Politecnico di Torino

Analysis of supervised learning techniques to predict the presence of Cervical Cancer

Ruggiero Francavilla S278312

Mathematics in Machine Learning

Prof. Francesco Vaccarino

Prof. Mauro Gasparini

Contents

Introduction

Cervical cancer is a type of cancer that occurs in the cells of the cervix – the lower part of the uterus that connects to the vagina. It remains a significant cause of mortality in low-income countries.

When identified early, cancer is more likely to respond to effective treatment and can result in a greater probability of surviving, less morbidity, and less expensive treatment. Significant improvements can be made in the lives of cancer patients by detecting it earlier. Therefore, the most important problems during diagnosis are determination of the finest screening plan and estimation of individual risk of each patient.

In this work, a dataset composed of data collected from a survey has been classified, with some supervised learning techniques.

Dataset description

The dataset was collected at 'Hospital Universitario de Caracas' in Venezuela. It comprises demographic information, habits, and historic medical records of 858 patients.

The attributes information is described in the following table.

Feature	**Type**	**Feature**	**Type**
Age	int	STDs: pelvic inflammatory disease	bool
# of partners	int	STDs: genital herpes	bool
Age of 1st intercourse	int	STDs: molluscumcontagiosum	bool
# of pregnancies	int	STDs: AIDS	bool
Smokes	bool	STDs: HIV	bool
Smokes (years)	int	STDs: Hepatitis B	bool
Smokes (packs/year)	int	STDs: HPV	bool
Hormonal Contraceptives	bool	STDs: Number of diagnosis	int
Hormonal Contraceptives (years)	int	STDs: Time since first diagnosis	int
IUD	bool	STDs: Time since last diagnosis	int
IUD (years)	int	Dx: Cancer	bool
STDs	bool	Dx: CIN	bool
STDs (number)	int	Dx: HPV	bool
STDs: condylomatosis	bool	Dx	bool
STDs: cervical condylomatosis	bool	Hinselmann: target variable	bool
STDs: vaginal condylomatosis	bool	Schiller: target variable	bool
STDs: vulvo-perineal condylomatosis	bool	Cytology: target variable	bool
STDs: syphilis	bool	**Biopsy: class or target variable**	**bool**

The dataset is composed of 35 attributes used to predict the final class. The Biopsy result is a Boolean variable, then the analyzed problem is a binary classification.

Exploratory Data Analysis

Missing values

Several patients of the hospital decided not to answer some of the questions because of privacy concerns. This generates missing values.

Many factors present missing values. Not all the factors are managed in the same way.

The first features analyzed are the ones with an elevated number of NaN's, that could create more problems.

The percentage of missing values for

STDs: Time since first diagnosis	91.7%
STDs: Time since last diagnosis	91.7%

The elevated number of missing values of these factors has to be verified.

Then, before analyzing these two, the feature 'STDs: Number of diagnosis' is analyzed.

STDs: Number of diagnosis	Count
0	787
1	68
2	2
3	1

The 787 patients, who have never been diagnosed STDs correspond to the 91.7%, then the missing values of the other two factors is a consequence.

It is decided to replace these values with 0, and not to remove the features, because in the other 71 records there may be some useful information.

Then, factors related to smoking are analyzed. The first to be analyzed is Smokes factor, a Boolean factor that indicates if the patient smoke or not.

Smokes, and therefore 'Smokes (years)' and 'Smokes (packs/year)' have a small percentage of missing value, corresponding to 1.52 %, then it is chosen to replace them with a zero value.

Now, some Boolean factors with about 12-13 % of missing values in each are analyzed:

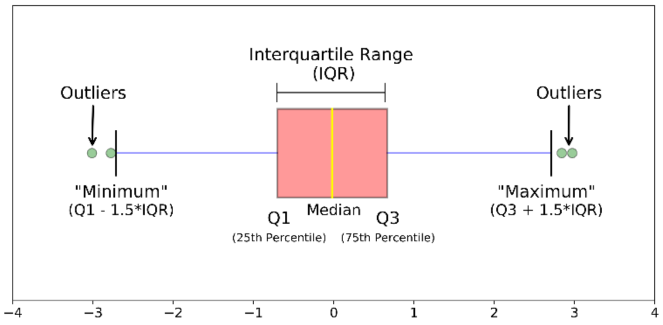
Hormonal Contraceptives	12.59 %
IUD	13.64 %
STDs	12.24 %

It is decided to replace the missing values in each factor with the value between 0 and 1 that have the highest frequency.
IUD and STDs Nan's are filled with 0, while Hormonal Contraceptives Nan's are filled with 1.

Then, some numerical factors having missing values between 1% and 12% are analyzed. To decide how to replace the missing values of these factors, count plot and box plot are displayed, and some statistics, as min, mean, median, mode and max, for each one are computed.

Count Plot is a plot which shows the counts of observations in a categorical factor. It can be thought as a histogram across a categorical variable.

Box Plot is a useful graph to display the distribution of a group of numerical data, through its quartiles.



It is based on five number summary:

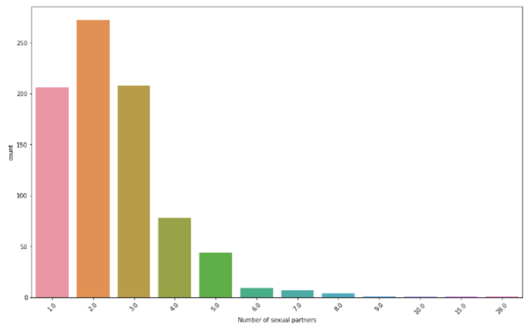
- Median (Q2/50th Percentile): the middle value of the dataset;
- First Quartile (Q1/25th Percentile): the middle number between the smallest number and the median of the dataset;
- Third Quartile (Q3/75th Percentile): the middle value between the median and the highest value of the dataset;
- Minimum: Q1 -1.5*IQR
- Maximum: Q3 + 1.5*IQR

These objects can also have lines extending from the boxes, called *whispers* that indicate the variability outside the upper and lower quartiles.

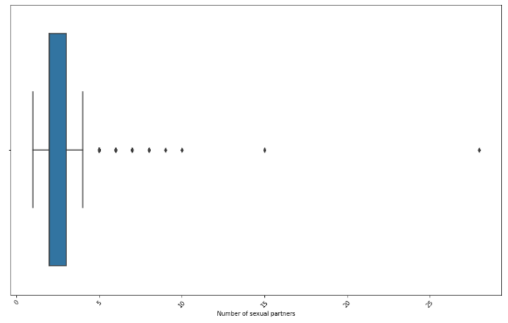
Individual points after the whispers are called *outliers.*

As an example, two of these numerical factors are shown, and analyzed.

Number of sexual partners:

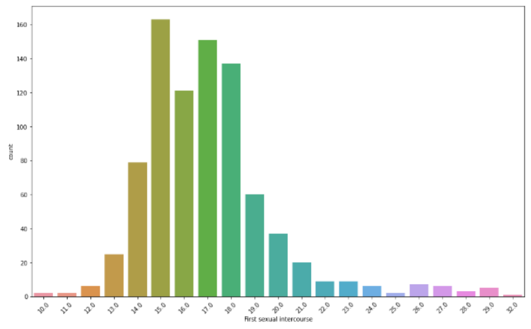


Min	1.0
Mean	2.5
Median	2.0
Mode	2.0
Max	28.0
% of missing values	3.03%

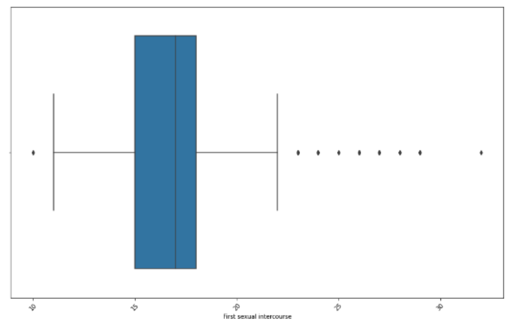


The mean of this factor is pulled higher by some outliers. The missing values will be replaced with the value of the mode and the median.

First sexual intercourse:



Min	10.0
Mean	16.995
Median	17.0
Mode	15.0
Max	32.0
% of missing values	0.82%



Here, less than 1% of values is missing. They will be simply replaced with the median value.

The missing values of the other numerical factors are replaced with the same technique of the previous examples.

There are also other 12 individual Boolean factors, each for a different STD, where each has about 12% missing values. Because these values are related to ‘STDs (number)’, the missing values in each of these factors will be replaced with 0.

Further features visualization

A fundamental task is also to characterize the location and the variability of a data set. A further characterization of the data includes Skewness and Kurtosis measures.

Skewness is a measure of lack of symmetry. A distribution is symmetric if it looks the same to the left and the right of the center point.

For univariate data $X_1, X_2, \dots, X_N$, the formula for Skewness is:

$$g_1 = \frac{\sum_{i=1}^{N}(X_i - \bar{X})^3/N}{s^3}$$

where $\bar{X}$ is the mean, s is the standard deviation, N is the number of data points.

More a distribution tends to be symmetric, more the Skewness measure tends to zero. For a normal distribution, it is equal to zero. Negative values for the Skewness indicate data that are skewed left and positive values for the Skewness indicate data that are skewed right.

Kurtosis is a measure of whether the data are heavy-tailed of light-tailed relative to a normal distribution. Data sets with high kurtosis tend to have light tails, or outliers, instead data sets with low kurtosis tend to have light tails, or lack of outliers.

For univariate data $X_1, X_2, \dots, X_N$, the formula for Kurtosis is:

$$kurtosis = \frac{\sum_{i=1}^{N}(X_i - \bar{X})^4/N}{s^4}$$

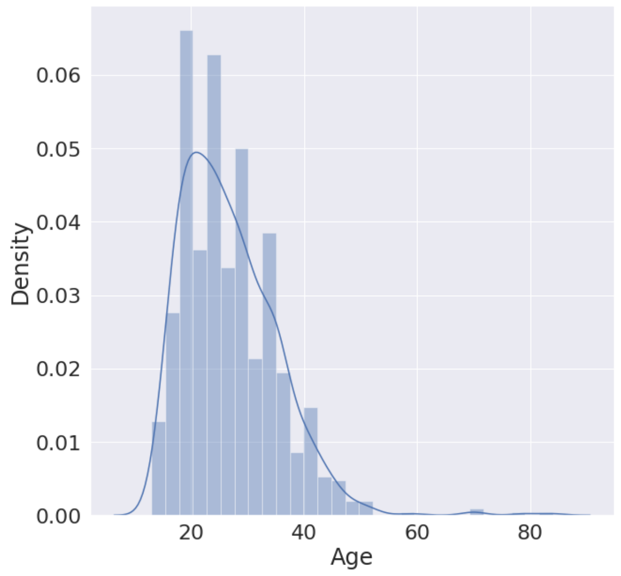
The Kurtosis for a standard normal distribution is 3.

If the Kurtosis value is very low, the tail of distribution will be less long than the tail of a normal distribution.

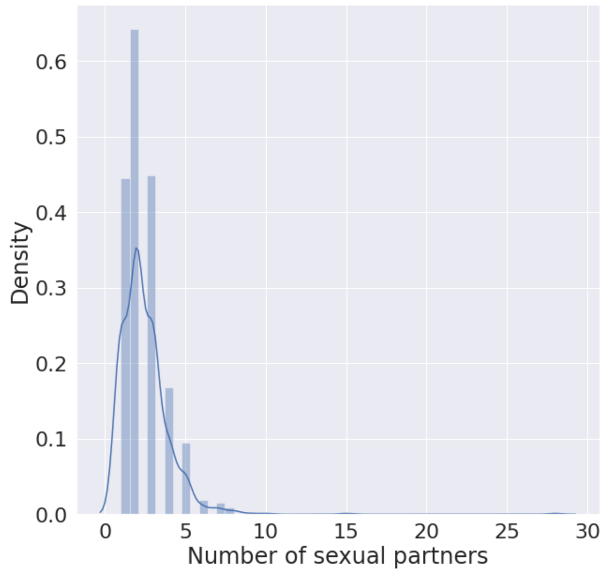
A large value of Kurtosis is often considered as riskier because data may tend to give an outlier value as an outcome with greater distance from the mean.

An effective graphical technique to show these two measures is the histogram. The used function is Dist Plot, that plots a univariate distribution of observations.

