified. In the majority of cases it is not possible to perfectly linearly separate the data and misclassifications may happen due to noise and outliers. For this reason, a variation of the problem exists and it is called Soft margin SVM.

Soft margin SVM
In the Soft margin SVM the constraint about the null misclassification is relaxed, allowing the model to make some mistakes. Of course the mistakes add penalties to the objective functions, where a “mistake” is a misclassified point. Soft margin SVM is therefore a regularization applied to the SVM algorithm.
The following is a formulation of the problem:

$$argmin_{w,b,\xi}\ (\lambda||w||^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i)$$
$$\text{s.t. } y_i(< w, x_i > +b) \geq 1 - \xi_i \qquad \forall i$$

where ξ_i are called slack variables and represent a sort of “toll” that the point has to pay for being over the border.

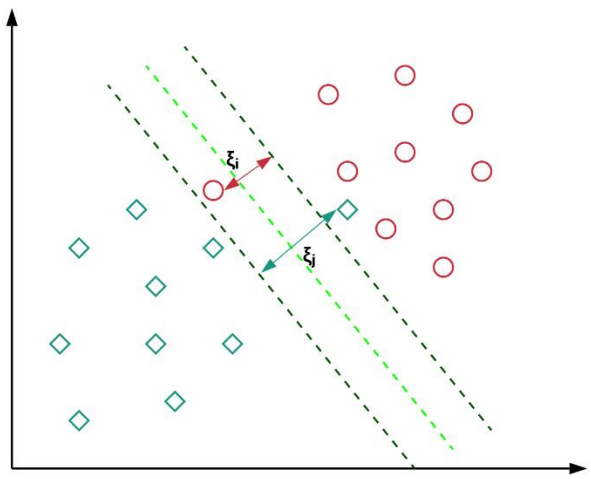


Fig 16. The Soft margin SVM, with highlight on the slack variables. Image from the web [6].

By putting an upper bound to the sum of these slack variables trough a positive constant value C, the width of the margins can be controlled. In particular, given

$$\sum_{i=1}^{m} \xi_i \leq C$$

If such C is small it means that the tolerance is very low (the total penalty cost must not exceed it) and therefore the margin is small. On the contrary, if C is large means that the margin is larger.

Kernel trick
Although data may not be linearly separable in the original feature space, it may be possible to find a higher dimensional space where this is possible and where they are more distant.
Hence a solution would be sending the original data into a higher dimensional space (through a transformation function) where a separating hyperplane exists, and then taking the projection of such hyperplane on the original feature space.
The problem is that applying the transformation to the data may be computationally expensive. The solution is the so-called “kernel trick”, that uses kernel functions where the data are represented through pairwise similarity comparisons between the original data. The output of such functions is the dot product of the transformed vectors in the higher dimensional space, used then in the objective function.
There exist different kernel functions, among which: polynomial, Gaussian, Gaussian radial basis fuction (RBF).

The main benefit of the SVM algorithms is that they have high accuracy, but on the opposite the lack of interpretability is a downside for several application (for example when dealing with clinical data).

Experiment
In the code, the SVM has been added into a pipeline after standardization, oversampling and PCA (although as stated, SVM works well in high dimensional spaces). The grid search gave the following results for the choice of the hyperparameters (max F1-score as training objective):

{'PCA__n_components': 11, 'classifier__C': 1, 'classifier__kernel': 'rbf'}

Finally, the confusion matrix after the testing phase on the starting test set.

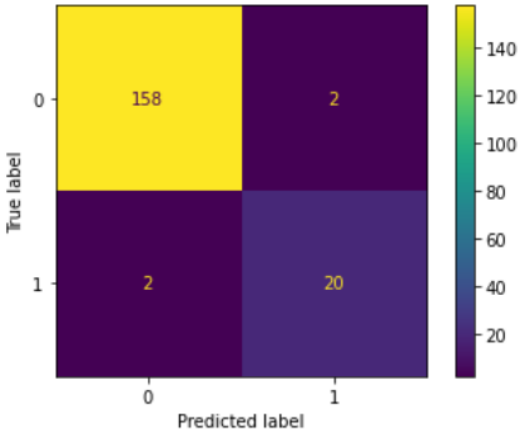
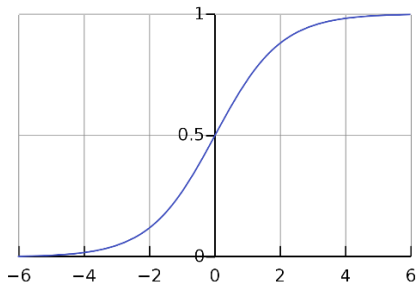


