UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Machine Learning I

Bachelor's Degree in Data Science and Engineering

Comparative study of machine learning models for the detection of heart diseases

Authors:
Laia Mogas Pladevall
Roger Bargalló Roselló

Teachers:
Marta Arias Vicente
Alexis Molina Martínez de los Reyes

Spring Semester 2023-2024

Resum

La medicina preventiva és clau en la reducció de morts per cardiopaties. Aquest projecte té com a objectiu avaluar diferents models d'aprenentatge automàtic per a la seva detecció. El dataset utilitzat ha estat obtingut a partir de Heart disease de UCI Machine Learning. Conté 76 variables de 617 pacients relacionades amb problemes cardíacs. Després de preprocessar degudament les dades, els següents models supervisats han estat ajustats amb amb els millors paràmetres segons la F2-score: QDA, LDA, Naive Bayes, Regressió Logística, KNN, Random Forest, Gradient Boosting i SVC. Tenint en compte les mètriques i propietats dels models, Gradient Boosting és el model més avantatjós. Posteriorment, s'han eliminat variables de poca importància. Al test s'ha obtingut un F2-score = 0.8923 i una accuracy = 0.8381.

Abstract

Preventive medicine is key in reducing deaths from heart disease. This project aims to evaluate different machine learning models for its detection. The dataset used has been obtained from the UCI Machine Learning Heart disease dataset. It contains 76 variables of 617 patients related to heart problems. After properly preprocessing the data, the following supervised models have been fitted with the best parameters according to the F2-score: QDA, LDA, Naive Bayes, Logistic Regression, KNN, Random Forest, Gradient Boosting, and SVC. Considering the metrics and properties of the models, Gradient Boosting has proven to be the most advantageous model. Subsequently, unimportant variables have been removed. In the test, an F2-score = 0.8923 and an accuracy = 0.8381 were obtained.

Contents

1 Introduction

The World Health Organization estimates that 17.9 million deaths result from cardiovascular diseases, making them the leading cause of death globally[1]. In the current context, there is an increasing emphasis on preventive medicine, making it crucial to have an efficient system for the early detection of heart diseases to prevent long-term fatal consequences.

In this project, we will evaluate the predictive capability of different supervised learning models in the diagnosis of heart diseases. Firstly, extensive data preprocessing will be carried out to optimize the data for analysis. Subsequently, various supervised models will be trained and the best performing model will be selected based on several metrics, prioritizing the F2-score. Next, we will attempt to improve the chosen model to achieve maximum performance and estimate the generalization error.

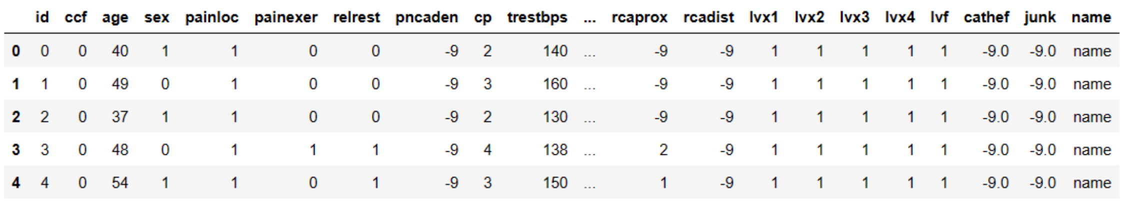
The dataset used is Heart disease - UCI Machine Learning Repository, published on 06/30/1988. It consists of data from four medical centers in Europe and the United States:

1. Cleveland Clinic Foundation (cleveland.data)
2. Hungarian Institute of Cardiology, Budapest (hungarian.data)
3. V.A. Medical Center, Long Beach, CA (long-beach-va.data)
4. University Hospital, Zurich, Switzerland (switzerland.data)

Firstly, we note that the file cleveland.data is corrupted, as it contains unreadable characters. This issue is mentioned in a dataset annotation. Therefore, we will have to limit our analysis to the other medical centers. Since all the datasets contain the same 76 variables A, we can combine the remaining three datasets into one. Unfortunately, the files are not structured as we would like, so we will create a script to properly structure the files. Once they are well-structured, we will combine them.

It is worth noting that we had considered using two hospitals for training and the remaining hospital for testing. However, since they come from different countries with different socioeconomic levels and healthcare systems, their distributions are likely not the same. Consequently, any model we propose would likely perform poorly on the test set. Conversely, if all the hospitals were from the same country, we would likely have applied this methodology.

In the combined dataset (hereafter simply referred to as the dataset), we have 617 rows and 76 variables.



	id	ccf	age	sex	painloc	painexer	relrest	pncaden	cp	trestbps	...	rcaprox	rcadist	lvx1	lvx2	lvx3	lvx4	lvf	cathef	junk	name
0	0	0	40	1	1	0	0	-9	2	140	...	-9	-9	1	1	1	1	1	-9.0	-9.0	name
1	1	0	49	0	1	0	0	-9	3	160	...	-9	-9	1	1	1	1	1	-9.0	-9.0	name
2	2	0	37	1	1	0	0	-9	2	130	...	-9	-9	1	1	1	1	1	-9.0	-9.0	name
3	3	0	48	0	1	1	1	-9	4	138	...	2	-9	1	1	1	1	1	-9.0	-9.0	name
4	4	0	54	1	1	0	1	-9	3	150	...	1	-9	1	1	1	1	1	-9.0	-9.0	name

Figure 1: Head of the dataset

[1] World Health Organization on Cardiovascular Diseases

2 Data exploration and preprocessing

2.1 Feature engineering

The target variable originally took four values based on the severity of the heart disease, with 0 being the only value described in the repository (absence of heart disease). Upon reviewing the documentation, it only referenced the following binary classification: *Value 0: < 50% diameter narrowing* and *Value 1: > 50% diameter narrowing.*

Due to the lack of information regarding values 1, 2, 3, 4, we have chosen to convert the problem into a binary classification problem, where values 2, 3, and 4 are mapped to 1, considering 0 as absence of disease and 1 as presence.

On the other hand, we also had two closely related variables: *rldv5* and *rldv5e*, where one measures height (presumably of some measurement peak) normally and the other after exercising. In this case, what provides valuable information for predictions is the difference between them. Therefore, we will discard the variable *rldv5e* and create the variable *diff_rldv5*, which will be the difference between them.

2.2 Elimination of variables

In the same documentation, some variables were indicated as 'dummy, 'not used' or 'irrelevant' without any additional explanation. Others included no information, and we could not determine what they represent. As a precaution, we decided to discard all these variables. We also removed variables that indicated test dates or the column with patient names. Additionally, we have some variables that do not take values according to their description, such as *proto*, which according to its description should be categorical, but we see that it takes different values than indicated. Consequently, and given that we had more than 70 variables, we decided to eliminate them.

2.3 Missing values

From data exploration, we found that missing values are not consistently indicated. The documentation states that some missing values are indicated with a -9. However, we noticed that some were indicated with 0. We will replace all these values with NaNs.

Since we will be performing value imputation later, we have removed all numerical variables with more than 30% NaNs. For categorical variables, we will not impute but rather remove rows with NaNs. Therefore, we have been more stringent with the number of NaNs in categorical variables, allowing a maximum of 10% NaNs. It is worth noting that many of these variables would have been very useful for the task, such as cholesterol levels or whether the person is a smoker. Unfortunately, the dataset is of lower quality than desired.

2.4 Outliers

Given that this is a dataset of heart diseases and unusual situations can occur, we have been relatively permissive with outliers. However, as will be seen later, we obtained similar results with models sensitive to outliers and more robust models.

Nonetheless, in the case of the variable *met*, the difference is too drastic. After consulting references, we discovered that more than 25 *mets* is infeasible for humans. In fact, we found values in the dataset that greatly exceed this threshold. Therefore, we considered these to be measurement errors or incorrect units. We decided to mark them as NaNs to impute them later instead of removing the instances, as we have a fairly limited amount of data.

The variable *tpeakbpd* also has an unlikely value because it is too low. In this case, we just remove it.

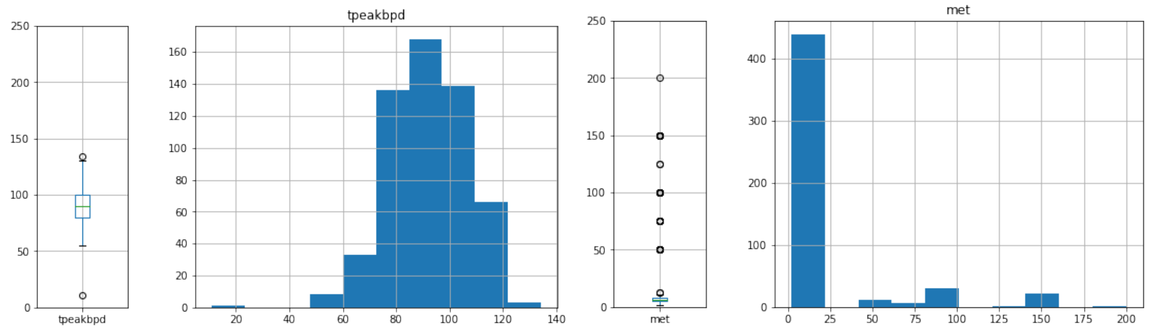


Figure 2: Histograms of tpeakbpd and met

2.5 Distribution of variables

It is convenient to look at the distributions of the variables to get an idea of the impact they might have on the models.

Clearly, we observe that the distribution of some numerical variables differs between those with the disease and those without. The most evident distribution differences between those with and without the disease are found in *age, thaldur, thalach, thalrest.*

Regarding the categorical variables, *painexer* and *relrest* initially seem to be good indicators of disease, as the proportion of those with the disease varies significantly depending on whether the value is 0 or 1.

On the other hand, we see that some categorical variables are unbalanced, such as *cp*, *painloc*, and *sex*. In the case of *sex*, we observe that women are underrepresented. Additionally, there are more healthy women than women with the disease, while for men it is the opposite. We need to be careful that our models do not underdiagnose women.

As for the target variable, *num*, we see that it is fairly balanced with 304 positives and 220 negatives.

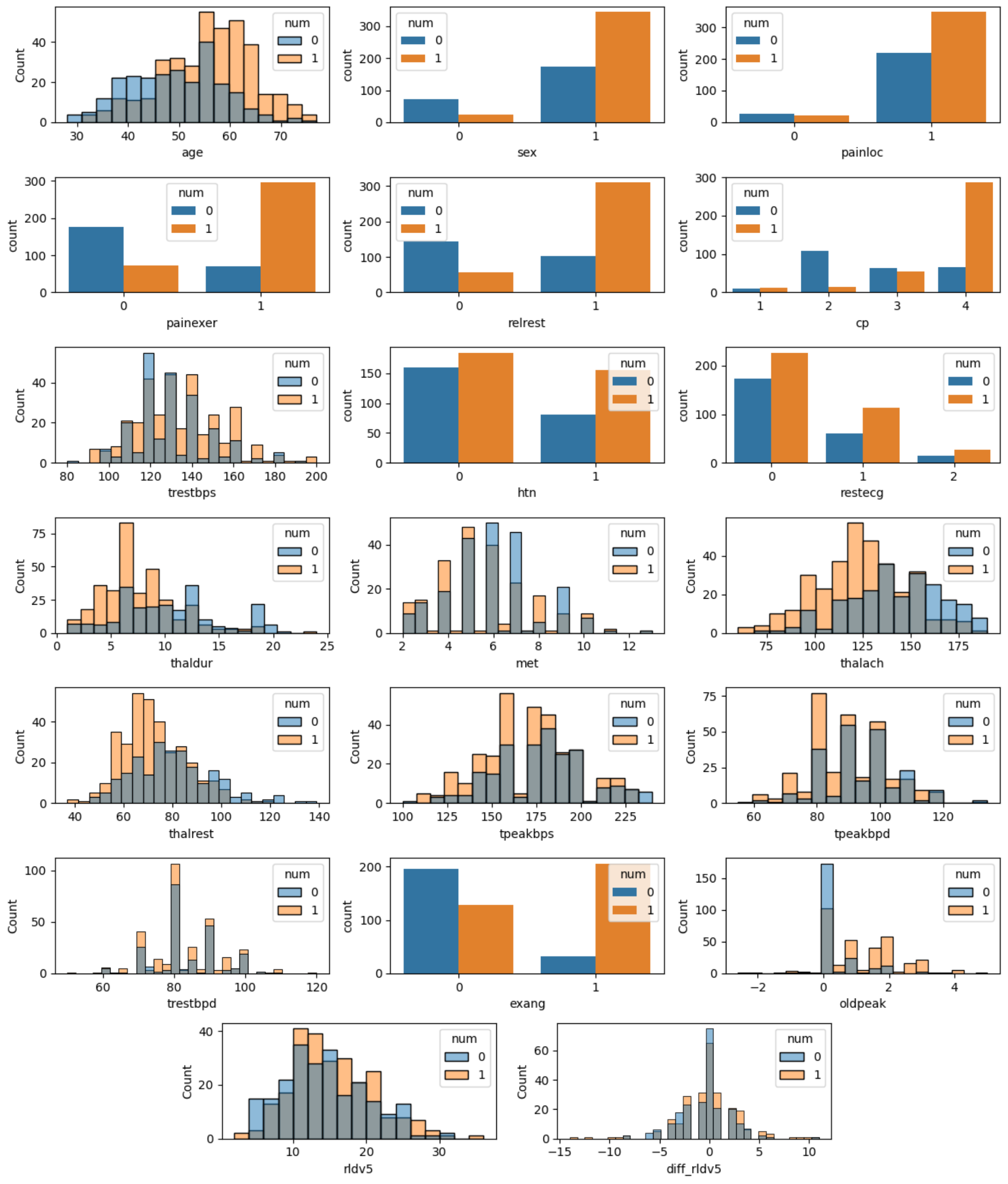


Figure 3: Variable distribution with respect to the target

2.6 Correlation among variables

Now we will look at the correlation between numerical variables[2]. We can observe that we do not have any correlations greater than 0.8, which would be the threshold we would use to decide whether to eliminate a variable due to multicollinearity. However, we do have several notable correlations,

[2]For more details, see A.1

such as between *met* and *thaldur*, *trestbpd* and *trestbps*, *thalach* and *thaldur*, and *thalrest* and *thalach*.

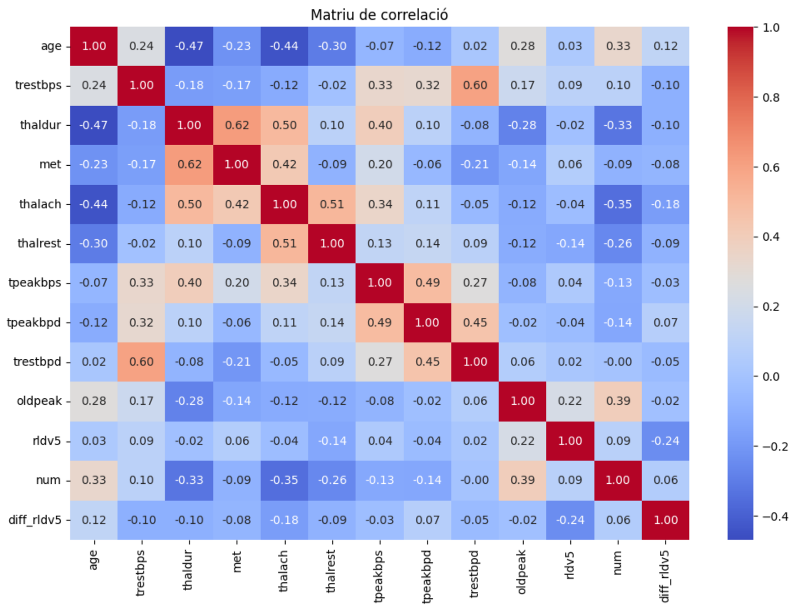


Figure 4: Correlation matrix of numerical variables

2.7 Train-test split

Before performing the final preprocessing steps, we allocated 80% of the data to training and the remaining 20% to testing to avoid data leakage. To ensure reproducibility, we set $random\_state = 1234$.

2.8 Scaling of numerical variables

The magnitudes of the numerical data are not the same, so there is a risk that the models will focus only on the variables with the highest magnitudes. Therefore, we scale the numerical variables using MinMax. To avoid data leakage, we fit the scaler on the training set and then use this fit to scale both the training and test sets.

2.9 Encoding of categorical variables

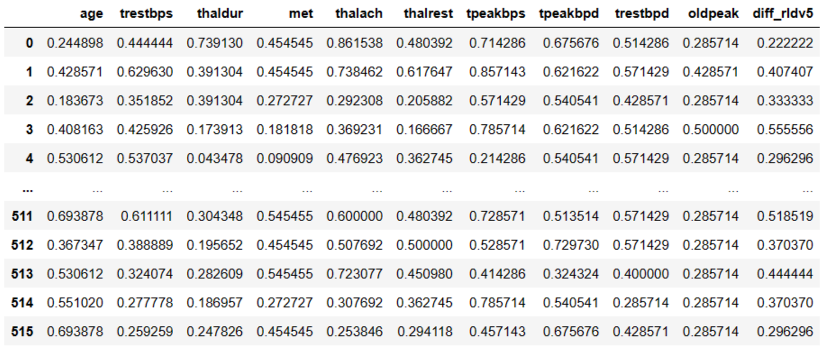
To enable the models to interpret the categorical variables, we will apply one-hot encoding using the *get_dummies* function from pandas.

2.10 Treatment of missing values

Since most supervised classification learning models do not accept NaN values and it is considered good practice to avoid them, we will impute the missing values. We will only impute numerical variables, while rows with NaNs in categorical variables will simply be removed. We have decided on this procedure because an error in numerical variables is generally less harmful as long as it falls within a range close to the value that should have been imputed.

In contrast, errors in categorical variables would have a greater impact on the results. Therefore, we remove rows that have any NaN value in any categorical column, reducing the dataset from 616 to 524 rows. Since we retain more than 80% of the original rows, we consider this reduction acceptable.

Furthermore, for imputing numerical data, we will use the K-Nearest Neighbor (KNN) imputer, as it provides more robust estimates of features compared to methods like mean imputation. The KNN imputer uses a local neighborhood to make predictions. Conceptually, if we consider each row as a point in the variable space and define a distance function and an imputation function, it selects the k nearest points and uses the imputation function on the column values corresponding to the NaNs to make predictions. Considering this, we choose the Euclidean distance metric as the distance function and arithmetic mean as the imputation function.



	age	trestbps	thaldur	met	thalach	thalrest	tpeakbps	tpeakbpd	trestbpd	oldpeak	diff_rldv5
0	0.244898	0.444444	0.739130	0.454545	0.861538	0.480392	0.714286	0.675676	0.514286	0.285714	0.222222
1	0.428571	0.629630	0.391304	0.454545	0.738462	0.617647	0.857143	0.621622	0.571429	0.428571	0.407407
2	0.183673	0.351852	0.391304	0.272727	0.292308	0.205882	0.571429	0.540541	0.428571	0.285714	0.333333
3	0.408163	0.425926	0.173913	0.181818	0.369231	0.166667	0.785714	0.621622	0.514286	0.500000	0.555556
4	0.530612	0.537037	0.043478	0.090909	0.476923	0.362745	0.214286	0.540541	0.571429	0.285714	0.296296
...	...	...	...	...	...	...	...	...	...	...	...
511	0.693878	0.611111	0.304348	0.545455	0.600000	0.480392	0.728571	0.513514	0.571429	0.285714	0.518519
512	0.367347	0.388889	0.195652	0.454545	0.507692	0.500000	0.528571	0.729730	0.571429	0.285714	0.370370
513	0.530612	0.324074	0.282609	0.545455	0.723077	0.450980	0.414286	0.324324	0.400000	0.285714	0.444444
514	0.551020	0.277778	0.186957	0.272727	0.307692	0.362745	0.785714	0.540541	0.285714	0.285714	0.370370
515	0.693878	0.259259	0.247826	0.454545	0.253846	0.294118	0.457143	0.675676	0.428571	0.285714	0.296296

Figure 5: Scaled numerical variables

Like before with the scaler, we have adjusted the KNN just with the train set in order to avoid data leakage from the test set. Regarding the hyperparameter k, for simplicity we kept the default value, $k = 5$. Besides, it is a large enough value to assume a certain degree of robustness.

3 Metrics

Since our dataset concerns medical data, misdiagnosing someone who is sick as not sick (false negative) is much more serious than the reverse. In the case of a false positive, further medical tests can later confirm the absence of the disease. Therefore, we want a metric that penalizes false negatives especially. This metric will be recall, which is defined as:

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

Where TP are the positives that have been predicted as so, and FN the false negatives.

However, if we base our model decisions solely on recall, we might predict all cases as positive. Therefore, we need a metric that considers precision to some extent but gives more weight to recall. This metric will be the F2-score:

$$F_2 = \frac{5TP}{5TP + 4FN + FP} = 5 \cdot \frac{precision \cdot recall}{4 \cdot precision + recall}$$

Finally, in addition to these metrics, we will also use accuracy, F1-score, precision, and recall to get a more detailed understanding of the models' performance.

4 Proposed models

To train and evaluate our models, we will use the train (80%) and test (20%) partitions that we created in the section 2.7. Notice that this data was scaled.

We have considered both linear and non-linear models.

- Linear: LDA, Logistic Regression, SVM
- Non-linear: QDA, Naive Bayes, KNN, Random Forest and Gradient Boosting.

In order to determine the best parameters for the proposed models, we used the Sklearn *GridSearchCV* function. By providing different values for each parameter and metrics, this function tries all parameter combinations, and with the *refit='f2'* argument, we obtain the parameters that yield the best F2-score.

Due to our limited data, we performed 5-fold cross-validation and averaged the results to obtain more robust estimations. Furthermore, to ensure reproducibility of our results, we set *'random_state=1234* for all models where applicable.

4.1 Logistic Regression

We used the L2 penalty as we are not interested in producing sparsity at the moment. Additionally, we set *max_iter = 10000* to ensure that the solver algorithm converges.

For the remaining hyperparameters of the model, we conducted an exhaustive search. We chose values for the parameter C, which represents the inverse of the regularization strength, across different orders of magnitude to test significantly different values. Furthermore, we tried two solvers: liblinear, which is designed for large datasets and uses coordinate descent, and lbfgs, which performs better with dense datasets and utilizes gradient descent.

Parameter	**Values tested**
Solver	'lbfgs', 'liblinear'
C	0.0001, 0.001, 0.01, 0.1, 1, 10, 100

Table 1: Paràmetres i valors per la Regressió Logística

The best configuration found is *solver = liblinear* and $C = 0.001$, which implies a strong regularization.

4.2 Discriminant Analysis

The discriminant analysis models assume normality in the data, so we had to transform the numerical variables to follow a Gaussian distribution. Since some variables had negative values, we used a Yeo-Johnson transformation. However, even after applying it, the data did not fully conform to a normal distribution, which will result in significantly lower performance compared to other models, as will be seen later.

Next, with the data transformation completed, we evaluated the Quadratic Discriminant Analysis (QDA) and Linear Discriminant Analysis (LDA) models. We also tried Naive Bayes for its simplicity. Since we have numerical variables and categorical variables converted to dummy variables, we used a GaussianNB and a BernoulliNB, respectively. In this way, assuming that the features are independent, we made predictions for each model separately and combined them by taking their product, which we then scaled to ensure it forms a probability distribution.

4.3 K-Nearest Neighbors

With KNN, we explored different numbers of neighbors, a parameter directly impacting performance. The fewer neighbors considered, the higher the risk of overfitting. We also modified the 'weights' parameter. Choosing 'uniform' gives equal importance to all neighbors, whereas 'distance' gives more weight to closer neighbors. Finally, to simplify the search space, we only considered Euclidean and Manhattan distances (p = 1 and p = 2).

Parameter	**Values tested**
n_neighbors	2, 3, 4, 5, 6, 7, 8, 9, 10
weights	'uniform', 'distance'
p	1, 2

Table 2: Parameters and values for K-Nearest Neighbors

The best configuration obtained is *n_neighbors* = 10, *p* = 2, *weights* = 'distance'. It is logical that *n_neighbors* = 10 and *weights* = 'distance' are in the optimal configuration, as we are weighting the values of the maximum number of evaluated neighbors, giving more weight to those closer.

4.4 Random Forest

Regarding the hyperparameters of Random Forest, we have considered the number of trees involved in the final decision, as well as the maximum number of variables to consider for each split. The parameter *sqrt* attempts to reduce the correlation between trees by choosing a small number of variables without losing too much information. The value *log2* is even more aggressive than *sqrt*, and *None* uses all available information by employing all variables.

As for model evaluation, although the Out-Of-the-Bag (OOB) error could replace cross-validation error and save computational cost, we have chosen to use cross-validation error to be consistent with the other models and have a slightly more robust estimate. Additionally, since we do not have a large number of data points, the additional computation time is not excessive.

Parameter	**Values tested**
n_estimators	1, 5, 25, 50, 75, 100, 200
max_features	'sqrt', 'log2' i None

Table 3: Parameters and values for Random Forest

The best parameters obtained are *max_features* = 'log2' i *n_estimators* = 50.

4.5 Gradient Boosting

For Gradient Boosting, we will modify the *n_estimators*, which represents the number of trees we will use. Due to the sequential nature of these models, where each tree corrects the errors of the previous one, this parameter is extremely important.

Additionally, we have tried different values of *max_depth*, the depth of the trees. We observed that when we set *max_depth* = None, we achieved a recall very close to or equal to 100%, while precision was below 60%.

We have also adjusted *learning_rate*, a parameter that reduces the influence of each additional tree, and *loss*, the loss function we seek to optimize. Setting *loss* = 'exponential' is equivalent to using AdaBoost.

Parameter	**Values tested**
n_estimators	1, 5, 25, 50, 75, 100, 150, 200
max_depth	3, 4, 5
learning_rate	0.05, 0.1, 0.15, 1, 2
loss	'log_loss', 'exponential'

Table 4: Parameters and values for Gradient Boosting

The optimal configuration of parameters among tested is: *learning_rate* = 0.1, *loss* = exponential, *max_depth* = 3 i *n_estimators* = 100. It makes sense, since a low *learning_rate* provides a smooth and stable learning and the obtained *n_estimators* ensures considerably robust results.

4.6 Support Vector Classifier

In this case, we have modified the parameter C, which is the regularization parameter inversely proportional to the regularization strength of the model. We have also tested different types of kernels (*linear*, *rbf*, and *polynomial*), which have a crucial effect in classifying non-linearly separable data. Finally, *gamma*, the kernel coefficient for 'rbf' and 'poly', and *degree*, which is only necessary for polynomial kernels.

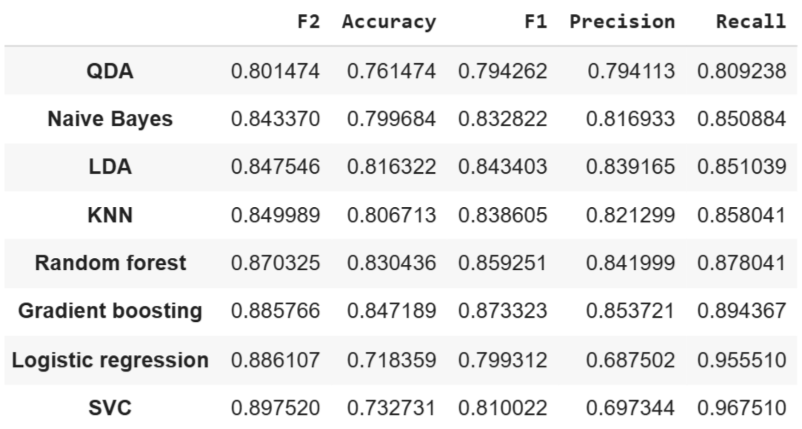
Parameter	**Values tested**
C	0.001, 0.005, 0.01, 0.05, 0.1, 1, 5, 10, 20
kernel	'lineal', 'rbf', 'poly'
gamma	'scale', 'auto', 0.1, 0.5, 1, 5
degree	2, 3, 4, 5

Table 5: Parameters and values for SVM

The best configuration found is: $C = 0.05$, $gamma = 0.5$, $kernel =$ rbf. This means we are applying moderate regularization and using a kernel that allows handling linearly non-separable data.

5 Results discussion and model election

To analyze the performance of the models, we cannot rely on the test partition, as these are data we must reserve to estimate the generalization error once a model is chosen. We used the results with the best parameters obtained during cross-validation with *gridSearch*. These are the metrics obtained:



	F2	Accuracy	F1	Precision	Recall
QDA	0.801474	0.761474	0.794262	0.794113	0.809238
Naive Bayes	0.843370	0.799684	0.832822	0.816933	0.850884
LDA	0.847546	0.816322	0.843403	0.839165	0.851039
KNN	0.849989	0.806713	0.838605	0.821299	0.858041
Random forest	0.870325	0.830436	0.859251	0.841999	0.878041
Gradient boosting	0.885766	0.847189	0.873323	0.853721	0.894367
Logistic regression	0.886107	0.718359	0.799312	0.687502	0.955510
SVC	0.897520	0.732731	0.810022	0.697344	0.967510

Figure 6: Cross-validation metrics

We observe the worst performance with the discriminant analysis models (QDA, Naive Bayes, and LDA). This is consistent with the fact that the variables, despite being transformed, did not fit a normal distribution. Additionally, having categorical variables, which are not inherently compatible with these models, forced them to interpret the variables unnaturally through one-hot encoding.

On the other hand, the performance of KNN is not very good either. This could be because, as a distance-based model, it gives the same importance to all numerical variables. Moreover, with the scaled variables and the categorical variables encoded with one-hot encoding, these might have the same or even more weight than the numerical variables.

Regarding the performance of the two ensemble models evaluated, Random Forest and Gradient Boosting, we see that it is very similar, with Gradient Boosting being slightly superior. We observe that, besides having a quite good F2-score, they have a fairly balanced recall and precision. This makes sense as they use a more robust method for classification, exploiting the information that each variable provides in relation to the target to improve predictions. Additionally, being a combination of different models makes them more robust and balanced in terms of metrics.

Finally, we see that the best F2-scores were obtained by the two linear models different from LDA: Logistic Regression and SVC. Both models focused on maximizing recall at the expense of precision. That is, they focused on adjusting a hyperplane that contains most positive cases, sacrificing negative cases that are erroneously included within the positive prediction semi-plane (predicted as diseased). Analyzing the impact of the parameters, we see that this happens with small values of C, whereas with larger values of C we obtain similar recall and precision, although the F2-score is not as good. Between SVC and Logistic Regression, we lean towards SVC, as all metrics are better.

The final choice, therefore, would be between Gradient Boosting and SVC. With Gradient Boosting, despite having a lower F2-score, we see that the rest of the metrics are much more balanced, while SVC clearly focuses on maximizing recall. In line with the idea of not misdiagnosing the diseased, we might choose SVC.

However, it is important to remember the use case of the models, and that medical professionals will use these models. The fact that Gradient Boosting is a highly explainable model is a valuable feature for doctors. Moreover, if we consider the situation in Spain, given the saturation of waiting lists and the often lacking resources in public healthcare, an excessively high number of false positives could worsen this situation.

Therefore, we will finally choose the **Gradient Boosting** model, which, despite having a recall approximately 5% lower than SVC, is more explainable and has a 15% higher precision. This higher precision would greatly reduce the burden on doctors, who would not have to waste time on false positives, and potentially allow for the adequate care of genuinely ill patients.

6 Variable importance in the model

When training a model, some variables contribute more significantly to performance. The Gradient Boosting model allows us to estimate the importance of these variables. For each decision tree used, and for each node, the decrease in the Gini metric weighted by the number of observations affected by the node is calculated. The overall importance of each variable is determined by averaging these loss reductions across all trees.

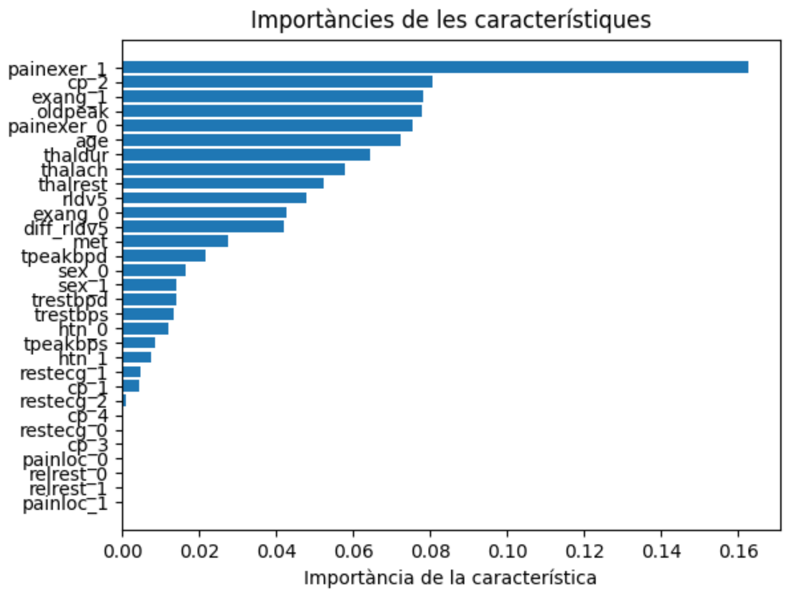


Figure 7: Variable importance in Gradient Boosting

To understand these results, it is necessary to keep in mind section **??**. We can observe that the model does not give importance to variables such as *relrest* or some values of *painloc*. Regarding these values of *painloc*, it is probably because the vast majority of cases are concentrated in class 1 of the variable, and therefore, the rest of the values are proportionally negligible. As for *relrest*, it is surprising that it is not very relevant to the model, considering that depending on whether it is 0 or 1, the proportions of patients change significantly. A plausible explanation is that the effect of this variable is diluted by other variables with a greater impact on predicting the target and that convey information quite similar to what *relrest* conveys about the patient's condition.

On the other hand, it is natural that the model considers variables related to chest pain such as *painexer_1*, *exang_1*, or *cp_2* important for predicting heart disease. Additionally, we see that *age* is quite relevant, which is consistent with age being a risk factor for many diseases.

7 Dimensionality reduction

If we select a subset of the variables used to fit the model, given that not all have the same importance, and perform training with them, we will achieve models with less tendency to overfit, with fewer parameters, and therefore require less computational cost. Additionally, they will be more easily interpretable for doctors and potentially we could improve the metrics by eliminating the noise from less important parameters.

Although there are functions like *RFECV*, which eliminates variables based on the importance given to them by a given model, for research purposes, we manually removed the least important features and, at each iteration, re-fitted the Gradient Boosting with *gridSearch*. The reason is that the best parameters for all the variables are not necessarily the best for a subset of variables. It is worth noting that this methodology is computationally very intensive.

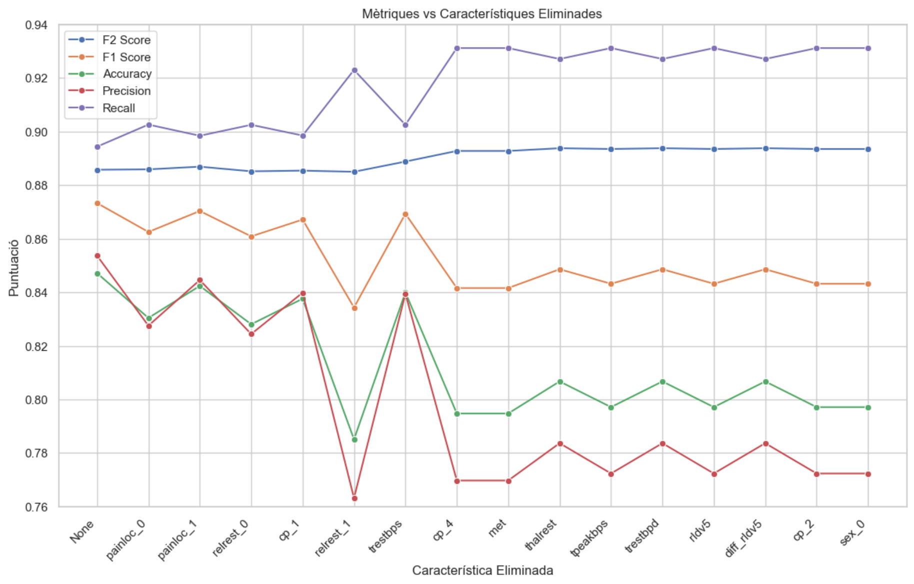


Figure 8: Metric evolution depending on variable reduction[3]

In terms of precision and accuracy, it is evident that we should not eliminate variables beyond *trestbps*. Additionally, if we consider the F2-score, we would choose to remove all variables up to this point. Recall also improves by eliminating up to *trestbps* compared to the model with all variables. Many of these variables have already been discussed in the previous section. The only one not previously mentioned as less important is *trestbps*, which has a similar distribution for both sick and healthy individuals (section 2.5).

Therefore, we consider it a good idea to eliminate the variables *painloc_0, painloc_1, relrest_0, relrest_1, cp_1* and *trestbps*.

8 Estimation of the generalization error

To see how our model would perform with new data, we will evaluate it with the test set. The Gradient Boosting model with feature reduction manages to maintain a performance on the test set very similar to that on the training set, indicating that the model has not overfitted.

Mètrica	**Valor al train**	**Valor al test**
F2 Score	0.8888	0.8923
Accuracy	0.8399	0.8381
F1 Score	0.8693	0.8618
Precision	0.8395	0.8154
Recall	0.9025	0.9138

Table 6: Metrics of Gradient Boosting with dimensionality reduction

[3]This graph should be interpreted sequentially from left to right. Each point on the horizontal axis implies having removed that variable and all variables to its left.

To get a clearer idea of the model's performance, we will present the confusion matrix. The performance is as expected: we have more false positives than false negatives due to recall being higher than precision. A desirable aspect to highlight is that the number of false positives is less than the number of true positives.

In section 2.5, we highlighted the dataset's lack of parity and the difference in the proportion of patients by gender. To ensure that our model is not biased, we will plot the confusion matrix by gender. For men, the confusion matrix is very similar to the overall one. The most notable difference is seen in the women's confusion matrix. We observe that most women are correctly predicted as healthy. However, the number of false positives, false negatives, and true negatives is the same. Even so, we cannot infer any bias from the model because the amount of data is not significant.

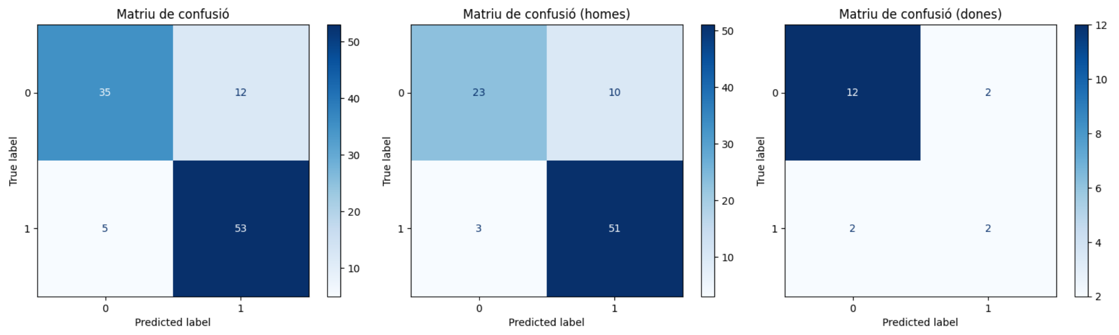


Figure 9: Confusion matrices, from left to right: general, men and women

Next, we will analyze the ROC curve. The ROC curve represents the true positive rate against the false positive rate. Therefore, the better the model's performance, the closer the ROC curve will be to the top left corner of the graph, indicating a higher true positive rate and a lower false positive rate.

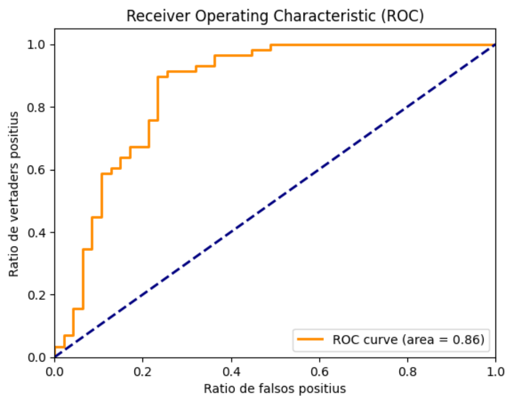


Figure 10: ROC curve

We can observe that the top left corner of the curve approaches the edge. Finally, regarding the area under the curve (AUC), a value of 1 indicates that a true positive rate of 1 and a false positive rate of 0 have been achieved. Similarly, a value of 0.5 would be obtained by a random classifier. Therefore, the higher this value, the better the model distinguishes between positive and negative cases.

In our case, we obtain an AUC of 0.86, which is considered quite good, indicating that our model distinguishes significantly between the positive and negative class.

9 Conclusions

The main issue with this dataset is the limited quantity and quality of the data. On one hand, we had to discard many variables and instances due to the presence of numerous NaNs. On the other hand, the data was not well balanced, especially concerning gender, where women were underrepresented. Therefore, we cannot draw strong conclusions, for example, regarding gender bias in the model.

Nevertheless, with proper preprocessing, we achieved good results. In any problem, defining metrics that are most suitable is crucial. In this clinical project, it was crucial to minimize the number of undetected sick individuals (false negatives), making the F2-score the most appropriate metric.

After an exhaustive search for parameters across different models, we opted for Gradient Boosting. Subsequently, we eliminated the less important variables and obtained a model with an F2-score of 0.8923 and accuracy of 0.8381 on the test set. Therefore, the final Gradient Boosting model is able to strike a balance between few false negatives and few false positives, thus adapting well to the clinical context and the tasks of healthcare professionals. Moreover, it has the advantage of being highly interpretable.

As an extension of this work, one could consider evaluating whether the results would differ significantly if the models were trained on data from two hospitals and the remaining hospital reserved for testing. Additionally, exploring the predictive capacity enhancement through ensemble methods using the best fitted models could also be worthwhile.

Appendices

A Variables of the dataset (original)

0. **id:** patient identification number
1. **ccf:** social security number (replaced with a dummy value of 0)
2. **age:** age in years
3. **sex:** sex (1 = male; 0 = female)
4. **painloc:** chest pain location (1 = substernal; 0 = otherwise)
5. **painexer:** provoked by exertion (1 = yes; 0 = no)
6. **relrest:** relieved after rest (1 = yes; 0 = no)
7. **pncaden:** sum of painloc, painexer, and relrest
8. **cp:** chest pain type
   - Value 1: typical angina
   - Value 2: atypical angina
   - Value 3: non-anginal pain
   - Value 4: asymptomatic
9. **trestbps:** resting blood pressure (in mm Hg on admission to the hospital)
10. **htn**: **Suposem que vol dir 0 no hipertens, 1 hipertens**
11. **chol:** serum cholesterol in mg/dl
12. **smoke:** smoking status (1 = yes; 0 = no)
13. **cigs:** cigarettes per day
14. **years:** number of years as a smoker
15. **fbs:** fasting blood sugar > 120 mg/dl (1 = true; 0 = false)
16. **dm:** history of diabetes (1 = yes; 0 = no)
17. **famhist:** family history of coronary artery disease (1 = yes; 0 = no)
18. **restecg:** resting electrocardiographic results
    - Value 0: normal
    - Value 1: having ST-T wave abnormality
    - Value 2: probable or definite left ventricular hypertrophy by Estes' criteria
19. **ekgmo:** month of exercise ECG reading

20. **ekgday:** day of exercise ECG reading
21. **ekgyr:** year of exercise ECG reading
22. **dig:** digitalis used during exercise ECG (1 = yes; 0 = no)
23. **prop:** beta blocker used during exercise ECG (1 = yes; 0 = no)
24. **nitr:** nitrates used during exercise ECG (1 = yes; 0 = no)
25. **pro:** calcium channel blocker used during exercise ECG (1 = yes; 0 = no)
26. **diuretic:** diuretic used during exercise ECG (1 = yes; 0 = no)
27. **proto:** exercise protocol
    - Value 1: Bruce
    - Value 2: Kottus
    - Value 3: McHenry
    - Value 4: fast Balke
    - Value 5: Balke
    - Value 6: Noughton
    - Value 7: bike 150 kpa min/min
    - Value 8: bike 125 kpa min/min
    - Value 9: bike 100 kpa min/min
    - Value 10: bike 75 kpa min/min
    - Value 11: bike 50 kpa min/min
    - Value 12: arm ergometer
28. **thaldur:** duration of exercise test in minutes
29. **thaltime:** time when ST measure depression was noted
30. **met:** mets achieved
31. **thalach:** maximum heart rate achieved
32. **thalrest:** resting heart rate
33. **tpeakbps:** peak exercise blood pressure (first of 2 parts)
34. **tpeakbpd:** peak exercise blood pressure (second of 2 parts)
35. **dummy**
36. **trestbpd:** resting diastolic blood pressure
37. **exang:** exercise induced angina (1 = yes; 0 = no)
38. **xhypo:** (1 = yes; 0 = no)

39. **oldpeak:** ST depression induced by exercise relative to rest
40. **slope:** the slope of the peak exercise ST segment
    - Value 1: upsloping
    - Value 2: flat
    - Value 3: downsloping
41. **rldv5:** height at rest
42. **rldv5e:** height at peak exercise
43. **ca:** number of major vessels (0-3) colored by fluoroscopy
44. **restckm:** irrelevant
45. **exerckm:** irrelevant
46. **restef:** rest radionuclide ejection fraction
47. **restwm:** rest wall motion abnormality
    - 0 = none
    - 1 = mild or moderate
    - 2 = moderate or severe
    - 3 = akinesis or dyskinesis
48. **exeref:** exercise radionuclide ejection fraction
49. **exerwm:** exercise wall motion
50. **thal:**
    - 3 = normal
    - 6 = fixed defect
    - 7 = reversible defect
51. **thalpul:** not used
52. **thalsev:** not used
53. **earlobe:** not used
54. **cmo:** Month of cardiac catheterization (presumed)
55. **cday:** Day of cardiac catheterization (presumed)
56. **cyr:** Year of cardiac catheterization (presumed)
57. **num:** Diagnosis of heart disease (angiographic disease status)
    - Value 0: Less than 50% diameter narrowing
    - Value 1: More than 50% diameter narrowing

58. **lmt:** Left main trunk
59. **ladprox:** Proximal left anterior descending artery
60. **laddist:** Distal left anterior descending artery
61. **diag:** Diagonal branch
62. **cxmain:** Main circumflex artery
63. **ramus:** Ramus intermedius
64. **om1:** First obtuse marginal
65. **om2:** Second obtuse marginal
66. **rcaprox:** Proximal right coronary artery
67. **rcadist:** Distal right coronary artery
68. **lvx1:** not used
69. **lvx2:** not used
70. **lvx3:** not used
71. **lvx4:** not used
72. **lvf:** not used
73. **cathef:** not used
74. **junk:** not used
75. **name:** last name of patient (replaced with a dummy string ”name”)

A.1 Scatterplot de les variables

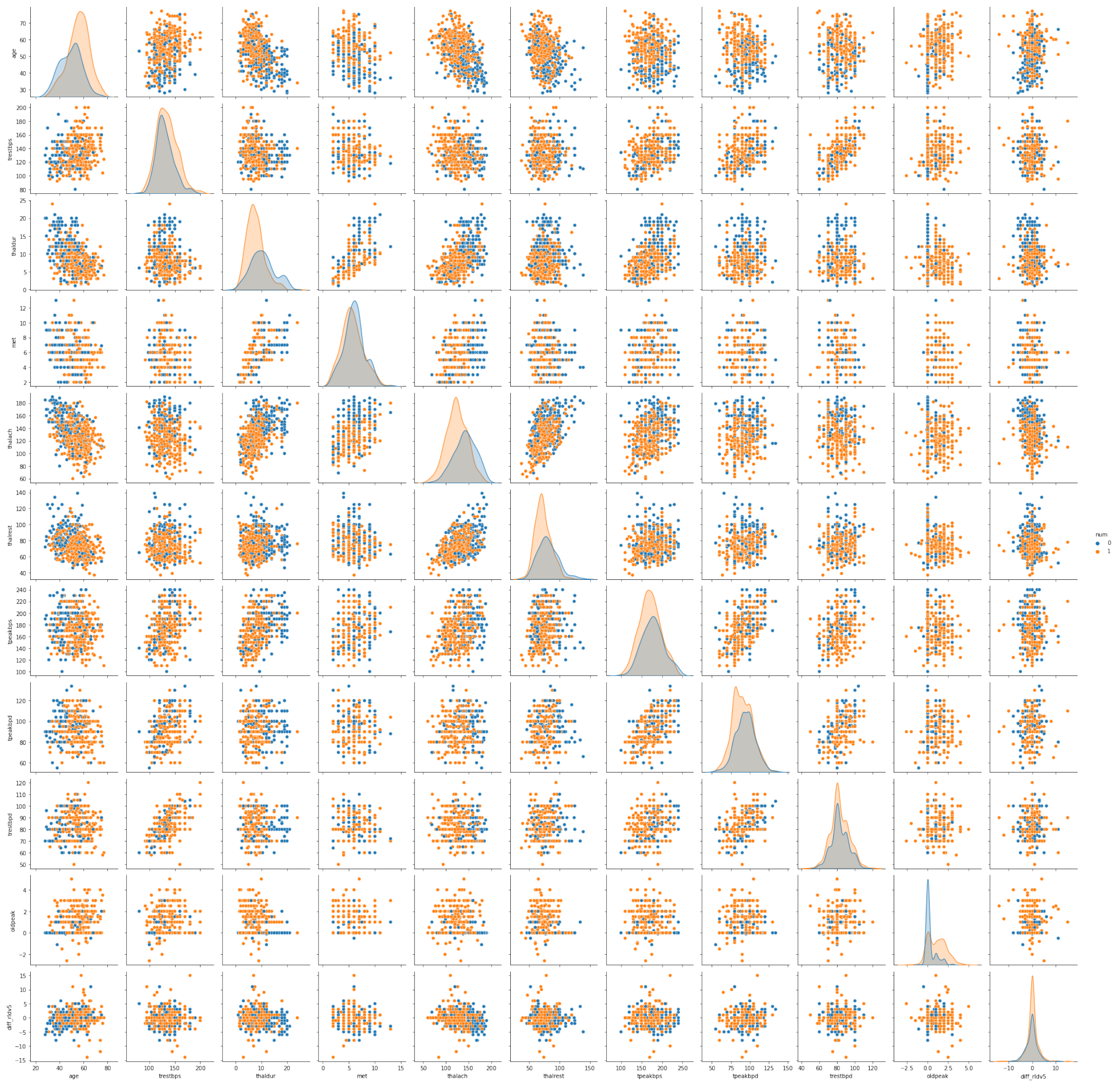


Figure 11: Scatterplot de les variables numèriques

Explicació de correlacions:

- met-thaldur: correlació entre met, que és la quantitat d'oxigen consumida per fer l'exercici i thaldur, la duració de l'exercici té sentit que estiguin positivament correlades.
- trestbpd (pressió sanguínia en repòs) i trestbps (pressió sanguínia en repòs mesurada en entrar al centre mèdic) haurien d'estar força positivament correlades, ja que en principi la diferència

que hi hauria d'haver hauria de ser poca, probablement fruit de l'alteració que ha dut el pacient a acudir a l'hospital.

- Quant a correlacions una mica menys significatives tindríem, entre d'altres, thalach i age, que és lògic que estiguin negativament correlades, ja que normalment els problemes que podrien ocasionar un major heart rate en repòs acostumen a accentuar-se amb l'edat. Exemples d'això són l'enduriment de les artèries, menor activitat física o el mateix envelliment del cor, que al perdre capacitat de bombeig de sang necessita compensar-ho augmentant els batecs per minut
- age-trestbps: En aquest gràfic podem veure que les persones, tan les malaltes com no, estàn equidistribuides en la pressió en estat de repòs de quan van entrar al centre mèdic. Això ens mostra la poca correlació entre les variables.
- age-thalach: Aquest scatter-plot mostra un correlació negativa entre les variables, i podem observar que la concentració de gent amb cardiopaties té lloc a una edat més elevada i un hearth rate màxim inferior.
- thaldur-thalach: Aquí veiem la correlació positiva que havíem comentat abans entre les dues variables. A més a més, veiem que la gent amb cardiopaties es concentra en els nivells més baixos de la variable thaldur. Thalach i thaldur també té sentit que estiguin correlades, ja que representen el màxim heart rate assolit i la duració de l'exercici.
- thalach-thalrest: thalrest i thalach també estan positivament correlats, fet que indica que a priori un major heart rate en repòs implica un major heart rate en l'exercici.