Here, two relevant examples of distributions of the numerical features are shown.



$Skewness = \ 1.39$
$Kurtosis = 4.78$



$Skewness = \ 5.54$
$Kurtosis = 71.21$

Both ‘Age’ and ‘Number of sexual partners’ factors, are skewed to the right. The second factor also presents a high Kurtosis, and therefore it is supposed to have many outliers.

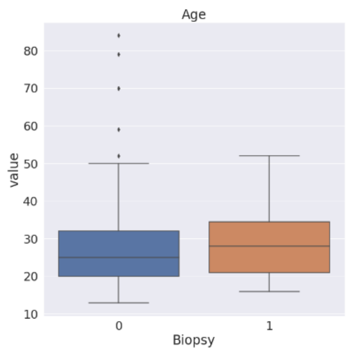
The other numerical feature presents these values for Skewness and Kurtosis:

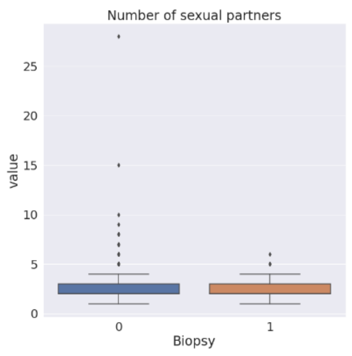
Feature	Skewness	Kurtosis
First sexual intercourse	1.57	4.35
Num of pregnancies	1.50	3.69
Smokes (years)	4.50	24.19
Smokes (packs/year)	9.38	116.6
Hormonal Contraceptives (years)	2.87	10.74
IUD (years)	5.42	35.28
STDs (number)	3.68	13.77
STDs: Number of diagnosis	3.79	17.46
STDs: Time since first diagnosis	6.20	42.03
STDs: Time since last diagnosis	6.38	44.96

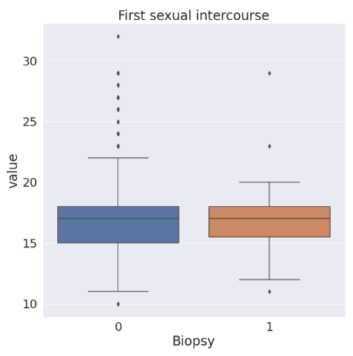
It is very difficult to interpret and analyze the data which is skewed. Likewise, a collection of data points that are normally distributed or nearer to symmetrical are easier for computation and more probable for producing better inferences.

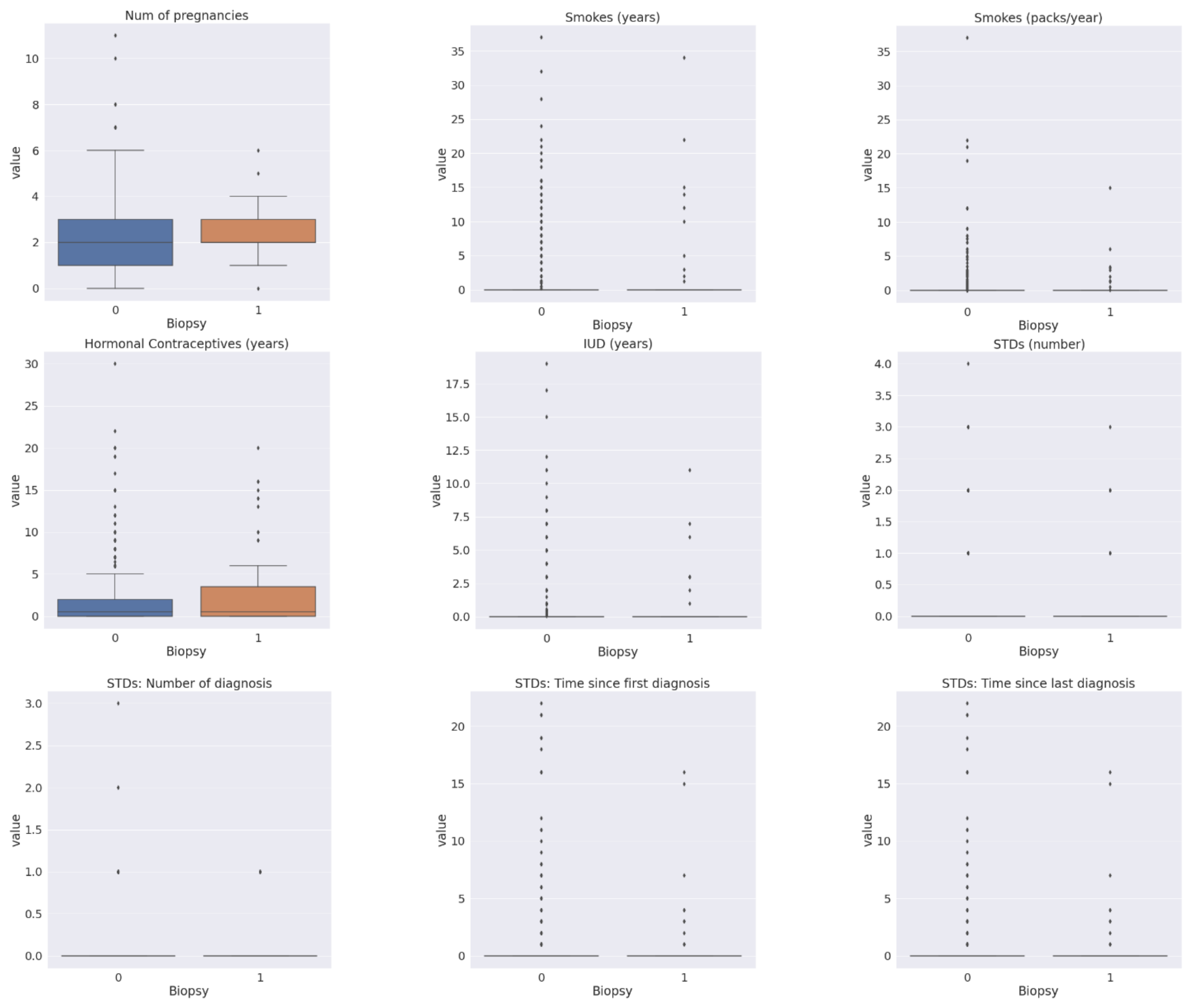
As previously seen, Box Plot helps to choose the best value to replace the missing values and could be also used to detect the outliers of a numerical factor.

Another interesting use of boxplot, consists of showing how the numerical factors are distributed, related to the target class.







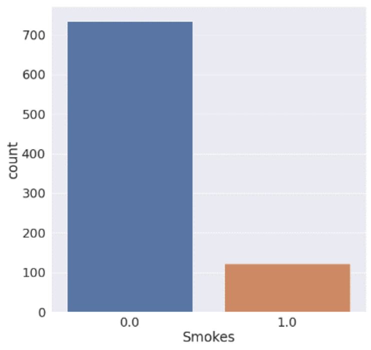


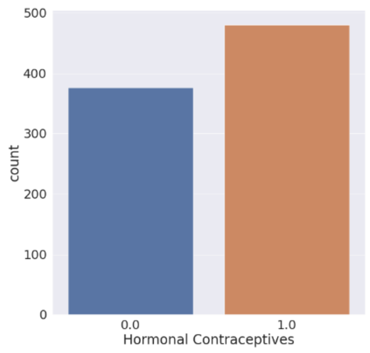
From these graphs another problem arises. Most of the features show an overlap between the distribution of the two classes. This is not good in terms of clear distinction and will result in a model which will find more difficult to distinguish the two labels.

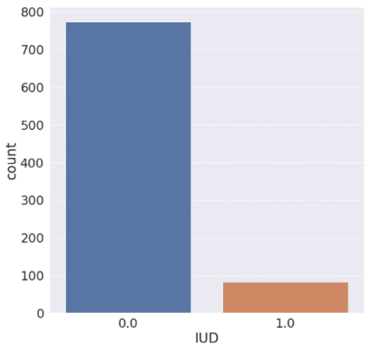
Given these considerations, it can be deducted that we are facing a complex classification problem, mainly in finding the samples belonging to the minor class.

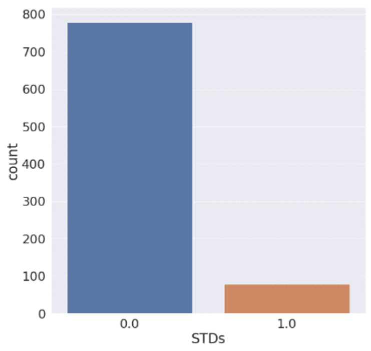
Binary features

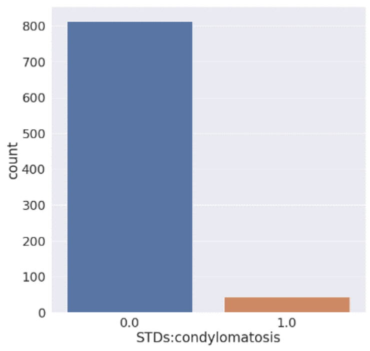
For the visualization of the binary features, it is used the count plot, that show the counts of observations for each value of the target class.

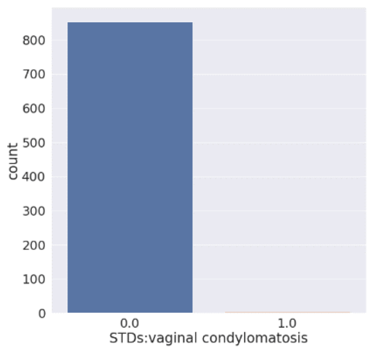


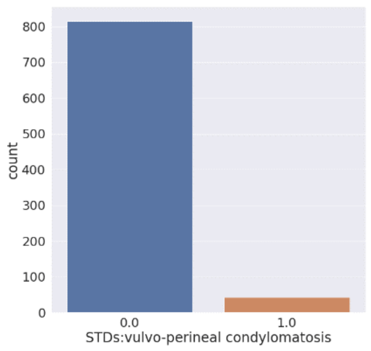


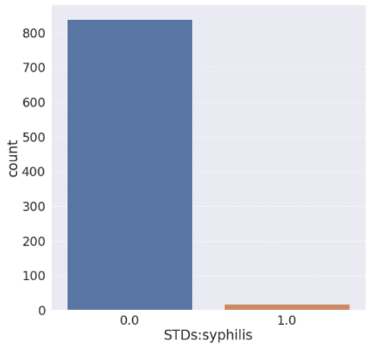


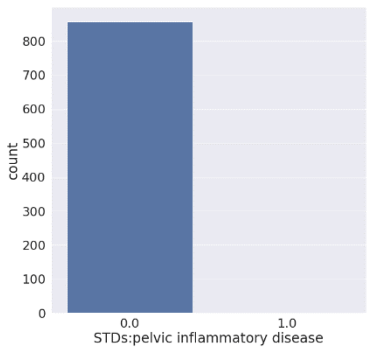


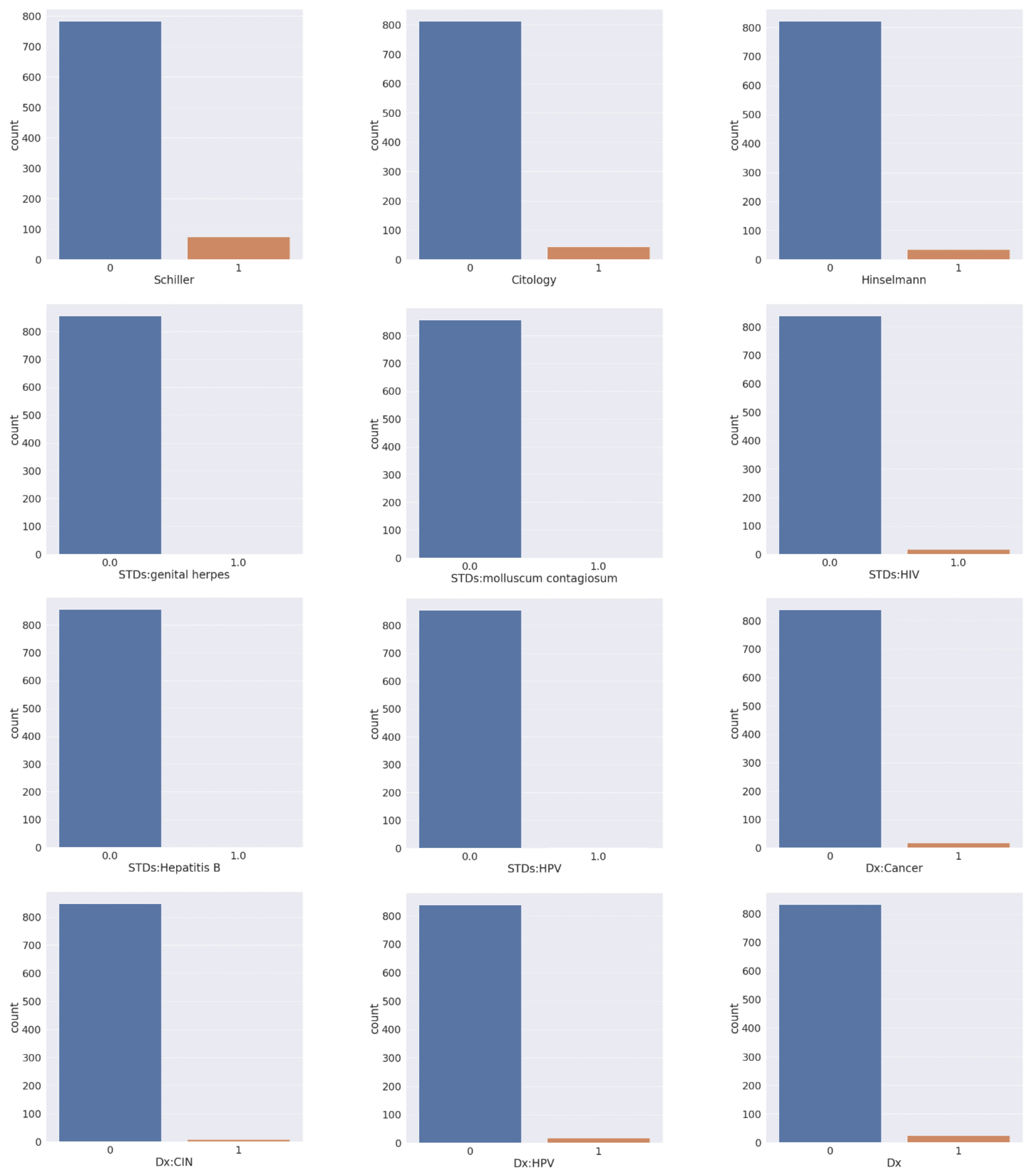












These factors present a clear imbalance of individuals diagnosed with cervical cancer and healthy individuals, except for ‘Hormonal Contraceptives’ factor.

This problem anticipates a problem of imbalanced data that will be seen after.

It is also possible to see from the initial analysis of the data that the two factors ‘STDs: AIDS’ and ‘STDs: cervical condylomatosis’ have not information, because they only contain information about healthy individuals, then it is decided to eliminate them.

Heatmap with Pearson's correlation

The data set has many attributes and in order to analyze them in the best way, it is important to understand the relationships between the attributes.

Correlation matrix, which is made of correlation coefficients, helps us to simply see those relations.

Correlation coefficients are used in statistics to measure how strong a relationship is between two variables.

It can be computed by dividing the covariance of the variables by the product of the standard deviations of the same value.

The most common correlation coefficient is Pearson's.

Given paired data $\{(x_1, y_1), \dots, (x_n, y_n)\}$ consisting of n pairs, the correlation between variable x and y is:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

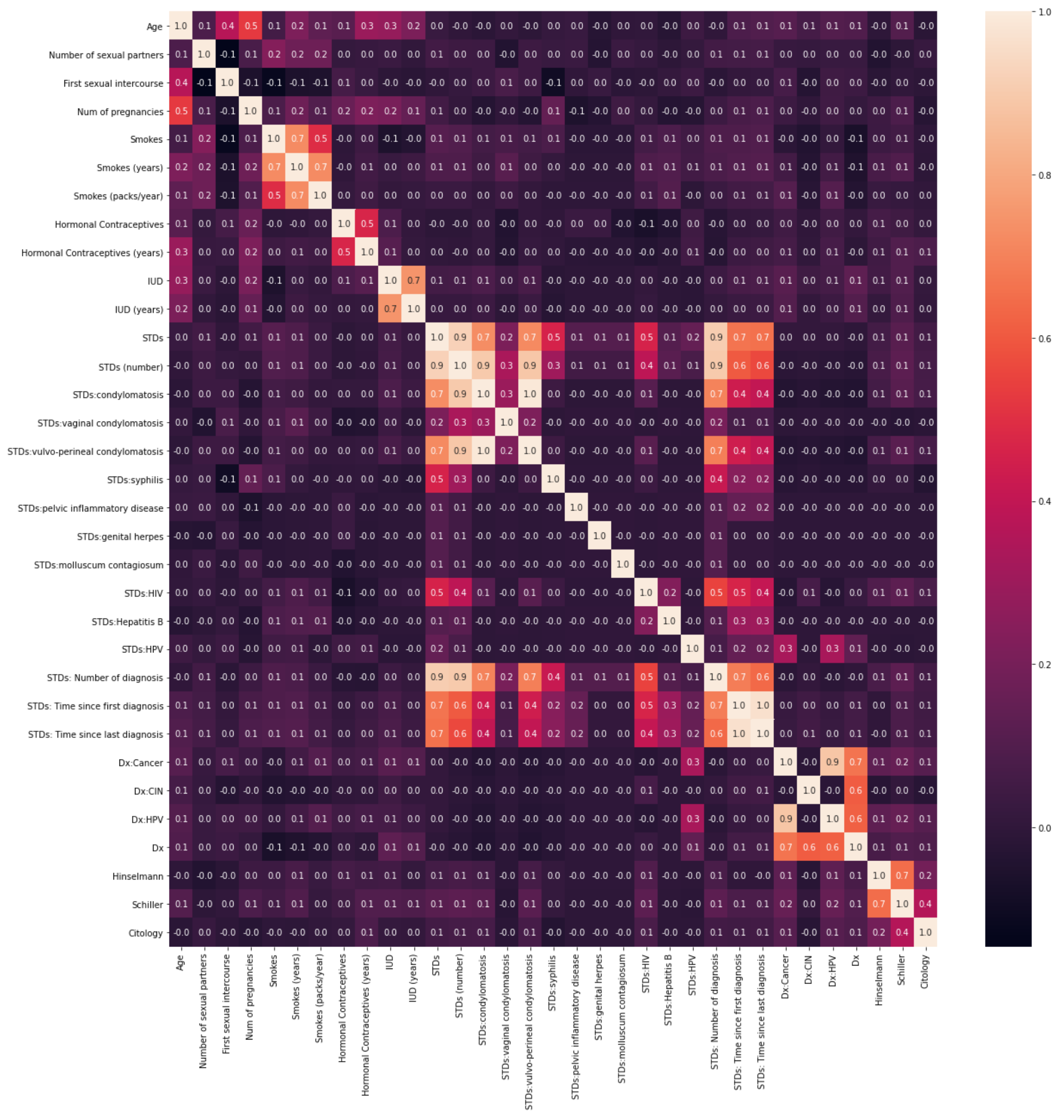
where:

n is the sample size,

x_i, y_i are the individual sample points indexed with i,

$\bar{x}$ and $\bar{y}$ are the sample means.

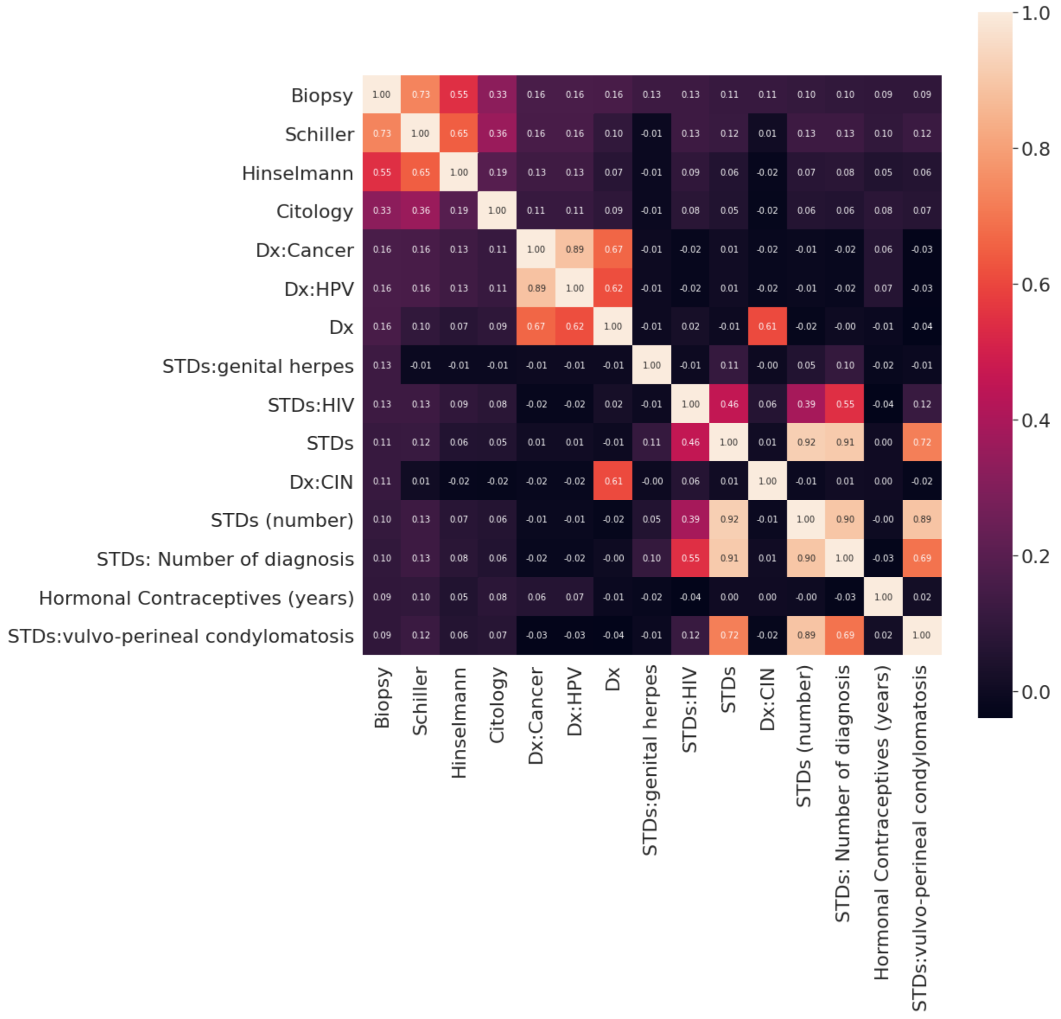
It is a statistic that has a value between +1 and -1. A closer result to 1 means positive linear correlation between the variables, whereas a close result to -1 means negative linear correlation. If instead the correlation is closer to 0, this means that there are no/less relationships between the variables.



From this heatmap, representing only the correlation between the training factors, and not with the target one, it is possible to see that there is strong correlation between some factors, but this is not surprising. For example ‘STDs’, ‘STDs (number)’ and ‘STDs: Number of diagnosis’ all show significant correlation to the individual STD factors, but this would be expected.

Other expected correlations were between ‘Age’ and ‘Number of sexual partners’, ‘First sexual intercourse’, ‘Number of pregnancies’, ‘Smokes(years)’, ‘Hormonal Contraceptives(years)’, ‘IUD’ and ‘IUD(years)’.

Now, in the following heatmap are shown the 15 features with higher correlation with the target variable ‘Biopsy’.



‘Schiller’, ‘Hinselmann’ and ‘Citology’ are features representing medical diagnostic tools used to identify cervical cancer, then it is not surprising that they have the strongest correlations to ‘Biopsy’.

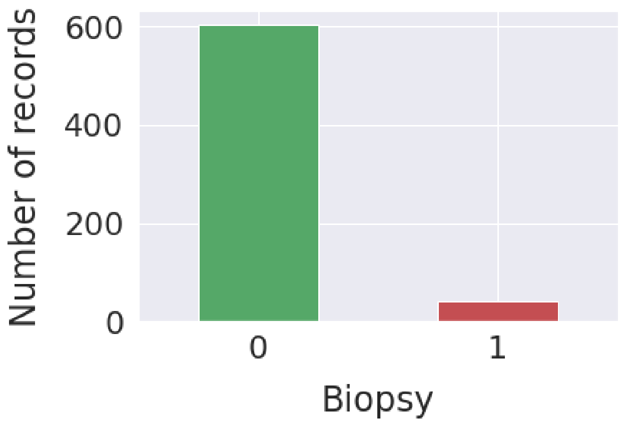
Preprocessing

Training and Test set

The dataset has to be divided into set used for training phase and the set used to test the implemented models.

I chose to consider the 75% of my data as training set, and the remaining part, 25%, as test set. I used ‘stratify’ option to maintain the class proportions in the two subsets.

Below, a count plot of my training set:



Biopsy = 0	602
Biopsy = 1	41

The dataset is unbalanced, as it is possible to seen people with the disease are in a proportion of about 1:15 with healthy individuals.

Most machine learning classification algorithms are sensitive to unbalance in the predictor classes. An unbalanced dataset will bias the prediction model towards the more common class.

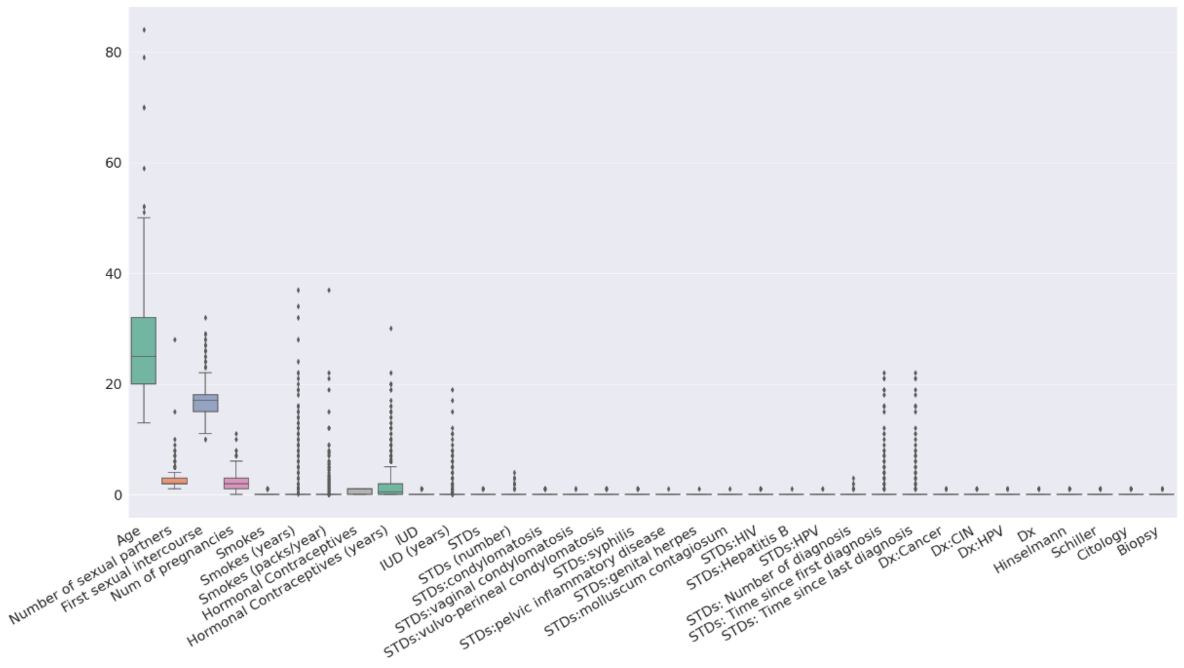
Standardization and Normalization techniques

An important step of preprocessing phase is characterized by normalization and standardization of data.

Standardization and Normalization processes are important for many Machine Learning estimators, because they might behave badly if the individual features do not more or less look like standard normally distributed. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

Standardization is important for many Machine Learning estimators, because they might behave badly if the individual features do not more or less look like standard normally distributed.

To see the ranges of values covered by each of the factors, it is used Box Plot.



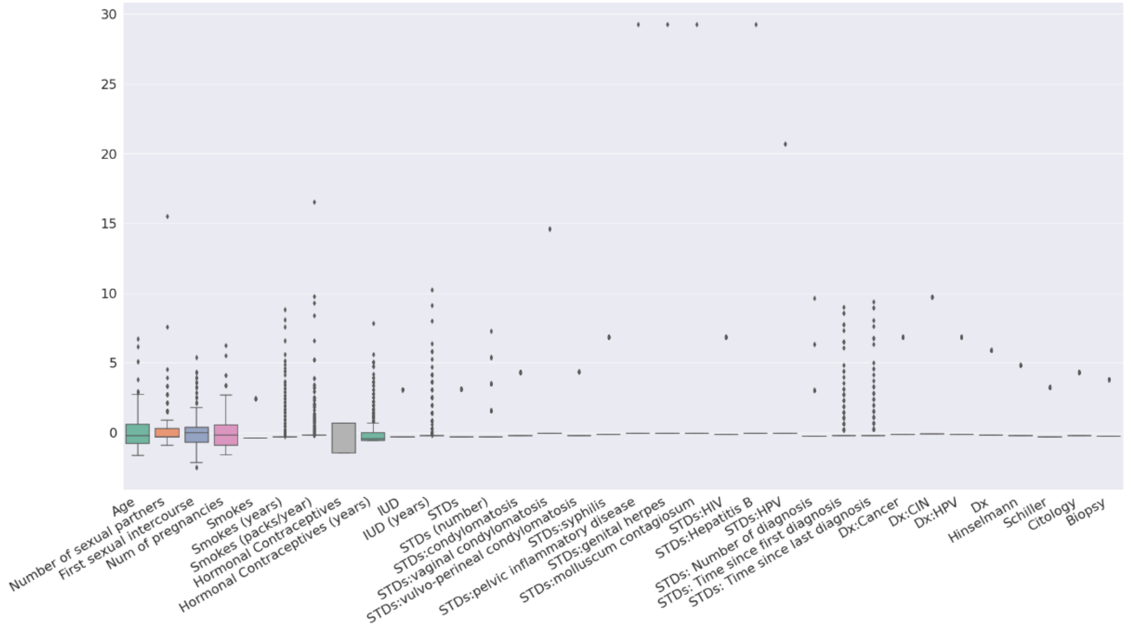
Several feature scaling methods can be applied according to distribution of data. Data are standardized with StandardScaler and MinMaxScaler.

StandardScaler standardizes the features by removing the mean and scaling to unit variance.

The standard score of a sample x is calculated as:

$z = \frac{x-\mu}{s}$ where μ and s are respectively the mean and the standard deviation of the training samples.

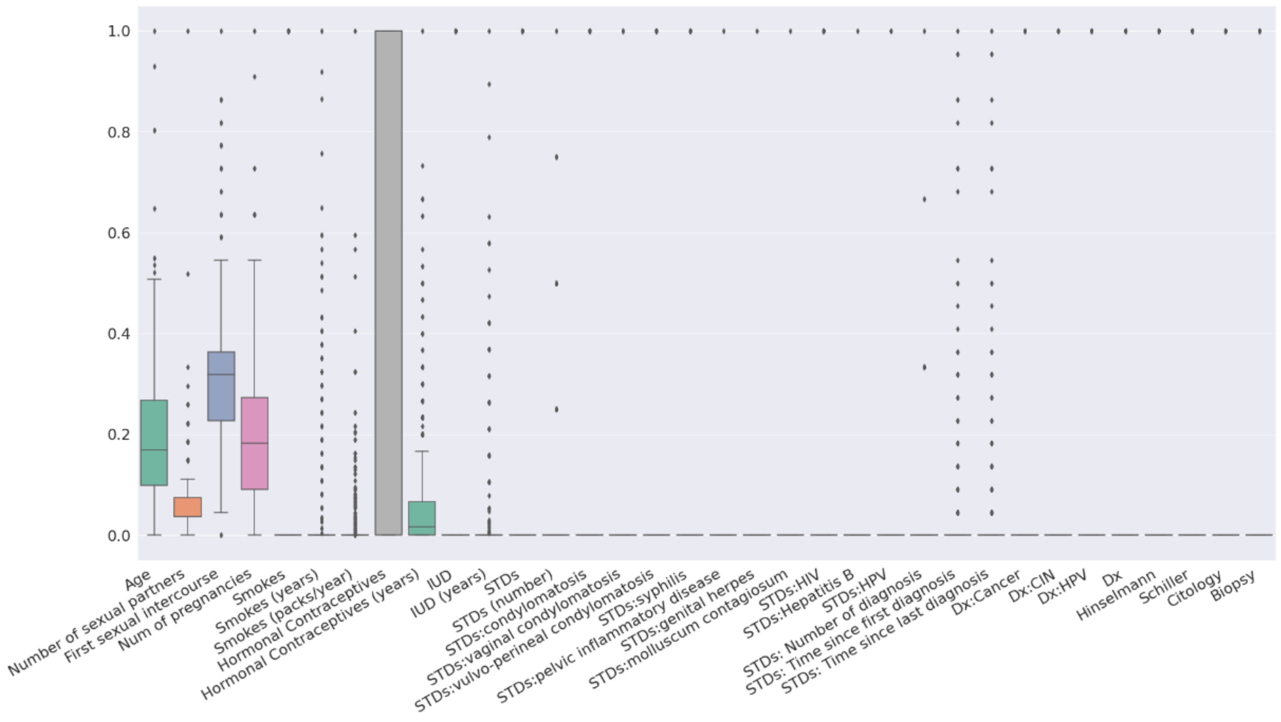
The resulting box plot of the features is the following:



MinMaxScaler represent a way to normalize the input features/variables. All the features are scaled into the range [0,1], with the following formula:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

The box plots of the features after normalization are the following:



In both cases, it is possible to see that features variance is reduced.

Dimensionality Reduction

The number of input variables or features for a dataset is referred to as its dimensionality.

Dimensionality reduction, or dimension reduction, refers to the transformation of data from a high-dimensional space int a low-dimensional space.

Working in high-dimensional spaces can be not very desirable, it can mean that the volume of that space is very large, and in turn, the points that we have in that space often represent a small and non-representative sample.

This can dramatically impact the performance of Machine Learning algorithms fit on data with many input features, generally called *curse of dimensionality*.

Therefore, it is often desirable to reduce the number of input features in order to capture the “essence” of data.

Large number of input features can cause poor performance for Machine Learning algorithm, and dimensionality reduction is a general field of study concerned with reducing the number of input features.

It is important to note that fewer input dimensions often mean correspondingly fewer parameters or a simpler structure in the Machine Learning model, referred to as degrees of freedom. A model with too many degrees of freedom is likely to overfit the training set and then not performing well on new data.

Dimensionality reduction is a data preparation technique performed on data prior to modelling. Furthermore, any dimensionality reduction technique performed on training data, must also be performed on new data, such as test or validation dataset.

PCA - Principal Component Analysis

The idea of PCA is to reduce the number of variables of a data set, while preserving as much information as possible.

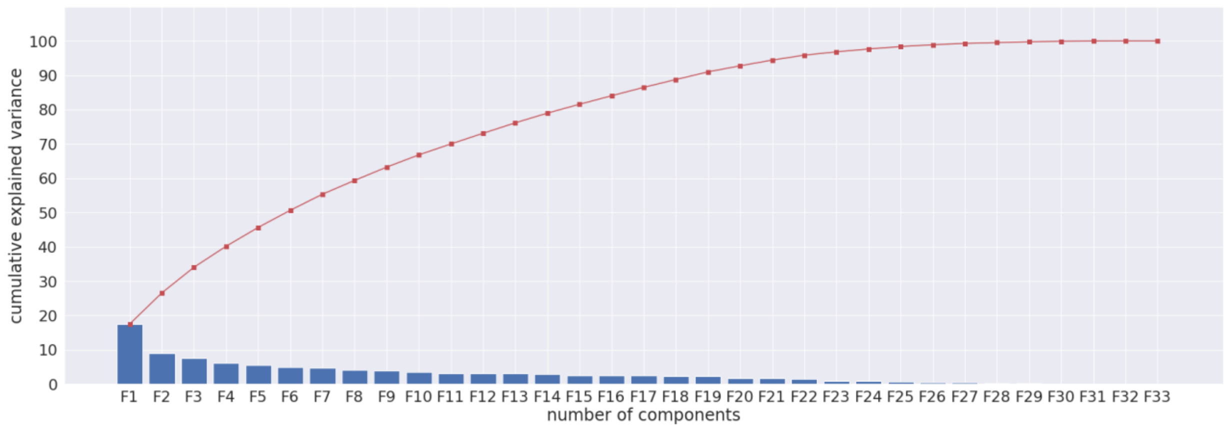
It is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variable into a set of values of linearly uncorrelated variables, called Principal Components. Principal components are constructed in such a manner that the first principal component accounts for the largest possible variance in the data set, and each succeeding component, in turn has the highest possible variance possible under the constraint that is orthogonal to the preceding ones. The resulting vectors are un uncorrelated orthogonal basis set.

The steps involved in PCA are:

1. *Standardization of the dataset*: the procedure is quite sensitive regarding the variances of the initial variables.
2. *Computation of the Covariance matrix*, to understand how the variables of the input fata set are varying from the mean with respect to each other.
3. *Compute the eigenvectors and eigenvalues of the covariance matrix*, in order to determine the principal components.
4. Sort eigenvalues and their corresponding eigenvectors.
5. Pick *k* eigenvalues and forma a matrix of eigenvectors
6. Transform the original matrix

The number *k* of variables to retain mainly depends on the amount of variance that you want to preserve. According to the scree test, we need to find the “elbow” of the graph where the variance seems to level off is found and factors or components to the left of this point should be retained as significant.

Now, with Elbow rule, it is tried to find the right number of components in order to predict the quality label.



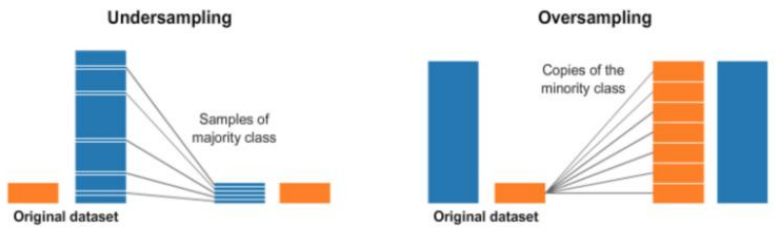
Here, in the graph it is not clear visible an elbow, then it is followed the rule of thumb to take a number of Principal Components in such a way that more than 80% of the cumulative explained variance is taken. With Principal Component Analysis technique are taken 15 out of 33 features are taken.

Imbalanced Data Distribution

In Machine Learning often data sets present a problem of Imbalanced Data Distribution. It generally happens when the observations in one of the class are much higher or lower with respect to the other classes, as in this case.

Standard ML techniques as Logistic Regression, Decision Tree or simplest SVM have a bias towards the majority class and have a poorer predictive accuracy over the minority classes compared to the majority classes. This is because they are designed based on the assumption that the class distribution is relatively balanced and the misclassification costs are equal, classification rules that predict the minority classes tend to be rare, undiscovered or ignored, end test samples belonging to the minority classes are more often misclassified than the samples corresponding to the majority class.

Oversampling and undersampling in data analysis are techniques used to adjust the class distribution of a data set.



Undersampling

Undersampling is a technique that balance the dataset by reducing the dimension of the majority class in order to have all classes balanced.

In this situation, this technique is not a useful way to improve the performance of our model. The class of unhealthy individuals has 41 samples, then our model should consider only 41 samples of the class belonging to healthy individuals. It is not a good idea to discard many samples and train the model on such a small number of samples.

Oversampling

Oversampling is a technique that balance the dataset by increasing the dimension of the minority class in order to have balance.

To do this, it is used SMOTE(synthetic minority oversampling technique), one of the most used oversampling methods.

SMOTE works by selecting minority class examples close in the feature space at random and by finding its k nearest minority class neighbors. The synthetic instance is created by choosing one of the k nearest neighbors at random and connecting neighbors with selected point to form a line segment in feature space. The synthetic instances are generated as a convex combination of the two chosen instances in the feature space.

Considering a sample x_i, a new sample x_{new} will be generated considering its k nearest-neighbors. For each selected nearest neighbor is generated a new sample.

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

where λ is a random number in the range [0,1]. This interpolation will create a sample on the line between x_i and x_{zi}.

More precisely, the steps of SMOTE algorithm are:

1. Selection of the minority class set. For each point belonging to the minority class set, the k-nearest neighbors of that point are obtained by calculating the Euclidian distance between x and every other sample in that set.
2. Setting of the sampling rate N according to the imbalanced proportion. The new set is created by randomly selecting N elements from the k-nearest neighbors of each point belonging to the minority class.
3. For each point belonging to the new set, new data are created by using the formula above.

It is important to remark that the oversampling technique has to be performed during cross-validation process and not before. Synthetic data are created only for the training set without affecting the validation set.

If not, it is probable a data leakage problem, where the validation set was not composed of true value, but of synthetic ones.

Outliers Management

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population.

Outliers significantly affect the process of estimating statistics, resulting in overestimated of underestimated values. Therefore, the results of data analysis are considerably dependent on the ways in which these values are processed.

To identify the outliers in our dataset is used the *IQR rule*.

Any set of data can be described by its five-number summary. The range of a set of data X can be computed as $\max X - \min X$, and it is one indicator of how spread out the data is in a set.

The interquartile range is similar to the range but is less sensitive to outliers and then more helpful.

This range shows how data is spread about the median. It is calculated as the difference between the 75th and the 25th percentiles.

$$IQR = Q_3 - Q_1$$

Once outliers are identified, then they can be treated in 2 different ways:

1. *Outliers elimination*
   The values that do not belong in the interquartile range are dropped from the dataset.
2. *Outliers substitution*
   For each feature the values of the outliers are substituted with the median value.

In this specific data set, removing the outliers or modifying the values they assume, doesn’t bring any advantage.

Outliers are legitimate observation from the considered population, then it is incorrect to eliminate or substitute these values, they may indicate something interesting in the data set and can be helpful in this specific task.

Analysis Methodology

Different approaches of implementation of classifiers models, of the hyper-parametrization, cross-validation, and evaluation techniques are analyzed.

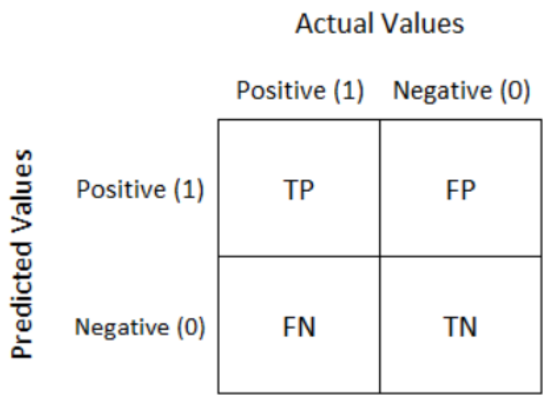
Evaluation Metrics

Evaluation metrics are used to measure the quality of the Machine Learning model. To understand these metrics, some definitions must be given.

- ***True Positive (TP)***: samples for which the prediction is positive, and the true label is positive.
- ***False Positive (FP)***: samples for which the prediction is positive, but the true label is negative.
- ***True Negative (TN)***: samples for which the prediction is negative, and the true label is negative.
- ***False Negative (FN)***: samples for which the prediction is negative, and the true label is positive.

Starting from these metrics, it is possible to define:

- ***Confusion Matrix***:
  It is a performance measurement for Machine Learning classification problem where output can be two or more classes, helpful to define the following measures.

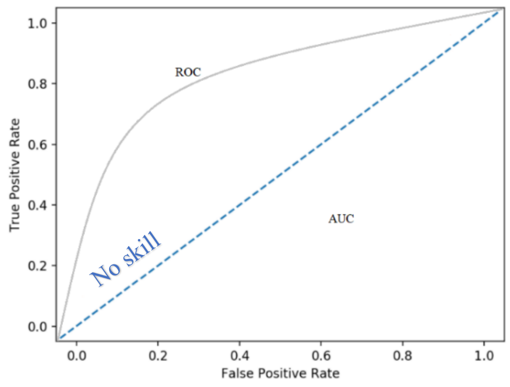


- $\boldsymbol{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$
- $\boldsymbol{Recall\ or\ True\ Positive\ Rate\ (TRP)} = \frac{TP}{TP+FN}$
- $\boldsymbol{Precision} = \frac{TP}{TP+FP}$
- $\boldsymbol{Specificity} = \frac{TN}{TN+FP}$
- $\boldsymbol{False\ Positive\ Rate\ (FPR)} = 1 - specificity = \frac{FP}{FP+TN}$
- $\boldsymbol{F1_{score}} = 2 * \frac{precision*recall}{precision+recall}$

- ***AUC – ROC curve***:

  It is a performance measurement for classification problem at various threshold settings. ROC is a probability curve and AUC represents degree of measure of separability. It tells how much the model is capable of distinguishing between classes.

  It is plotted with TPR against FPR where TPR in on y-axis and FPR on x-axis.

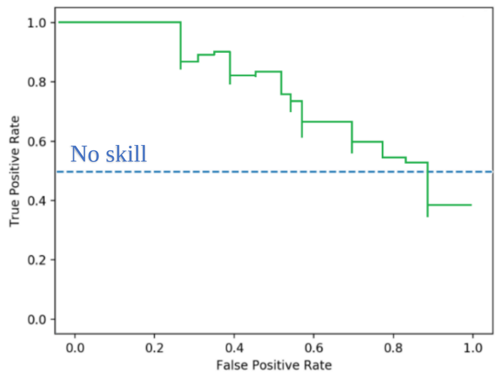


  Higher the AUC, better the model is at predicting, at distinguishing between patients with disease and no disease.

  An excellent model has the AUC near to the 1, which means a good measure of separability, instead a poor model has AUC near to 0.

- **Precision-recall curve**:

  It shows the tradeoff between precision and recall for different threshold. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate.



  A No-skill classifier is one that cannot discriminate between the classes and would predict a random class or a constant class in all cases.

Model Evaluation Process

Learning the parameters of a prediction function and testing it on the same data is a methodology mistake. A model tested on the same data, would have a perfect score on that data, but would fail in predicting anything useful on yet-unseen data. This situation is called *overfitting*.

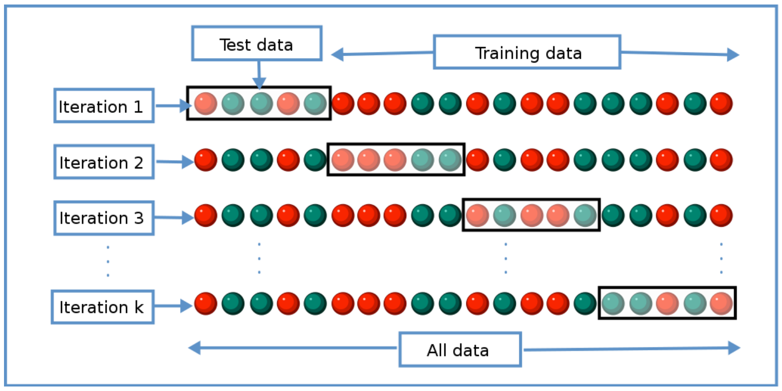
Cross-validation

To tackle this problem there is a procedure called *cross-validation.* The dataset is divided in training dataset, used for the training and test dataset, used for the final evaluation.

K-fold cross-validation is the type of cross-validation used.

The training set is split into K smaller sets, 5 in this case, with an equal number of samples and with the same proportion among the classes.

In the training process K-1 parts are used for the training and the last one is used for the evaluation. This procedure is followed for each of the K folds.



The performance measure reported by k-fold cross-validation is the average of the measures computed in the loop.

Hyperparameter tuning

Estimators have different settings themselves, different hyperparameters used to control the learning process, that can assume different values. *Hyperparameter tuning* is the problem of choosing the set of parameters that better fit the model.

The traditional way of performing hyperparameter optimization is the GridSearch, through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.

Specifically, in this case it is adopted the Grid Search with Cross Validation. The performance metric that guides the grid search is the precision- recall AUC metric.

It is chosen to analyze this metric, because it gives an idea on how many False Positive and how many False negative the model generates.

The searched hyperparameters are those that maximize this metric, and then corresponds to the lower FPR and lower FNR.

Analysis of Classification Models

After Exploratory Data Analysis and Pre-Processing, it is time to build the model training with different classifier and compare the results. For each classifier 6 instances are trained, with different techniques:

- Default model;
- Standardization;
- Oversampling;
- Standardization and Oversampling;
- PCA;
- PCA and Oversampling.

Logistic Regression

Logistic regression is a generalized linear model for classification. Logistic regression fits an S-shaped logistic function, also called Sigmoid, which goes from 0 to 1.

This curve tells which is the conditional probability that in a binary classification problem the class Y is 1, given the predictors x and considering the weights w.

$$p(X) = P(Y = 1|\, x; w) = \frac{e^{\langle w,x \rangle}}{1 + e^{\langle w,x \rangle}}$$

where $h_w(x) = \frac{e^{\langle w,x \rangle}}{1+ e^{\langle w,x \rangle}}$ is the logistic function.

Using the logarithm to transform the logistic function, a new expression is obtained:

$$logit(p) = log\,\frac{p(X)}{1-p(X)} = \langle w, x \rangle$$

This function, also called the logit link, is equal to the inner product between the weights and the predictors. In order to estimate the weights, it is exploited the Maximum Likelihood Estimation, equivalent to minimize the logistic loss function.

$$l\big(h_w, (x, y)\big) = \log(1 + e^{-y\langle w,x \rangle})$$

It gives the probability of observed zeros and ones (classes) in data and parameters are chosen in order to maximize it.

The tuned hyperparameters are:

- *C*: the inverse of regularization strength, where smaller values mean stronger regularization.
- *penalty*: used to specify the norm used in the penalization.

It could be:

- *'l1'*, also called *Lasso Regression*: it adds "absolute value of magnitude" of coefficient as penalty term to the loss function.

The cost function is $\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \frac{1}{C}\sum_{j=1}^{p}\left|\beta_j\right|$

- *'l2'*, also called *Ridge Regression*: it adds "squared magnitude" of coefficient as penalty term to the loss function.

The cost function is $\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \frac{1}{C}\sum_{j=1}^{p}\beta_j^2$

Moreover, for the different implementations of the algorithm, as solver it is used 'saga', that optimizes the sum of a finite number of smooth convex functions, and support both 'l1' and 'l2' penalization.

For each of the instances of the classifier, it is exploited the classification report, on the test data.

Default

	precision	*recall*	*f1-score*	*support*
No disease	0.93	1.00	0.97	201
Disease	0.00	0.00	0.00	14
accuracy			0.93	215
macro avg	0.47	0.50	0.48	215
weighted avg	0.87	0.93	0.90	215

Standardization

	precision	*recall*	*f1-score*	*support*
No disease	0.97	0.97	0.97	201
Disease	0.60	0.64	0.62	14
accuracy			0.95	215
macro avg	0.79	0.81	0.80	215
weighted avg	0.95	0.95	0.95	215

Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.57	0.93	0.70	14
accuracy			0.95	215
macro avg	0.78	0.94	0.84	215
weighted avg	0.97	0.95	0.95	215

Standardization and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.95	0.97	201
Disease	0.58	1.00	0.74	14
accuracy			0.95	215
macro avg	0.79	0.98	0.86	215
weighted avg	0.97	0.95	0.96	215

PCA

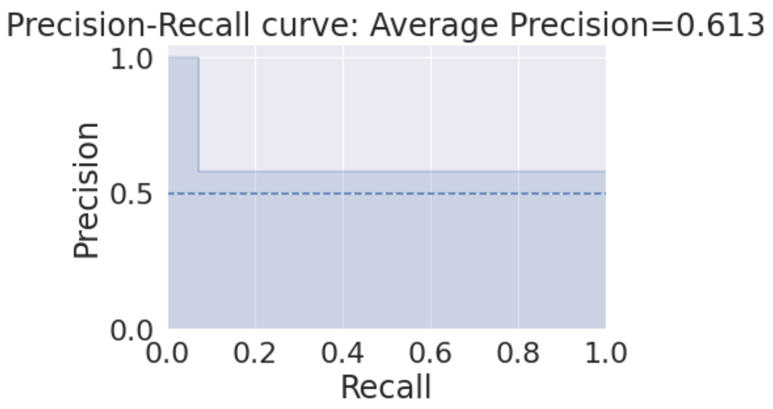
	precision	*recall*	*f1-score*	*support*
No disease	0.94	0.99	0.97	201
Disease	0.50	0.14	0.22	14
accuracy			0.93	215
macro avg	0.72	0.52	0.59	215
weighted avg	0.91	0.93	0.92	215

PCA and Oversampling

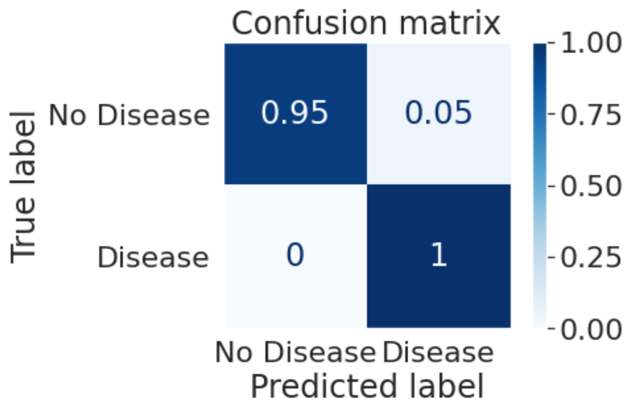
	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.94	0.97	201
Disease	0.52	1.00	0.68	14
accuracy			0.94	215
macro avg	0.76	0.97	0.82	215
weighted avg	0.97	0.94	0.95	215

According to the accuracy metric, the best performing instance of the classifier corresponds to the case in which it is applied Standardization and Oversampling techniques to the training data.

The obtained Precision-Recall curve of this instance on the training data is:



The obtained confusion matrix is:



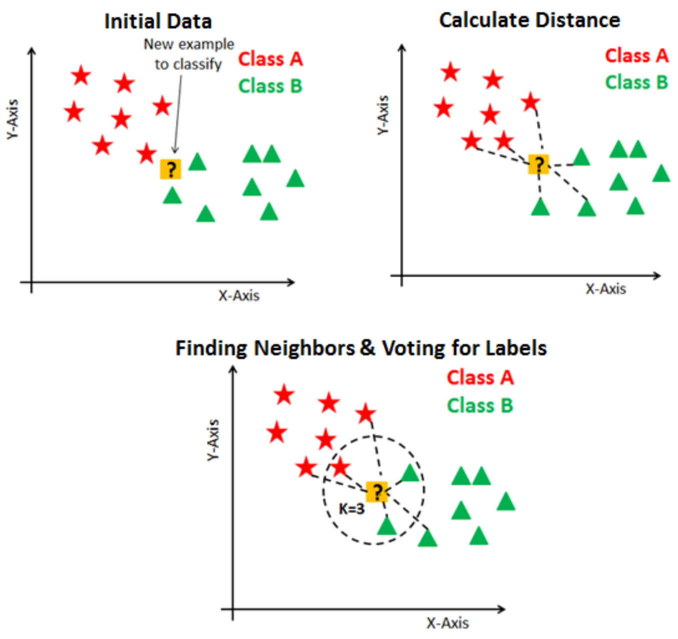
KNN – K nearest neighbors

K nearest neighbors algorithm is a simple non-parametric method. The idea is to memorize the training set and then to predict the label of any new instance on the basis of the labels of its closest neighbors in the training set.

Consider an instance domain X, endowed with a metric function ρ, that returns the distance between any two elements of X.

An example of this function could be the Euclidian distance

$$\rho(x, x') = \|x - x'\| = \sqrt{\sum_{i=1}^{d} (x_i - x_i')^2}$$



Let $S = (x_1, y_1), \ldots, (x_m, y_m)$ be a sequence of training examples. For each $x \in X$, let $\pi_1(x), \ldots, \pi_m(x)$ be a reordering of {1,…, m} according to their distance to x, $\rho(x, x_i)$. That is, for all i < m,

$$\rho\left(x, x_{\pi_i(x)}\right) \leq \rho\left(x, x_{\pi_{i+1}(x)}\right)$$

Therefore, for every point $x \in X$, return the majority label among $\left\{y_{\pi_i(x)} : i \leq k\right\}$.

The tuned hyperparameter is the number of neighbors, to consider in the majority voting of the algorithm.

For each of the instances of the classifier, it is exploited the classification report, on the test data.

Default

	precision	*recall*	*f1-score*	*support*
No disease	0.94	0.98	0.96	201
Disease	0.20	0.07	0.11	14
accuracy			0.92	215
macro avg	0.57	0.53	0.53	215
weighted avg	0.89	0.92	0.90	215

Standardization

	precision	*recall*	*f1-score*	*support*
No disease	0.98	0.97	0.97	201
Disease	0.59	0.71	0.65	14
accuracy			0.95	215
macro avg	0.78	0.84	0.81	215
weighted avg	0.95	0.95	0.95	215

Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.96	0.93	0.94	201
Disease	0.30	0.43	0.35	14
accuracy			0.90	215
macro avg	0.63	0.68	0.65	215
weighted avg	0.92	0.90	0.91	215

Standardization and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.98	0.97	0.97	201
Disease	0.59	0.71	0.65	14
accuracy			0.95	215
macro avg	0.78	0.84	0.81	215
weighted avg	0.95	0.95	0.95	215

PCA

	precision	*recall*	*f1-score*	*support*
No disease	0.98	0.97	0.97	201
Disease	0.59	0.71	0.65	14
accuracy			0.95	215
macro avg	0.78	0.84	0.81	215
weighted avg	0.95	0.95	0.95	215

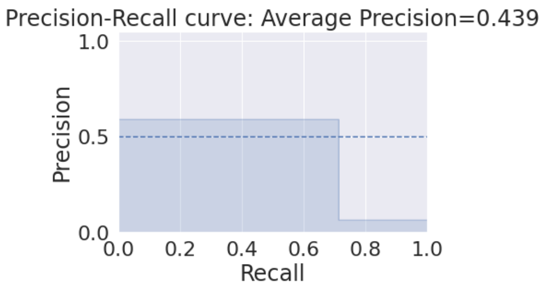
PCA and Oversampling

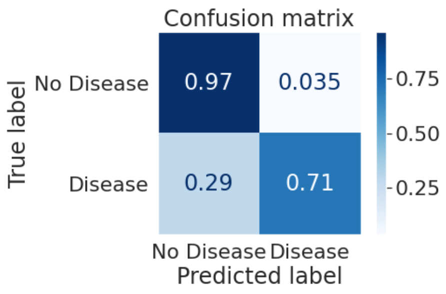
	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.94	0.97	201
Disease	0.52	1.00	0.68	14
accuracy			0.94	215
macro avg	0.76	0.97	0.82	215
weighted avg	0.97	0.94	0.95	215

The instance of K-Nearest Neighbors with PCA and the one with Standardization, presents the same results, higher than the others.

It could be also noted that the oversampling of the minority class, decrease the performances on the 'No disease' class, while upgrading the performances on the 'Disease' class, then it is discouraged the use of this approach.

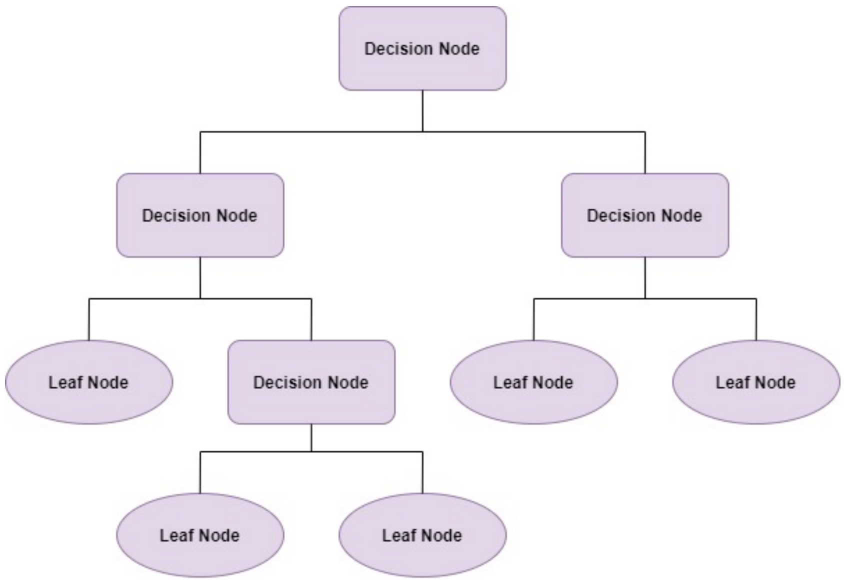
The Precision-Recall curve and the Confusion Matrix are shown for the instance with PCA:





Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions.



- Each internal node of the tree represents a “test” on an attribute;
- Each branch represents the outcome of the test;
- Each leaf node represents a class label;
- The path from root to leaf represent classification rules.

This method divides the predictor space in simple regions basing on features values: the predictor space, so the set of possible values $x_1, x_2, \dots, x_j$ is divided in j non overlapping regions $R_1, R_2, \dots R_j$.

It is used a greedy approach for this algorithm. The algorithm starts from a first node in the tree and then at each step the predictor space is split, by choosing the best split in that level rather than looking ahead to find the split that will lead to a better final result.

It is possible to use one of these two measures to choose the best split:

- *Gini index*:

  Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen.

  The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

where p_i is the probability of an object being classified to a particular class.

It is preferred choosing the attribute/feature with the least Gini index as the root node.

- *Entropy measure*:

  Entropy is a measure of disorder. It is used to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is equally divided it has entropy of one.

$$Entropy = \sum_{i=1}^{n} -p_i \log_2 p_i$$

Decision trees are simple to understand and to interpret, requires a little data preparation and have a cost of using it that is logarithmic in the number of data point used to train the tree.

However, these learners can create over-complex trees that do not generalize the data well. This problem is called overfitting, and could be solved with pruning mechanisms, by setting for example the minimum number of samples requires at a leaf node or setting the maximum depth of the tree.

The tuned hyperparameters are:

- *criterion*: Gini index or entropy;
- *max_depth*: the maximum depth of the tree

For each of the instances of the classifier, it is exploited the classification report, on the test data.

Default

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.96	0.98	201
Disease	0.64	1.00	0.78	14
accuracy			0.96	215
macro avg	0.82	0.98	0.88	215
weighted avg	0.98	0.96	0.97	215

Standardization

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.96	0.98	201
Disease	0.64	0.71	0.78	14
accuracy			0.96	215
macro avg	0.82	0.98	0.88	215
weighted avg	0.98	0.96	0.97	215

Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.95	0.97	201
Disease	0.56	1.00	0.72	14
accuracy			0.95	215
macro avg	0.78	0.97	0.84	215
weighted avg	0.97	0.95	0.96	215

Standardization and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.95	0.97	201
Disease	0.56	1.00	0.72	14
accuracy			0.95	215
macro avg	0.78	0.97	0.84	215
weighted avg	0.97	0.95	0.96	215

PCA

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.91	0.95	201
Disease	0.39	0.86	0.53	14
accuracy			0.90	215
macro avg	0.69	0.88	0.74	215
weighted avg	0.95	0.90	0.92	215

PCA and Oversampling

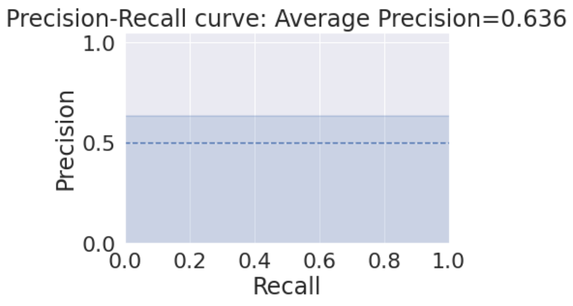
	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.94	0.96	201
Disease	0.48	0.86	0.62	14
accuracy			0.93	215
macro avg	0.73	0.90	0.79	215
weighted avg	0.96	0.93	0.94	215

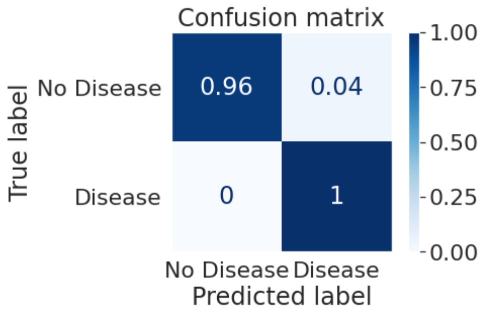
Decision Trees frequently perform well on imbalanced data. They work by learning a hierarchy of if/else questions and this can force both classes to be addressed. Indeed, Oversampling lowered the value of the accuracy of a bit.

It is evident, as it was expected, that Standardization or Oversampling technique does not improve the results, because Decision Tree consists of comparisons and branching down the tree, so it would not help.

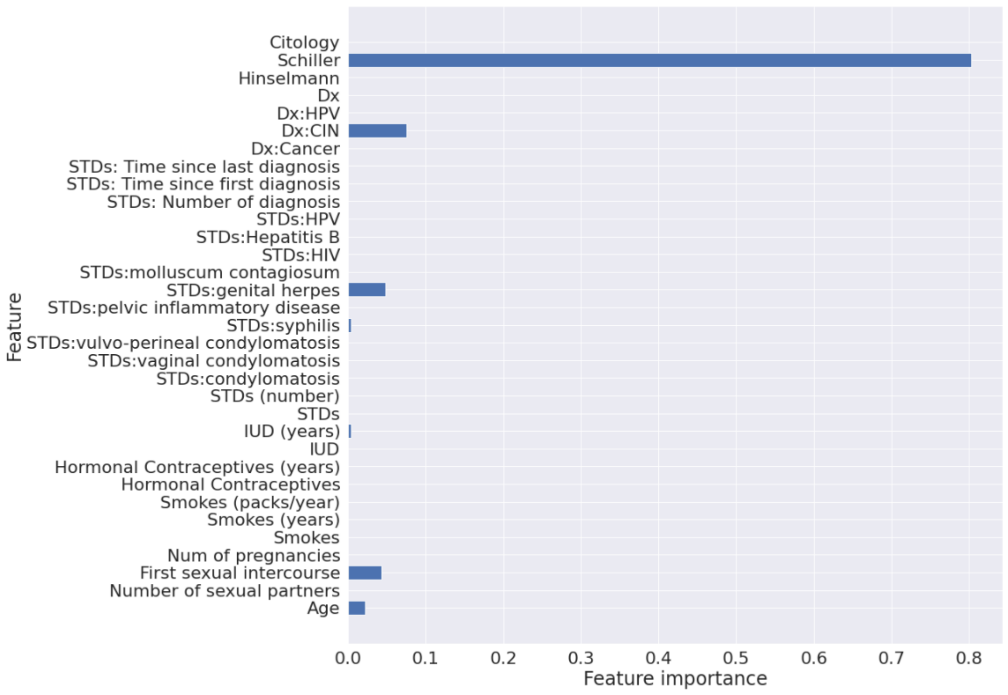
None of the discussed techniques improve Decision Tree, then the best performing instance found is the default one.

The Precision-Recall curve and the Confusion Matrix matching this instance are:





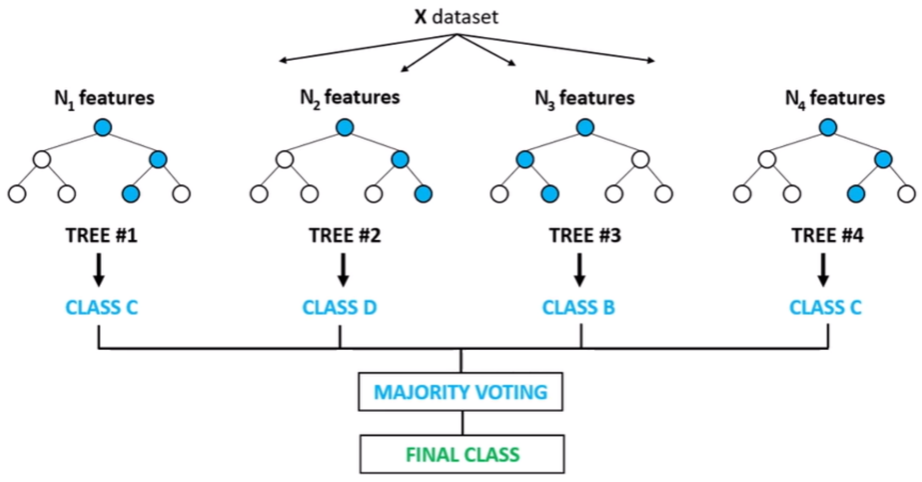
It is also possible to show which of the features are the most important ones in decision tree algorithm:



Random Forest

Random Forest or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time. For each candidate split in the learning process, it selects a random subset of the features, typically $\sqrt{p}$, being p the total number of features.

The prediction of the random forest is obtained by a majority vote over the predictions of the individual trees.



Random Forest is an improvement of Decision Trees.

In particular, trees that are grown very deep tend to learn highly irregular patterns: they overfit their training sets, i.e. have low bias, but very high variance. Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model. While the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners, that is the following:

Given a training set $X = x_1, \dots, x_n$ with responses $Y = y_1, \dots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, …, B:

- Sample, with replacement, n training examples from X, Y, called X_b, Y_b.
- Train a classification tree f_b on X_b, Y_b.

After the training, predictions for the unseen samples x' can be made by taking the majority vote among all the predicted values of the trees.

Random forests differ in only one way from this general scheme: by applying feature bagging procedure.

It is used a modified tree learning algorithm that selects, at each candidate split in the learning process, a different random subset of the features.

The tuned hyperparameters are the same of Decision Tree algorithm, except

- *n_estimators:* it indicates the number of trees in the forest.

For each of the instances of the classifier, it is exploited the classification report, on the test data.

Default

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.95	0.97	201
Disease	0.56	1.00	0.72	14
accuracy			0.95	215
macro avg	0.78	0.97	0.84	215
weighted avg	0.97	0.95	0.96	215

Standardization

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.95	0.97	201
Disease	0.56	1.00	0.72	14
accuracy			0.95	215
macro avg	0.78	0.97	0.84	215
weighted avg	0.97	0.95	0.96	215

Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.54	0.93	0.68	14
accuracy			0.94	215
macro avg	0.77	0.94	0.83	215
weighted avg	0.97	0.94	0.95	215

Standardization and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.54	0.93	0.68	14
accuracy			0.94	215
macro avg	0.77	0.94	0.83	215
weighted avg	0.97	0.94	0.95	215

PCA

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.94	0.97	201
Disease	0.52	1.00	0.68	14
accuracy			0.94	215
macro avg	0.76	0.97	0.82	215
weighted avg	0.97	0.94	0.95	215

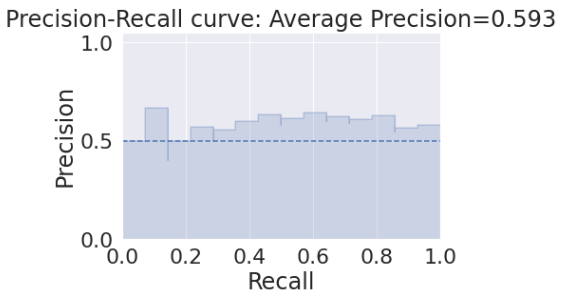
PCA and Oversampling

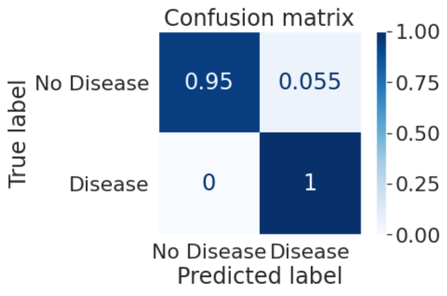
	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.94	0.96	201
Disease	0.50	0.93	0.65	14
accuracy			0.93	215
macro avg	0.75	0.93	0.81	215
weighted avg	0.96	0.93	0.94	215

The same considerations done on Oversampling and Standardization, obviously also apply to Random Forest algorithm.

Also in this case, the best accuracy performance is obtained in correspondence of the default configuration of the classifier.

The Precision-Recall curve and the Confusion Matrix matching the default instance are:





SVM

Support-Vector Machines are supervised models with associated learning algorithms that analyze data used classification and regression analysis.

A support-vector constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used in learning algorithms. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin) since in general the larger the margin, the lower the generalization of the classifier.

Linear SVM

The linear SVM goal in its hard margin formulation is to find a hyperplane in the feature space that correctly separates all the classes with the largest possible margin.

The equation of the plane is $\omega^T + b = 0$, where $\vec{\omega}$, b are defined in order to maximize its distance to the nearest points from either group.

For a binary classification, it is possible to note that the support vector has two possible solutions: $\omega^T + b = \pm 1$ where the margin is given by $\frac{2}{||\omega||}$.

Then, given the constraint that, the optimization problem can be defined as:

$$\arg\min \frac{\|\vec{\omega}\|^2}{2} \text{ subject to } y_i(\langle x_i, \omega \rangle + b) \geq 1$$

The limitation of this model is linear separability. If data are not linearly separable, then no feasible solution is found.

Soft-Margin SVM

The solution in case of no linearly separable data is the soft-margin SVM, which consists of introducing a slack variable ξ for each sample.

Then, the optimization problem is:

$$arg \min \frac{\|\vec{\omega}\|^2}{2} + C \sum \xi_i \quad \text{subject to } y_i(\langle x_i \omega \rangle + b) \geq 1 - \xi_i \text{ and } \xi_i \geq 0$$

where C determines the trade-off between increasing the margin size and ensuring that the x_i lie on the correct side of the margin.

The only parameter to tune in this case is C.

Default

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.57	0.93	0.70	14
accuracy			0.95	215
macro avg	0.78	0.94	0.84	215
weighted avg	0.97	0.95	0.95	215

Standardization

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.96	0.97	201
Disease	0.59	0.93	0.72	14
accuracy			0.95	215
macro avg	0.79	0.94	0.85	215
weighted avg	0.97	0.95	0.96	215

Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.92	0.96	201
Disease	0.45	0.93	0.60	14
accuracy			0.92	215
macro avg	0.72	0.92	0.78	215
weighted avg	0.96	0.92	0.93	215

Standardization and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.54	0.93	0.68	14
accuracy			0.94	215
macro avg	0.77	0.94	0.83	215
weighted avg	0.97	0.94	0.95	215

PCA

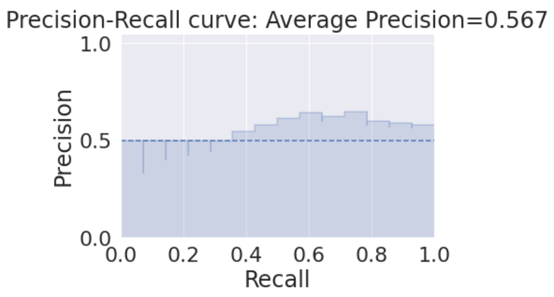
	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.96	0.97	201
Disease	0.59	0.93	0.72	14
accuracy			0.95	215
macro avg	0.79	0.94	0.85	215
weighted avg	0.97	0.95	0.96	215

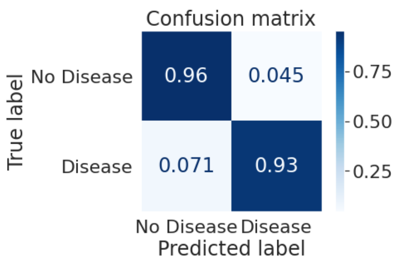
PCA and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.54	0.93	0.68	14
accuracy			0.94	215
macro avg	0.77	0.94	0.83	215
weighted avg	0.97	0.94	0.95	215

The instance with PCA, and the one with standardized data, are the ones obtaining higher accuracy.

The obtained Precision-Recall curve and Confusion Matrix for the PCA instance of the Linear SVM are:





Kernel trick

Kernel methods take their name from the use of the kernel functions. Kernel functions enable to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space.

Moreover, this operation is often computationally cheaper than the explicit computation of the coordinates.

$$K(x, x') = \langle \psi(x), \psi(x') \rangle$$

A symmetric function $K: X \times X \rightarrow \mathbb{R}$ can be a kernel function if and only if it respects the *Mercer Theorem.* The theorem says that the Gram matrix $G_{i,j} = K(x_i, x_j)$ needs to be positive semidefinite, therefore $x^T G x \geq 0, \forall x \in R^n$ and the eigenvalues are non-negative.

The two most used kernels are:

- $K(x, x') = (\langle x, x' \rangle + 1)^p$, *polynomial kernel* with p degree.
- $K(x, x') = e^{-\gamma \|x - x'\|^2}$, *RBF kernel.*

The exploited kernel in this case is the RBF kernel. Therefore, the parameters to tune in this case are the regularization parameter C and the kernel coefficient γ.

For each of the instances of the classifier, it is exploited the classification report, on the test data.

Default

	precision	*recall*	*f1-score*	*support*
No disease	0.93	1.00	0.97	201
Disease	0.00	0.00	0.00	14
accuracy			0.93	215
macro avg	0.47	0.50	0.48	215
weighted avg	0.87	0.93	0.90	215

Standardization

	precision	*recall*	*f1-score*	*support*
No disease	0.95	0.98	0.97	201
Disease	0.50	0.29	0.36	14
accuracy			0.93	215
macro avg	0.73	0.63	0.66	215
weighted avg	0.92	0.93	0.93	215

Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.57	0.93	0.70	14
accuracy			0.95	215
macro avg	0.78	0.94	0.84	215
weighted avg	0.97	0.95	0.95	215

Standardization and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.98	0.97	0.97	201
Disease	0.62	0.71	0.67	14
accuracy			0.95	215
macro avg	0.80	0.84	0.82	215
weighted avg	0.96	0.95	0.95	215

PCA

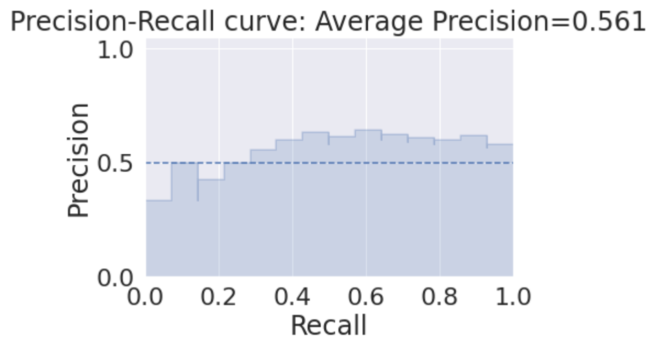
	precision	*recall*	*f1-score*	*support*
No disease	0.95	0.98	0.97	201
Disease	0.50	0.29	0.36	14
accuracy			0.93	215
macro avg	0.73	0.63	0.66	215
weighted avg	0.92	0.93	0.93	215

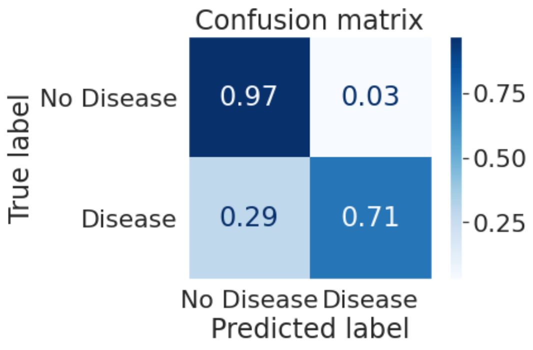
PCA and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.98	0.97	0.97	201
Disease	0.62	0.71	0.67	14
accuracy			0.95	215
macro avg	0.80	0.84	0.82	215
weighted avg	0.96	0.95	0.95	215

The best performing instance of the model is the one in which Standardization and Oversampling or PCA and Oversampling are applied to the training data.

The obtained Precision-Recall curve and the Confusion Matrix obtained in correspondence of PCA and Oversampling instance are:





Gradient Tree Boosting

Gradient boosting is a Machine Learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

The idea of gradient boosting is that boosting can be interpreted as an optimization algorithm on a suitable cost function.

Given a dataset $\{(x_1, y_1), \dots, (x_n, y_n)\}$ of known values of x and corresponding label y, the goal of this technique is to find an approximation $\hat{F}(x)$ to a function $F(x)$ that minimizes the expected value of some specified loss function $L\big(y, F(x)\big)$:

The gradient boosting method assumes a real-valued y and seeks an approximation $\hat{F}(x)$, that minimizes the average value of the loss function on the training set, i.e., minimizes the empirical risk.

The idea is to apply the steepest descent step to the minimization problem

It starts from a constant function $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$, and incrementally expands it in a greedy fashion:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

where:

- $h_m \in H$ is a base learner function, that in this specific case corresponds to a decision tree.
- $\gamma_m = \underset{y}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$ is the step length.

Loss function is minimized with a line search technique.

The tuned hyperparameters for the Gradient Tree Boosting are:

- *Loss*: the loss function to be optimized.
  ‘deviance’: A negative multinomial log-likelihood loss function for multi-class classification providing probability estimates:
- *Learning Rate*: the γ_m parameter which scales the step length of the gradient descent procedure. It is strongly correlated with the parameter n_estimators, because small values of learning rate require larger numbers of weak learners to maintain a constant training error;
- *Number of estimators*: The number of boosting stages to perform;
- *Criterion*: the function to measure the quality of a split
  - ‘Friedman mse’: this splitting criterion allow us to take the decision not only on how close we're to the desired outcome (mse), but also based on the probabilities of the desired k-class that we're going to find in the region l or in the region r;
  - ‘mse’: Mean Squared Error

- ‘mae’: Mean Absolute Error

For each of the instances of the classifier, it is exploited the classification report, on the test data.

Default

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.97	0.98	201
Disease	0.63	0.86	0.73	14
accuracy			0.96	215
macro avg	0.81	0.91	0.85	215
weighted avg	0.97	0.96	0.96	215

Standardization

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.97	0.98	201
Disease	0.65	0.93	0.76	14
accuracy			0.96	215
macro avg	0.82	0.95	0.87	215
weighted avg	0.97	0.96	0.97	215

Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.96	0.98	201
Disease	0.61	1.00	0.76	14
accuracy			0.96	215
macro avg	0.80	0.98	0.87	215
weighted avg	0.97	0.96	0.96	215

Standardization and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	1.00	0.95	0.97	201
Disease	0.56	1.00	0.72	14
accuracy			0.95	215
macro avg	0.78	0.97	0.84	215
weighted avg	0.97	0.95	0.96	215

PCA

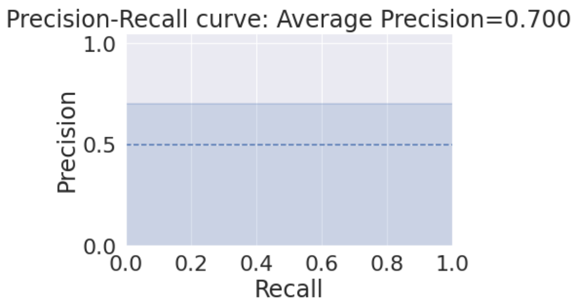
	precision	*recall*	*f1-score*	*support*
No disease	0.95	0.97	0.96	201
Disease	0.40	0.29	0.33	14
accuracy			0.93	215
macro avg	0.68	0.63	0.65	215
weighted avg	0.92	0.93	0.92	215

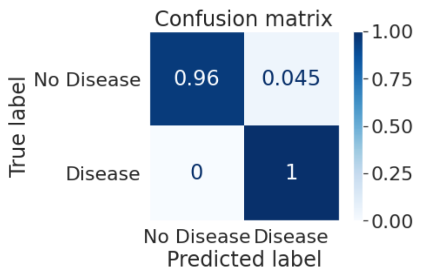
PCA and Oversampling

	precision	*recall*	*f1-score*	*support*
No disease	0.99	0.95	0.97	201
Disease	0.52	0.86	0.65	14
accuracy			0.94	215
macro avg	0.76	0.90	0.81	215
weighted avg	0.96	0.94	0.95	215

The best performing instance of the model is the one in which Standardization or Oversampling are applied to the training data.

The obtained Precision-Recall curve and the Confusion Matrix obtained in correspondence of Oversampling instance are:


Precision-Recall curve: Average Precision=0.700
Precision
Recall


Confusion matrix
True label
No Disease
Disease
No Disease
Disease
Predicted label