Fig 17. Confusion matrix for SVM

6.2.4 LOGISTIC REGRESSION

The last model analyzed is the logistic regression. It is a General Linear Model that overcomes the problem of the improper mapping returned by a linear regression in a classification context. The latter indeed is not working well with values belonging to a discrete set. For example if 3 classes existed, say “Apple”,”Pear,”Orange” encoded as 1,2,3, a linear approximation would not work well since in terms of number the difference between Orange and Apple is 3-1=2 and between Pear and Apple only 1.
The Logistic Regression return instead a learner model *h* that takes input in $x \in R^d$ and returns a value in the interval [0,1]. These can be seen as probability and $h(x)$ is the probability that $x = 1$ (binary classification case).
Such values is the result of the sigmoid function here represented:



As it can be seen it is an increasing monotone functions that asymptotically goes to 1 for positive values and to 0 for negative ones.
The analytical form of the function is the following:

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

And $p(x) = P(Y = 1|X = x)$. This is not the end of the story, since given a probability it is necessary to compare it to some threshold values in order to decide the class (with the use of different tools).
The name of the model derives from the quantities involved. Starting from the above formula it is indeed possible to compute the so-called "*logits*", as:

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x$$

The main advantage of Logistic regression is that the models are interpretable and provide a probabilistic measure of the prediction (hence, a degree of "certainty").

Experiment
In the code, the Logistic Regression Classifier has been added into a pipeline after standardization and oversampling. The grid search gave the following results for the choice of the hyperparameters (max F1-score as training objective):

{'classifier__C': 1, 'classifier__class_weight': None}

Finally, the confusion matrix after the testing phase on the starting test set.

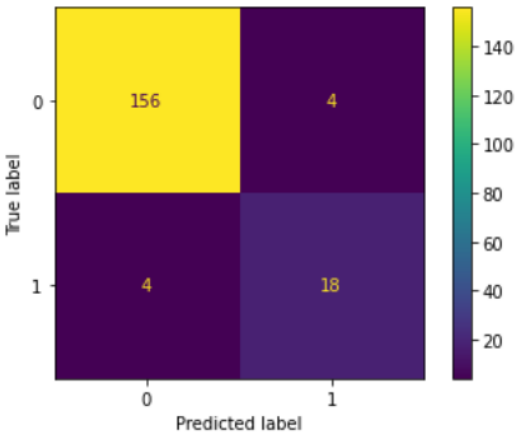


Fig 18. Confusion matrix for Decision Tree.

7 CONCLUSIONS

In the final step the four models (tuned with the settings suggested by the validation phase) have been compared in terms of Accuracy, Precision, Recall and F1-Score.
It can be seen that the models achieving the best performances are Random Forest (that outperforms all the other models) and SVM. Note that for the logistic regression and the SVM classifier the values of precision and recall (and consequently F1) coincide.

	accuracy	precision	recall	f1
Decision Tree	0.928571	0.681818	0.714286	0.697674
Random Forest	**0.989011**	**0.909091**	**1.0**	**0.952381**
Logistic Regression	0.956044	0.818182	0.818182	0.818182
SVM	0.978022	**0.909091**	0.909091	0.909091

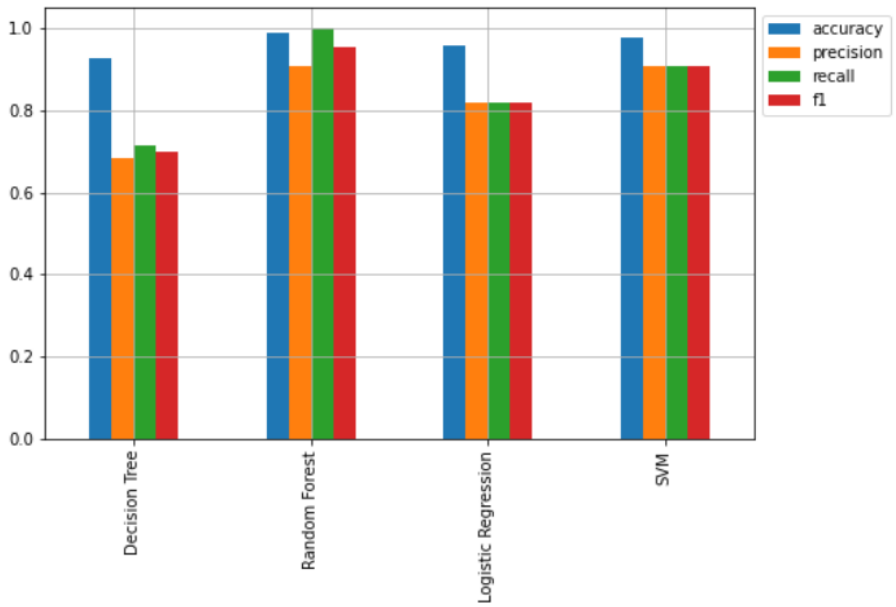


Fig 19. Test results on HCV Dataset for different models.

Furthermore, when plotting the features importance retrieved from the Decision tree and the Random forest (made of 200 trees) it can be seen the two do not agree.

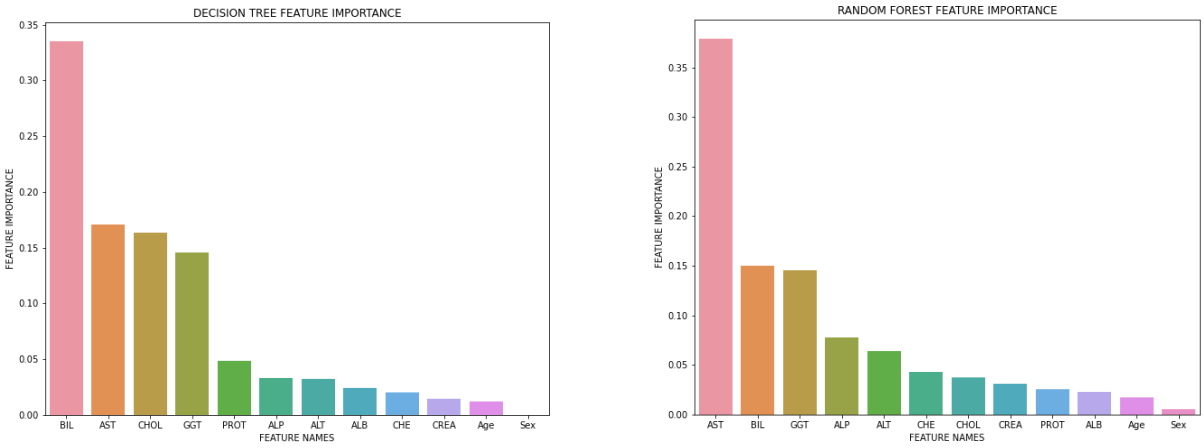


Fig 20. Comparison of feature importance maps for Decision Tree (left) and Random Forest (right).

8 EXTRA: PREDICTION ON THE DISCARDED "UNCERTAIN" DATA

Just to have a taste of a real case, the 7 records belonging to the class "0s= Suspect blood donor" have been fed to the 4 models that have been trained in the previous phases.
As it can be seen, the predictions do not agree, confirming the confounding nature of such points (those ground truth was uncertain too). The "final" prediction has done with a majority vote (although an even number is not the best

choice for this) and when no majority was reached the assigned class followed what predicted from the best model, the Random Forest (such points are marked with *).
Another option could have been weighting the vote according to the test score (after choosing the metrics).

Index of the record	DECISION TREE	RANDOM FOREST	SVM Classifier	Logisti Regression	MAJORITY
533	Ill	Healthy	Ill	Ill	Ill
534	Healthy	Healthy	Healthy	Healthy	Healthy
535	Healthy	Healthy	Healthy	Healthy	Healthy
536	Healthy	Healthy	Ill	Ill	Healthy*
537	Ill	Healthy	Ill	Ill	Ill
538	Healthy	Healthy	Ill	Ill	Healthy*
539	Healthy	Healthy	Ill	Ill	Healthy*

REFERENCES

General references:

- Notes from the course “Mathematics in Machine Learning” by Prof Vaccarino and Gasparini at Politecnico di Torino (year 2021).
- Book “Understanding Machine Learning: From Theory to Algorithms”, Shai Shalev-Shwartz and Shai Ben-David.
- Book “Data Science and Machine Learning: Mathematical and Statistical Methods”, D.P. Kroese, Z.I. Botev, T. Taimre, R. Vaisman [pdf available at https://people.smp.uq.edu.au/DirkKroese/DSML/]

[1] https://www.epicentro.iss.it/epatite/epatite-c

Images from the web:
[2] https://www.google.com/search?q=The-main-components-of-a-boxplot-median-quartiles-whiskers-fences-and-outliers&rlz=1C1ONGR_itIT952IT952&oq=The-main-components-of-a-boxplot-median-quartiles-whiskers-fences-and-outliers&aqs=chrome..69i57.812j0j4&sourceid=chrome&ie=UTF-8
[3] https://medium.com/@soumiksanku08/curse-of-dimensionality-293d0d16fe2a
[4] https://www.researchgate.net/figure/Graphical-representation-of-the-SMOTE-algorithm-a-SMOTE-starts-from-a-set-of-positive_fig2_317489171
[5] https://medium.com/analytics-vidhya/cross-validation-techniques-bacb582097bc
[6] https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe