Nicole Maria Formenti DSE, 941481

STATISTICAL LEARNING PROJECT
supervised learning

ABSTRACT

This report analyses a small dimensional dataset on heart diseases with binary response and mixed data type variables by applying different supervised learning methods. The goal is to study their predictive performance by comparing them and to provide a theoretical background. On average, all the models were able to properly fit the data, with a proportion of correctly classified instances between 80% and 90%. The methods that actually provided the best results were logistic regression and gradient boosting.

GOALS AND METHODS OF THE REPORT

The main aim of this report is to analyze a dataset containing mixed type variables (continuous, binary and categorical) with a binary response by means of several different supervised learning methods, in order to find the most suited ones to make predictions on new data. Furthermore, another point of interest is to find the variables with the highest effect on the response. The analysis will focus more on predictive capability rather than model interpretability. The models considered for this analysis are:

- Logistic regression
- Lasso
- Ridge regression
- Tree classifier
- Bagging
- Random forest
- Boosting

The study will be enriched by providing a theoretical background to the methods considered and by widely commenting the results. Finally, the performances of each approach will be evaluated, along with their main advantages and drawbacks, and then compared.

DATASET DESCRIPTION

Regarding the dataset used, it has been built by using 4 different datasets regarding the presence of heart diseases in patients. The source is ‘UCI machine learning repository’ and the dataset name is ‘Heart Disease Data set’. The original data collections contained 76 attributes of which only 14 have been used. The 4 datasets were collected from:

- Cleveland Clinic foundation, Cleveland
- Hungarian Institute of Cardiology, Budapest
- V.A. Medical Center, Long Beach (California)
- University Hospital, Zurich (Switzerland)

The 14 features included are the following:

- Age: age in years
- Sex:
    - 1 : male
    - 0 : female
- Cp: chest pain type
    - 1: typical angina
    - 2: atypical angina
    - 3: non-anginal pain
    - 4: asymptomatic
- Trestbps: resting blood pressure in mm Hg (on admission to the hospital)
- Chol: serum cholesterol in mg/dl
- Fbs: fasting blood sugar > 120 mg/dl
    - 1 : true
    - 0 : false
- Restecg: resting electrocardiographic results
    - 0: normal
    - 1: ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)
    - 2: probable or definite left ventricular hypertrophy by Estes' criteria
- Thalach: maximum heart rate achieved
- Exang: exercise induced angina
    - 1 : yes
    - 0 : no
- Oldpeak: ST depression induced by exercise relative to rest
- Slope: the slope of the peak exercise ST segment
    - 1: upsloping
    - 2: flat
    - 3: downsloping
- Ca: number of major vessels (0-3) colored by fluoroscopy
- Thal: thalassemia:
    - 3 : normal
    - 6 : fixed defect
    - 7 : reversable defect
- Num: diagnosis of heart disease (angiographic disease status)
    - 0: < 50% diameter narrowing
    - 1, 2, 3, 4: > 50% diameter narrowing

The target variable is ‘num’, which indicates whether a patient presents heart diseases or not

DATA CLEANSING

Before starting the actual analysis, it is necessary to inspect and prepare the dataset, according to one’s needs.
First, the target variable *‘num’* needs to be fixed, since the positive response in some datasets has been distinguished in 4 levels according to the percentage of diameter narrowing. This division is unnecessary because the outcome of interest is whether a patient has heart diseases or not, moreover the other datasets targets are already expressed in binary terms. Thus, the response is dichotomized, so that it is expressed as 0 or 1.

The next step is to remove missing values, which are useless for the analysis and might cause errors in computations. By looking at the dataset, it can be seen that some features present a large number of null values, as a consequence, in order to avoid removing too many instances, it is better to directly remove the entire column. The variables which present the highest number of missing values are ‘ca’ (66%) and ‘*thal’* (34%), therefore they are dropped. Next, every observation containing at least a null value is deleted.

Another important aspect to be aware of is the presence of outliers, which are extreme values, quite far away from the main bulk of data points. In order to detect them, data summary and boxplots are inspected.
From the statistics returned for each feature, two variables seem to have some problems. First, *‘trestbps’* has a mean of 16.28 with a minimum of 2.00 and a maximum value of 52, which is far from the mean but not that much from the third quartile (26.00). Next, the variable *‘chol’* is even more likely to present outliers, with a mean of 207 and a minimum (0.0) pretty far from the first quartile (184.5). Moreover, the maximum (603) is far from the third quartile (272.5).

```
      age        sex     cp         trestbps          chol       fbs
 Min.   :31.00   0: 94   1: 19   Min.   : 2.00   Min.   :  0.0   0:340
 1st Qu.:49.00   1:313   2: 55   1st Qu.: 7.50   1st Qu.:184.5   1: 67
 Median :56.00           3: 82   Median :10.00   Median :228.0
 Mean   :55.09           4:251   Mean   :16.28   Mean   :207.4
 3rd Qu.:61.00                   3rd Qu.:26.00   3rd Qu.:272.5
 Max.   :77.00                   Max.   :52.00   Max.   :603.0
 restecg    thalach       exang      oldpeak     slope   num
 0:228   Min.   :  2.00   0:184   Min.   : 6.0   1:124   0:142
 1: 70   1st Qu.: 20.00   1:223   1st Qu.:19.0   2:247   1:265
 2:109   Median : 39.00           Median :24.0   3: 36
         Mean   : 46.77           Mean   :25.1
         3rd Qu.: 77.00           3rd Qu.:28.0
         Max.   :107.00           Max.   :41.0
```

The boxplot for both features is plotted.

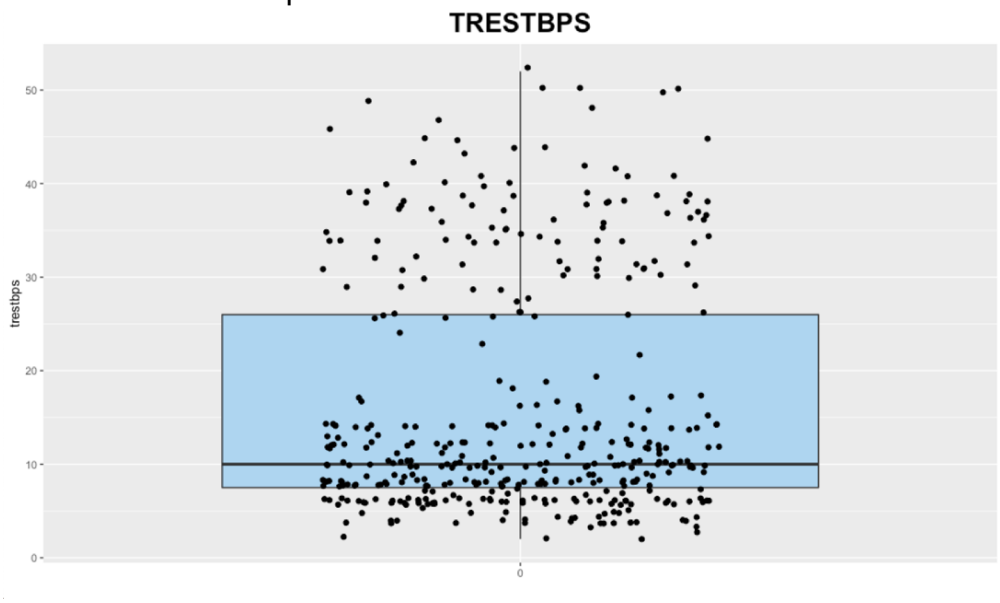

Figure 1: boxplot for trestbps

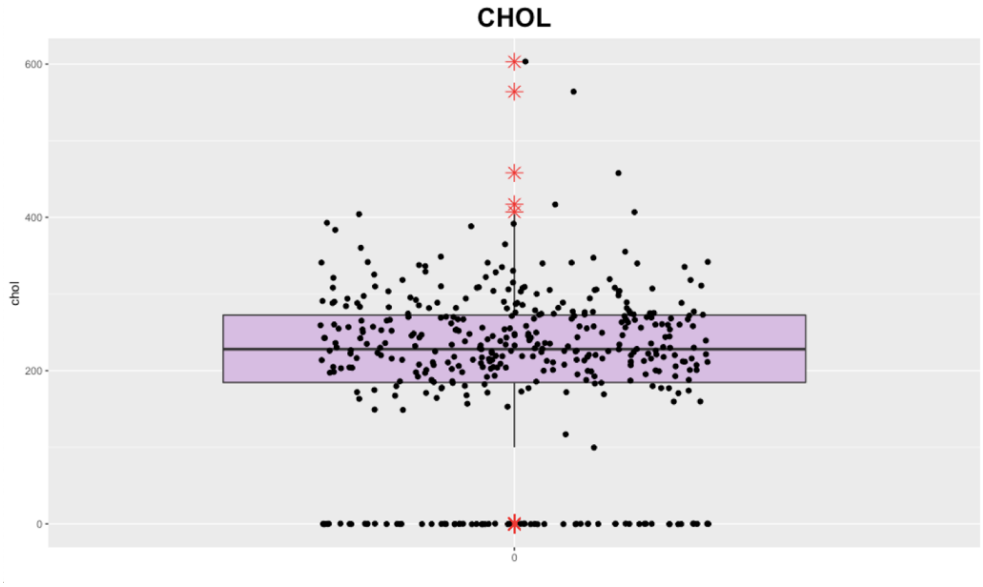

Figure 2: boxplot for chol

Concerning *'trestbps'* no outliers are found. On the other hand, *'chol'* presents outliers both in the lower and upper part of data distribution. An instance has value 0, which is impossible for serum cholesterol, meaning that there probably was a mistake. High values are possibly correct, but it is still better to remove some of them in order to avoid having problems when fitting model. Furthermore, there are few outliers, so removing them shouldn't have a significant negative effect even if the dataset is small. So, values below 100 and above 500 are removed.

The final dataset has 335 observations with 12 features. The dimension has been shrunk by about two thirds due to the fact that the original record quality wasn't quite high.
Luckily, the target variable is not excessively unbalanced, having 135 negative responses and 200 positive ones.

Because of the high number of variables it is not practical to visualize the relationship between them using a scatterplot, given it is not easily readable.

DATA ANALYSIS

The main body of the dataset analysis will be now presented and the models explained. The methods used are unsupervised learning models. They are applied to situations in which the true data response is known, so that once the model is fitted, the predicted response can be compared to it. In this way it is possible to assess how good is the model in making predictions.

LOGISTIC REGRESSION

Due to its simplicity the first model considered is linear regression, which captures the linear relationship between the variables and response through the use of some unknown parameters. These coefficients are estimated by the least squares approach, which minimizes the sum of squared residuals. This model works especially well with a quantitative target variable.
When the response variable is binary or more generally categorical, it's not a good idea to use linear regression, even if theoretically it could be used with a binary target. One can think of coding the y variable as a dummy variable and then assign

$$\begin{cases} 1 \; if \; y \; \geq 0.5 \\ 0 \; otherwise \end{cases}$$

The problem is that the response can take values outside the $[0,1]$ interval, so that it cannot be interpreted as a probability. In the case of multiple categories, creating dummies is not recommended, since we are implicitly assuming there is an ordering between the variables and moreover that the difference between them is always the same.
A natural way to model a binary response is to use logistic regression, which on the one hand is easily interpretable (much like a linear regression) and on the other hand it gives us not only the response class, but also the probability associated to it. Given that our aim is to predict whether a patient may suffer from heart diseases, we may be interested in the probability of the event, rather than a net classification.
In order to model $p(x)$, the logistic regression uses the logistic function, which returns a value between 0 and 1. The formula is

$$p(x) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + \; e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

Then, the log-odds or logits can be derived to obtain the model

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

In order to estimate the coefficients, the maximum likelihood method is used, which finds values of β such that they maximize the likelihood (probability) that the observed data were actually produced by the fitted model. It maximizes the probability that our model could have generated the observed data.
Another advantage of logistic regression is that when fitting the model, the interpretation of the found coefficients is analogous to the linear regression, since what it is shown in the output is the log-odds. For continuous variable, an increase of 1 in a predictor means an increase of β in the log-odds

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.965167   1.878699  -2.643 0.008221 **
age          0.031452   0.021804   1.442 0.149169
sex1         1.470295   0.432944   3.396 0.000684 ***
cp2         -0.131011   0.777775  -0.168 0.866234
cp3         -0.893503   0.766187  -1.166 0.243547
cp4          0.908182   0.729715   1.245 0.213290
trestbps     0.020301   0.014852   1.367 0.171674
chol         0.001077   0.003572   0.302 0.762986
fbs1         0.157597   0.533110   0.296 0.767522
restecg1    -0.056674   0.651403  -0.087 0.930669
restecg2     0.387720   0.430761   0.900 0.368078
thalach     -0.017644   0.006546  -2.695 0.007034 **
exang1       0.763811   0.449276   1.700 0.089113 .
oldpeak      0.047288   0.033007   1.433 0.151950
slope2       1.497639   0.458145   3.269 0.001080 **
slope3       0.564580   0.880272   0.641 0.521283
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 316.81  on 233  degrees of freedom
Residual deviance: 185.51  on 218  degrees of freedom
AIC: 217.51
```

From the output it can be seen that only few variables are significant

- intercept -4.965167
- sex1: 1.470295
  It is the distance of sex1 from sex0
- thalach: -0.017644
- exang1: 0.763811
- slope2: 1.497639
  It is the distance of slope2 from slope1

If the aim of the analysis would have been to interpret the results, the number of regression predictors should be reduced to get rid of non-significant features. They can not be simply eliminated by looking at their significance, since some variables may look significant or non-significant due to collinearity, even though actually it is the opposite. In this case the analysis is more concerned with obtaining a good prediction, so the non-significant features can be left as they are. However, for the sake of completeness, techniques that perform feature selection and reduction will be presented later. They may even lead to a performance improvement.

The performance of the logistic regression will be evaluated as follows.
When assessing the goodness of a model fitted on a certain dataset, it is more informative to look at the test error, which is the error in predicting new data, rather than the training error, which is, instead, the mistake made on the dataset used to train the model. What really matters is to have a model able to generalize, so that it can be used to make predictions and explain more general settings.
The test error plays a pivotal role in choosing the correct flexibility of the model, namely the number or value of parameters for parametric models. A good model finds the optimal tradeoff between bias and variance error, so that it is able to fit data properly without being affected too much by noise. If a model is overly flexible, it will fit the training data well but the new data will be fitted poorly, leading to overfitting. On the other hand, if it is not flexible enough it might perform poorly on both datasets, leading in this case to underfitting.
Another important aspect, other than model flexibility, is to always choose more parsimonious models when adding more parameters doesn't increase model performance, so that the model is easier to interpret.
The training error cannot directly be used as an approximation for test error, since the two often differ. The training error is always decreasing as the flexibility of the model increases, while the test error has a typical u-shape, first decreasing due to the effect of variance reduction, then increasing due to the increase in bias, which after a certain point dominates the decrease in variance. Unfortunately, most of the time a test set is not directly available.
In order to overcome this problem, one approach is to use a *validation set*. The original dataset is split in 2 parts, one used to train the model (training set), the other one used to assess it (validation set). The problem with the validation set approach is that the result may vary depending on the proportion of training and validation set chosen and on the specific observations included in them.
That's why in practice the most used approach is the cross-validation, which enhances the validation set method. It works by dividing the dataset into k folds, using $k-1$ folds at time to train the model and then test it on the k -th fold. The process is repeated k times, leaving out a different fold every time, then the resulting errors are averaged. A special case is the leave-one-out cross validation, in which a single observation is used to validate the model and the process is repeated as many times as the number of total observations.

To begin with, the validation approach is applied.
Since logistic regression provides probabilities, the percentage threshold used to classify the data points can be changed according to the type of problem. For instance, it's more important to detect people at risk of heart diseases than the opposite, so the threshold should be lower than 50% for positive responses. However, instead of lowering the threshold, it has been opted in favor of using appropriate misclassification metrics.
The misclassification rate is used to assess the performance of a model with binary response, providing the number of incorrectly classified instances.

The first step consists in building the confusion matrix, which shows the predicted and true value of an instance label. By using this matrix, different metrics can be applied.

The most immediate one is Accuracy

$$\frac{n.\,true\ positive\ obs + n.\,true\ negative\ obs}{n.\,of\ total\ predictions}$$

From Accuracy the Misclassification rate can be calculated as $1 - Accuracy$
However, this metric doesn't take into account the type of error made, which can be distinguished either as type I error (false positive) or as type II error (false negative). Other 2 measures, namely Precision and Recall, can be use instead.
Precision is defined as

$$\frac{n.\,true\ positive\ observations}{n.\,true\ positive\ obs + n.\,false\ positive\ obs}$$

Precision is recommended when the cost of false positive is high.

On the other hand, there is Recall

$$\frac{n.\,true\ positive\ observations}{n.\,true\ positive\ obs + n.\,false\ negative\ obs}$$

This metric is especially used when there is a high cost associated with false negative. Thus, it is fitted for the current problem of detecting sick patients, since misclassifying a patient with heart diseases as healthy is costlier than the opposite.
Furthermore, there is a fourth measure called F1 score, which seeks to balance Precision and Recall by averaging them

$$2 \times \frac{Precision \times Recall}{Precision + Recall}$$

It considers the 2 errors as being equally costly such as Accuracy, but it is able to deal with the problem of an unbalanced class distribution, with a large number of true negatives. Accuracy, on the other hand, is influenced by the true negatives which appear at the denominator.

Based on the type of analysis, which is detection of sick patients, Recall seems the appropriate metric to be used. Moreover, F1 score is not needed since the database is properly balanced.
The recall rate found with the validation approach is 95.16%. Then, 10-fold cross validation is used to assess the average Recall, which is 89.51%.
The two are not so different, meaning that the result found with the validation set approach is stable, even if upward biased. The performance of logistic regression is pretty good.

The recall rate is still high (89.51%). Logistic regression predicts quite well data, but since there are many variables, there probably is collinearity. It would be better then to reduce their

number, before refitting the model and exploring it. That's because some variables may look significant or non-significant due to collinearity, even though in truth it is the opposite.

VARIABLE SELECTION

There is a family of methods which performs the so-called variable selection or reduction, which is used to obtain reduced dimensionality models, containing only the most important features. Afterward, they are fitted through logistic regression.
They include best subset selection and forward / backward selection method.

BEST SUBSET SELECTION / EXHAUSTIVE

The best subset selection method tries exhaustively all the possible models for any parameters combination and chooses the best one, using $adjusted\ R^2$, Cp, BIC or cross validation prediction error.
This method is guaranteed to find the optimal solution, but on the other hand, it may be computationally expensive for many variables. For *p* variables, 2^p different models must be compared due to all the possible combinations

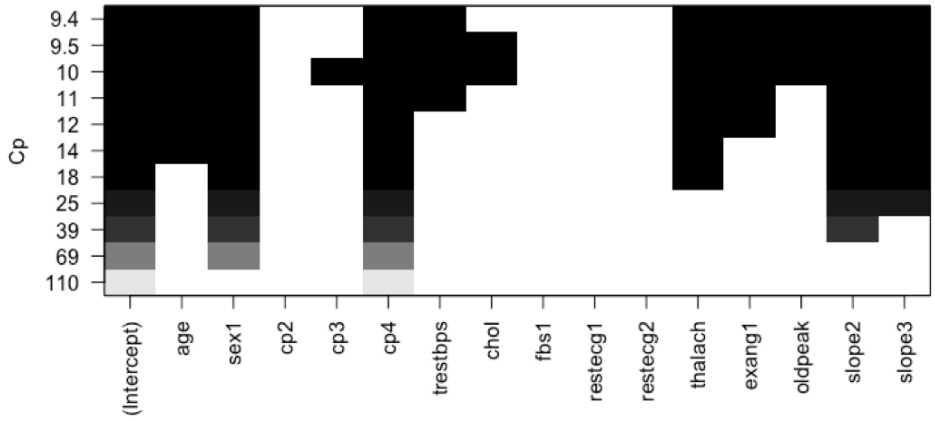

Figure 3: best subset selection Cp value for different variables subsets

FORWARD / BACKWARD STEPWISE SUBSET SELECTION

Forward and backward stepwise subset selection are greedy algorithms, which means that at every step they choose the optimal solution, without considering the overall final result. That's why there is no guarantee that these algorithms will reach the absolute best model in the end, as opposite to exhaustive selection. On the other hand, given they are faster than exhaustive subset selection, requiring to compute only $1 + \frac{p \times (p+1)}{2}$ models, they provide a good proxy for high dimensional data. In general, their results are pretty similar.

The backward selection starts from the full model and iteratively deletes one variable at a time, namely the one which decreases the least the R^2 or that increases the most the RSS (for classification instead of R^2 other metrics called pseudo R^2 are used, such as likelihood ratio).

A single best model for every different number of variables is taken and all of them are compared using $adjusted\ R^2$, Cp, BIC or cross-validation prediction error, with the aim of finding the best final model. These indicators penalize less parsimonious models. That's the reason why indicators such as R^2 and RSS, which always give a higher score to bigger models, cannot be used in this case.
Forward subset selection works in the opposite way, starting from the null model containing only the intercept and iteratively adding the variables which gives the greatest improvement in terms of R^2 or prediction error. The only difference is that in order to apply backward subset selection, the number of predictors must be smaller than the number of data points, while the forward subset selection works with any number of data points in respect to the variables.

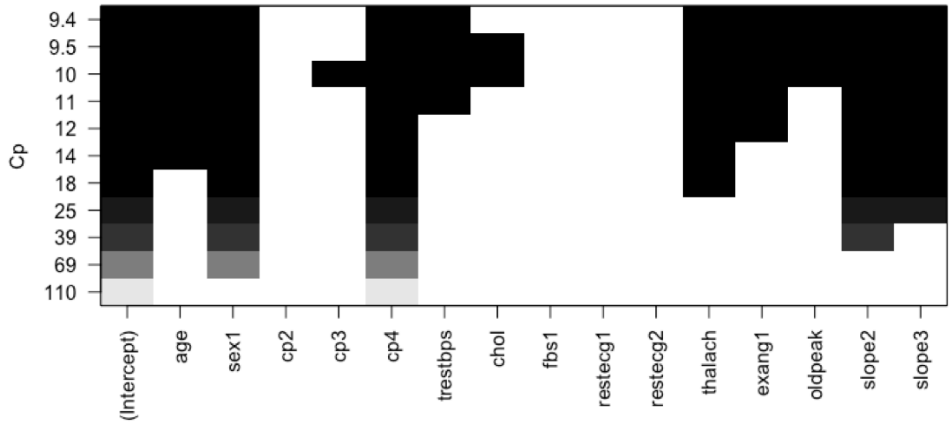

Figure 4: forward stepwise subset selection Cp value for different variables subsets

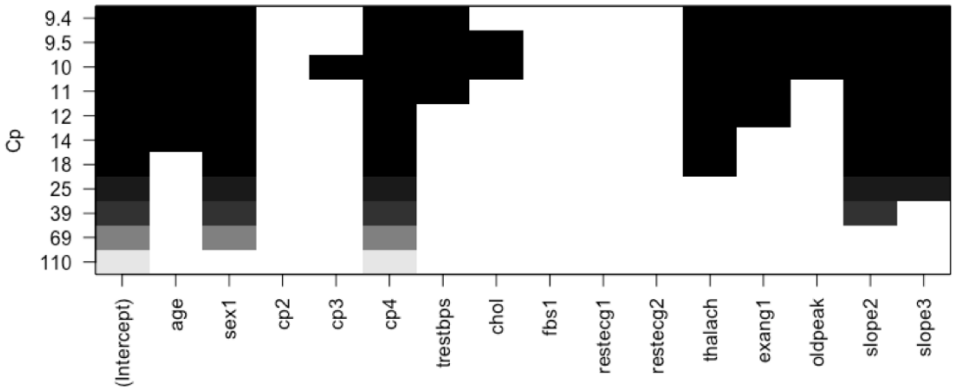

Figure 5: backward stepwise subset selection Cp value for different variables subsets

The plots showing the Cp score for every variables' subset can be obtained. It can be seen that the result is exactly the same for the three methods, so they can be used interchangeably in this case.
The variables selected are those contributing the most to correctly predicting the response, having the lowest Cp. They are:

- age
- sex1

- cp4
- trestbps
- thalach
- exang1
- oldpeak
- slope2
- slope3

The intercept is included. The corresponding Cp value is 9.4

These algorithms don't work optimally with multinomial categorical variables since they split them into dummies and test for their inclusion in the model separately, not as a single variable which should be dropped or kept altogether. As a consequence, it's not clear which features one should select to fit a model. Moreover, by changing the reference category one may end up with different results, since the selection is performed according to the given order.
Two possible solutions would be to repeat the procedure by changing the reference category or to transform integer encoded categorical variables through one-hot encoding, changing them into binary dummies, but some information might be lost. Other more appropriate methods for this specific case can be used instead. An example are the shrinkage methods, which are much faster to implement. they include Ridge regression and Lasso, the latter also performing variable selection.

SHRINKAGE METHODS

Ridge regression and Lasso work by running a linear or logistic regression while shrinking the coefficient size. This is achieved by imposing a penalty on them.
When there is a continuous response, the method minimizes the RSS, while with binomial categorical response it minimizes the negative binomial log-likelihood plus the penalty parameter in both cases. The added parameter is called 'L2 norm' in the case of ridge regression and 'L1 norm' in the case of lasso, which magnitude is controlled through a tuning parameter λ. The higher is λ, the higher the penalty on the coefficients. When $\lambda = 0$, it's like running a normal regression
Ridge regression formula for classification is

$$\sum_{i=1}^{n}\left[y_i x_i \boldsymbol{\beta} - \log\left(1 + e^{x_i \beta}\right)\right] - \lambda \sum_{j=1}^{p} {\beta_j}^2$$

While the formula for Lasso is

$$\sum_{i=1}^{n}\left[y_i x_i \boldsymbol{\beta} - \log\left(1 + e^{x_i \beta}\right)\right] - \lambda \sum_{j=1}^{p} \left|\beta_j\right|$$

The substantial difference between Ridge regression and Lasso is that the former shrinks the coefficients towards 0 as λ increases, but they are never equal to 0, while the latter can produce parameters with 0 value. That's why Lasso is considered also a variable selection method.

LASSO

After fitting the Lasso model, the corresponding plot can be inspected. In this plot all the parameters are shown as a function of Log(λ). On the vertical axis there are the coefficients' values, whereas on the upper horizontal axis the number of non-null parameters. As λ increases, more parameters become null.

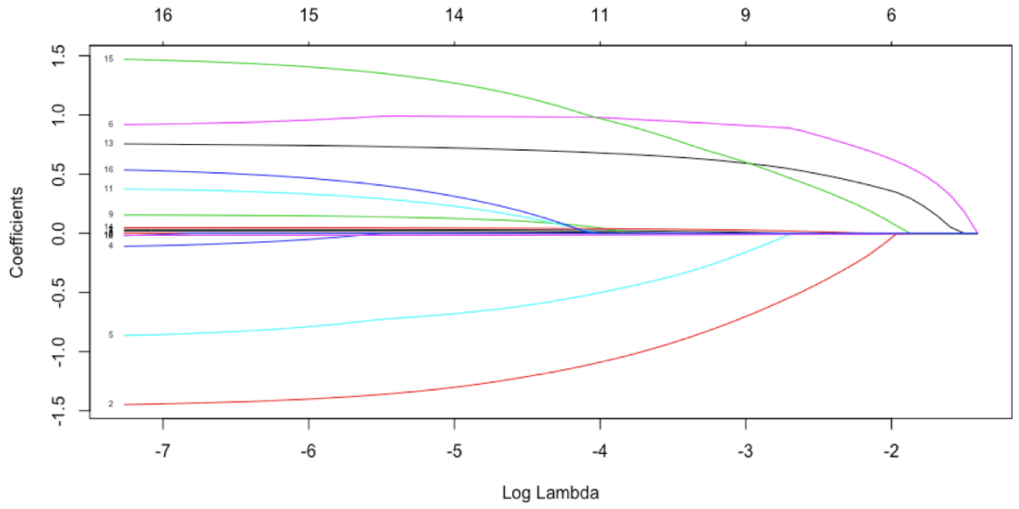

Figure 6: Lasso coefficients against Log(λ)

The optimal λ is still unknown. 10-fold cross-validation is applied to select the hyperparameter λ, but using only the training set data, so that the error can be computed on the validation set according to the validation approach. The proportion used to split the data is 70-30

The optimized fitted model parameters are shown below

```
(Intercept) -1.064926e+00
age          7.581644e-03
sex0        -6.975396e-01
sex1         8.172871e-14
cp2          .
cp3         -1.544978e-01
cp4          9.097191e-01
trestbps     .
chol         .
```

```
fbs1          .
restecg1      .
restecg2      .
thalach     -1.018163e-02
exang1       5.911411e-01
oldpeak      2.591614e-02
slope2       5.951653e-01
slope3        .
```

The parameters selected by the model are 9 out of 16, plus the intercept. It is noteworthy that their value is rather small, due to the fact that Lasso performs parameters shrinkage other than subset selection.

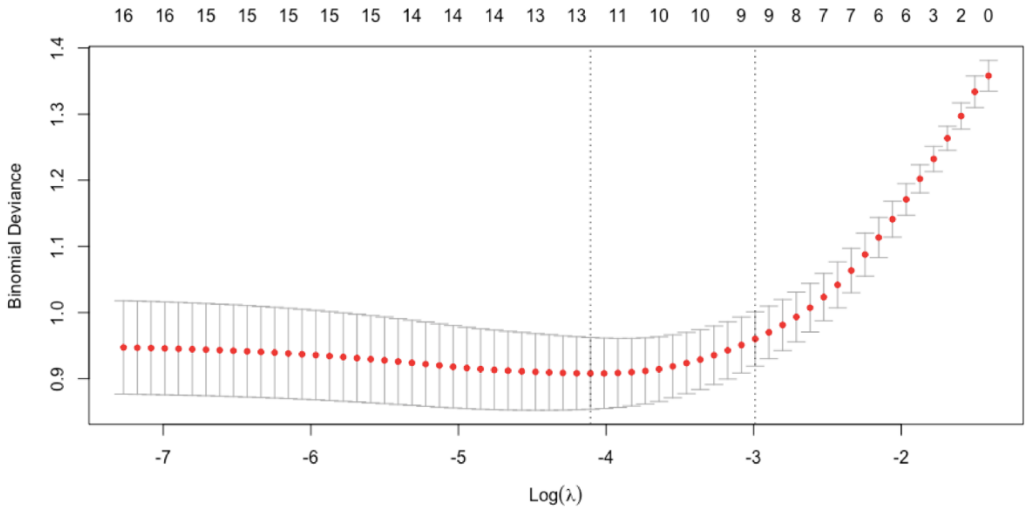

Figure 7: binomial deviance against Log(λ) for Lasso

To understand how the optimal λ is chosen, the plot showing the binomial deviance as a function of Log(λ) can be analyzed. The binomial deviance is a loss function used in classification.
The model selects two different λ, the first one (0.01647649) corresponding to the lowest binomial deviance with 13 parameters and the second one (0.05031683) to the lowest variance with 9 parameters. Other than a model with a low error, it is better to choose a parsimonious model, so the second λ has been chosen.
After having fitted the model, the probabilities can be computed as well through the log-odds, since the Lasso works as a logistic regression.

The recall rate found with the validation approach is 91.94%, which is pretty high.
However, a more reliable way to assess model performance is to use nested *K*-fold cross validation. Nested cross-validation consists in using only a part of the dataset to perform an inner cross-validation in order to find the optimal value for the hyperparameters of the model. Once the parameter is found, an outer cross-validation is performed on the left-out fold to assess the fitting goodness of the model. The procedure is repeated *K* times, changing every time the test fold.

The recall rate for nested 10-fold cross-validation is 86.97%. The average recall rate is slightly worse than before, with classification precision reduced by around 5 percentage points.

RIDGE REGRESSION

Concerning Ridge regression model, the same approach as with Lasso has been used.

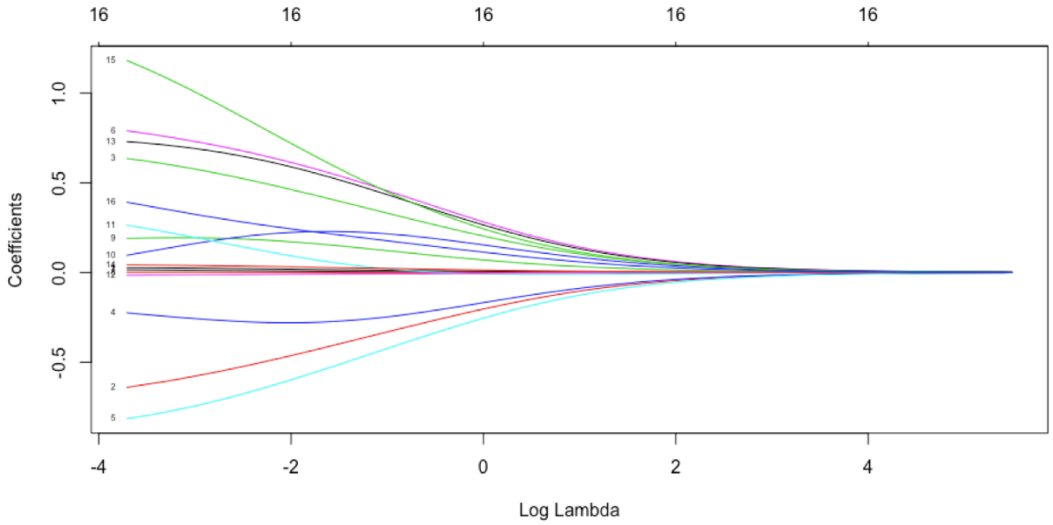

Figure 8: Ridge regression coefficients against Log λ

By looking at the plot showing Ridge regression coefficients as a function of Log(λ), the difference with Lasso is clear. Here no coefficient takes exactly value 0, instead all of them are shrunk towards 0 without becoming null. In fact, the number of non-null coefficients on the upper horizontal axis is always 16, which is the full model. As λ increases, the value of coefficients decreases as with Lasso.

Once again, in order to choose the best value for λ, the cross-validation method is applied only to the training set and then the validation approach used to assess the model performance.
The resulting regression is shown below

```
(Intercept) -1.621998133
age          0.012553342
sex0        -0.344953943
sex1         0.344875952
cp2         -0.253261803
cp3         -0.441938477
cp4          0.473829372
trestbps     0.004102180
chol         0.000691243
fbs1         0.127569194
restecg1     0.219584544
restecg2     0.026317787
```

```
thalach     -0.007262168
exang1       0.451819321
oldpeak      0.023539953
slope2       0.475815575
slope3       0.183859289
```

All the coefficients have been included. Their value has been shrunk, even though not as much as with Lasso.

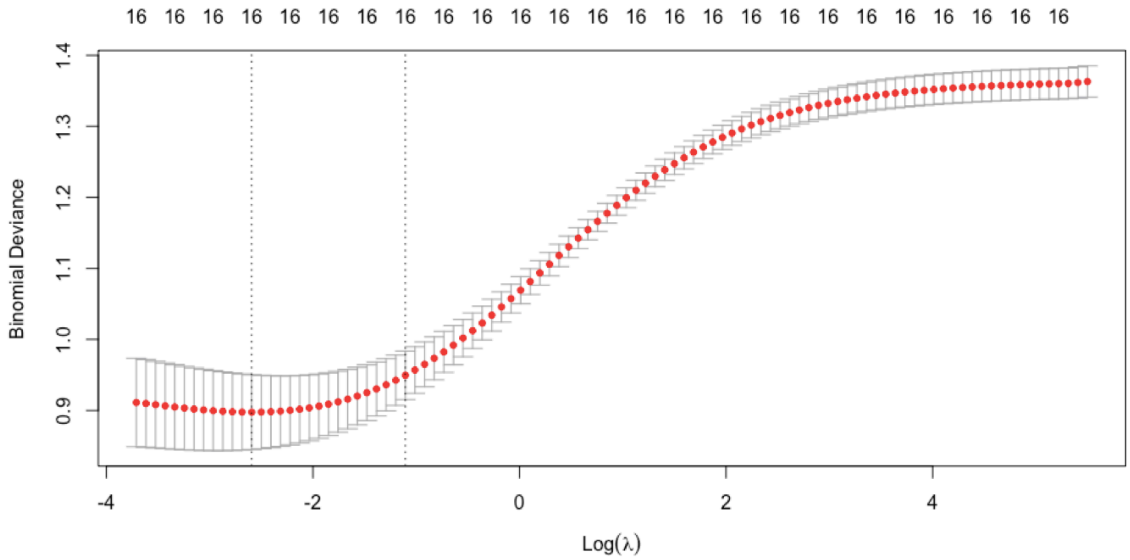

Figure 9: binomial deviance against Log(λ) for Ridge regression

By plotting the binomial deviance against log(lambda), the two possible values of λ are shown. One with the smallest possible binomial deviance (0.07471889) and the other with the smallest variance (0.3310511).

Using the validation approach, the recall rate found is 88.71%. Finally, once again nested cross validation is applied to assess the average performance, returning a recall rate of 83.52%.
By comparing Ridge regression and Lasso, the former (86.97%) performed slightly better than the latter (83.52%) with a difference of around 3 percentage points.
In general, the performance of Lasso and Ridge regression are not so different, none of them is univocally better than the other. In some cases Lasso performs better, especially when there is multicollinearity between variables, while in others Ridge regression dominates, such as when the response is function of many variables with roughly equal size.

CLASSIFICATION TREE

Moving away from the regression family methods, another model which is ideal for working with mixed data type is the tree model.
Trees work by splitting the predictor space into regions and assigning to each observation the mean, if the response is continuous, or the mode, if the response is categorical, of the region

the observation belongs to. Each split is referred to as internal node, while the final regions are called leaves or terminal nodes and the segments connecting each node branches.
The approach used by the tree is top-down, since it starts from the top of the tree, and greedy. The algorithm works by minimizing residual sum of squares in the case of regression.
As regard classification problems there are 3 alternative indices which can be used: misclassification rate,

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

Where $\hat{p}_{mk}$ is the percentage of training observation in the m-th region and k-th class

Gini index

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

and entropy

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

The Gini index measures the node purity, which is given by the percentage of observations of all classes in the final node. If Gini index is small, then there is a predominant single class in the node. Similarly, the entropy takes small values when there is a high node purity. In fact, the two indices are analogous. This means that, other than delivering the final classification, trees also provide the percentage of observations for each class in the final nodes, which is useful to assess the rules effectiveness.
An advantage of trees is that they implicitly perform subset selection by choosing the most important features for splitting, in particular the most relevant ones are used in the first splits, so that it is possible to rank them. Furthermore, they provide a nice graphical visualization of the splitting rules, making them rather easy to interpret.

As before, the dataset is split in training and test set using 70-30 proportion to assess the model and the tree is fitted.

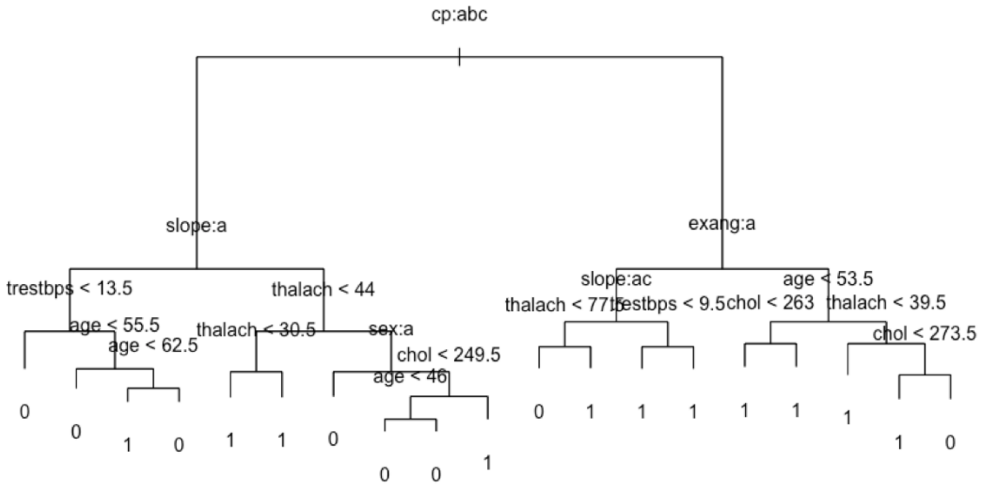

Figure 10: full-grown tree's splitting rules

By graphically inspecting the splitting rules of the tree, it can be seen that the most important variable used for splitting is *'cp'*, then other relevant variables are *'slope', 'exang', 'thalac'*, *'age'* and *'trestbps'*. The number of leaves found by the tree is 19 with 18 nodes, which is quite large, while the recall rate using the validation approach is 83.87%.
The large number of nodes is due to the fact that, given that trees are non-parametric, the splitting process continues until some stopping criterion is satisfied, for instance a minimum number of observations per node or a maximum number of nodes or splits. This may potentially lead to overfit, since the model becomes too complex and it is not able to generalize.
To solve this problem, some bias may be included in order to reduce the variance by growing a smaller tree. Instead of directly producing a small tree, a process called pruning is applied to the full-grown tree so as to reduce it and obtain a subtree. The best subtree is the one with the lowest test error rate, estimated through cross-validation. Instead of comparing all possible subtree, it is faster to apply a method called 'cost complexity pruning' or 'weakest link pruning', which adds to the error function to be minimized a tuning parameter α, multiplied by the number of terminal nodes T. α works as a penalization for having many terminal nodes. The general formula is

$$ERROR + \alpha|T|$$

Different subtrees are computed as a function of α and the value of the parameter is chosen through 10-fold cross-validation. As α increases, the size of the tree decreases since it becomes costlier to have many terminal nodes.

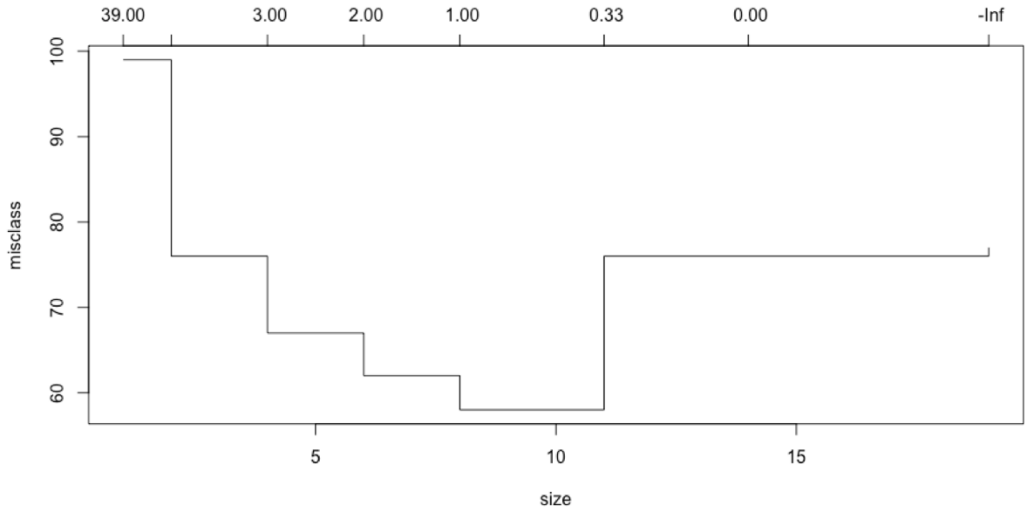

Figure 11: misclassification rate as a function of tree size

The loss function used to drive the pruning of the tree is the misclassification rate and the optimal size of the model is found by using a 10-fold cross validation.
The output provides different sizes of subtrees associated to the cost-complexity parameter α and the cross-validation error. The plot of the misclassification rate as a function of size is

used to select the optimal number of final nodes for pruning the tree, which seems to be within a range from 7 to 11.
In order to choose the best model, it's useful to make a comparation between pruned trees with different size belonging to the range with lowest classification error rate. The one with the lowest Recall rate is then selected to fit the new model.
All the subtrees present the same Recall rate (90.32 %). The optimal size to end up with the most possible parsimonious model is consequently 7.

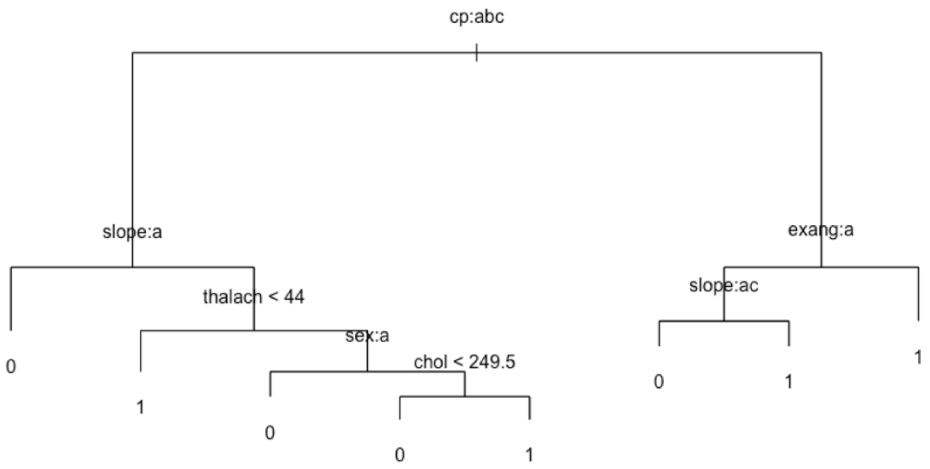

Figure 12: best pruned tree's splitting rules

The result obtained by using the new pruned tree is quite better with respect to the full-grown tree. By adding some bias the variance has been reduced, leading to a better performance on unseen data.
The splitting rules along with the percentage of observations for every class is shown below. In this case there are no pure final nodes, as opposite to the result obtained with the unpruned tree. It is useful to look at the proportion of classes in each leaf in order to decide whether a certain instance can be classified with certainty or not, according to the splitting rules.

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 234 316.800 1 ( 0.4103 0.5897 )
   2) cp: 1,2,3 97 118.300 0 ( 0.7010 0.2990 )
     4) slope: 1 51  36.950 0 ( 0.8824 0.1176 ) *
     5) slope: 2,3 46  63.770 0 ( 0.5000 0.5000 )
      10) thalach < 44 19  19.560 1 ( 0.2105 0.7895 ) *
      11) thalach > 44 27  32.820 0 ( 0.7037 0.2963 )
        22) sex: 0 8   0.000 0 ( 1.0000 0.0000 ) *
        23) sex: 1 19  25.860 0 ( 0.5789 0.4211 )
          46) chol < 249.5 13  14.050 0 ( 0.7692 0.2308 ) *
          47) chol > 249.5 6   5.407 1 ( 0.1667 0.8333 ) *
   3) cp: 4 137 138.800 1 ( 0.2044 0.7956 )
     6) exang: 0 33  45.470 1 ( 0.4545 0.5455 )
      12) slope: 1,3 16  19.870 0 ( 0.6875 0.3125 ) *
```

```
      13) slope: 2 17  18.550 1 ( 0.2353 0.7647 ) *
     7) exang: 1 104  78.370 1 ( 0.1250 0.8750 ) *
```

The most important features selected by the pruned tree are, in order of importance, *'cp', 'slope'* and *'exang', 'thalach', 'sex'* and *'chol'*. The model operated a feature selection, using only 6 of them out of 16.

Tree models are considered weak learners, since they generally don't provide a high predictive accuracy compared to other models. Furthermore, they are not very robust, since the result may vary a lot when the training data changes. This is due to the fact that there is a tradeoff between interpretability, obtained using simpler models, and prediction accuracy, which can be increased by more complex models. In this case we obtained a quite good accuracy with tree model, but it's better to use 10-fold nested cross-validation to evaluate whether this result depends on the dataset splitting.
The final result is an average error rate of 84.94%, which is not significatively worse with respect to the previous results. This means that the classification tree worked well on the considered dataset.

ENSEMBLE METHODS

The next step is to apply ensemble methods which are a family of models able to enhance weak learners, such as trees, by pooling them together aiming to build a strong classifier. They include bagging, random forest and boosting. It is worth running them to see whether they can increase tree performance.

BAGGING

Bagging, which stands for Bootstrap aggregation, is a method that reduces the variance by averaging a set of independent predictions. The methods try to build independence by bootstrapping the training data set many times and building a predictor for each subsample. Bootstrapping means taking subsamples from the original training set by randomly sampling some observations with replacement, in order to obtain many training sets. It's useful especially when working on a small training set. It can be proved that on average 2/3 of the data points appear in each subsample.
Bagging can be applied to different models. Trees are good candidates since they suffer from high variance, while having low bias, as a consequence averaging many models leads to a lower variance still holding the same bias. The trees grow deep without pruning, so that they have low bias.
When the underlying task is regression, bagging takes the average of the predicted values, while when the model does classification, a majority vote is taken to determine the most voted class. If bagging is able to obtain a fair amount of independence by bootstrapping, the prediction accuracy improves. One of the drawbacks of bagging compared to trees, is that the interpretability advantage is lost since it's no longer possible to visualize the tree.
The number of trees used is not so important, using many trees doesn't lead to overfitting. A proper choice would be to use a rather high number of trees while keeping a reasonable

computational time. That's because for large datasets it can be quite long to compute hundreds of tree models.

As for the previous models, the bagging one is fitted on the training set by splitting the dataset according to a 70-30 proportion. Then the recall rate is calculated on the training set, in order to compare it later with OOB. The training error is 76.55%, so the corresponding recall error is 23.45%.
No parameters need to be tuned in this case. The number of trees used is 500, which is a high enough number.

Another useful metric directly computed within the model is the out-of-bag error estimator, which returns the error rate for every observation. It is equivalent to the leave-one-out cross validation error, which is found by running a cross-validation with as many folds as the number of observations in the dataset. When the single trees are grown, around 2/3 of data is used to train the model, which is then tested on the remaining 1/3 of data. The procedure is repeated for every tree and the resulting error (MSE or misclassification rate) is averaged for every observation.
The OOB error rate is 20.42%, pretty close to the recall error rate (23.45%) even if the metrics are different.

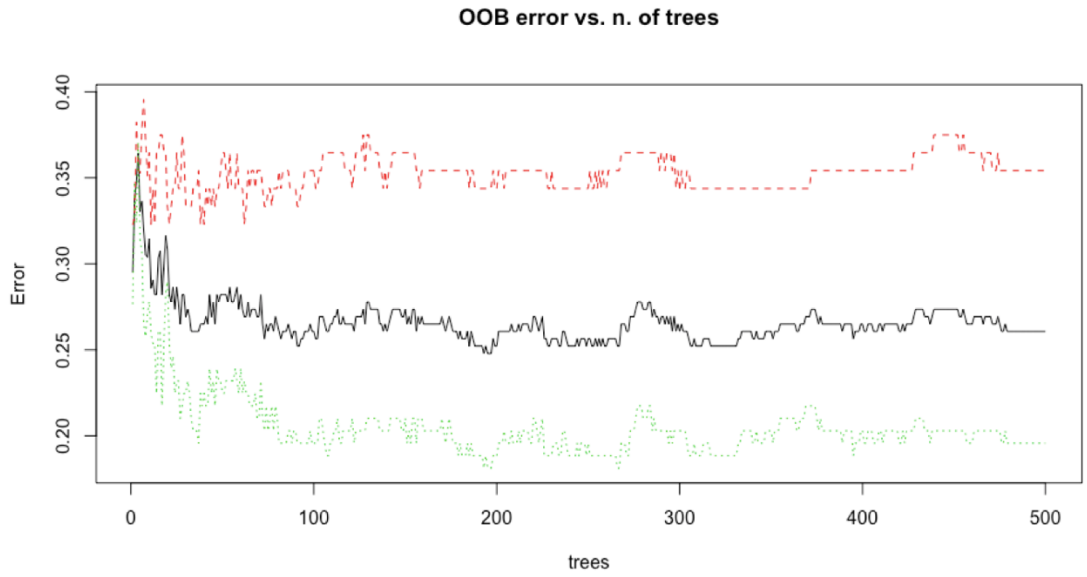

Figure 13: bagging OOB error for different number of trees

By plotting the OOB error for both responses, it can be seen that the misclassification rate of the negative responses (in red) is far higher than the one of the positive responses (in green). This is not a positive finding, given it is more important to correctly identify sick patients rather than the opposite. It would be preferable to obtain a lower misclassification rate for negative responses.
Since few trees are not enough to properly reduce variance, the error rate is quite variable with this number of trees. OOB seems to become more stable after around 300 trees.

Another useful feature of bagging is the possibility to obtain a ranking of the variables according to their importance. The two metrics used to assess it are:

- The mean decrease of accuracy in the predictions made on the out of the bag samples when the variable is excluded from the model
- The mean decrease in the Gini index, which is the decrease in node impurity due to the use of the variable for making splits. It's a measure of the variability of the expected value of a node.

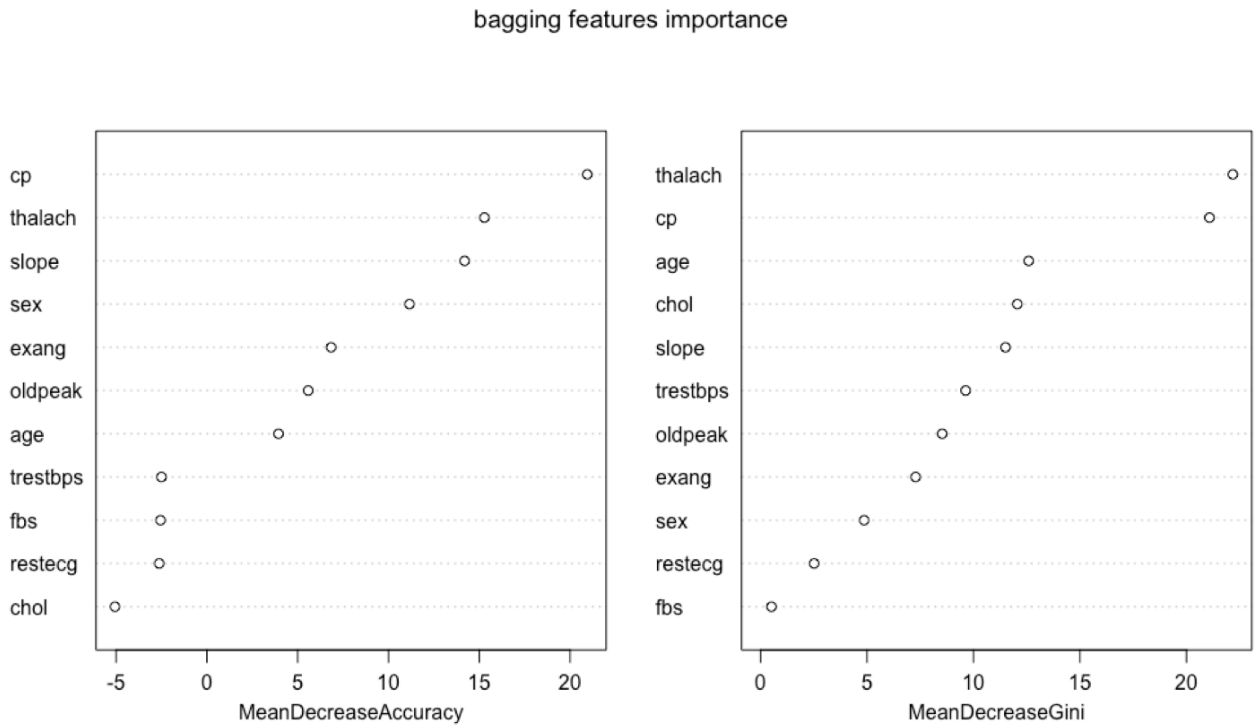

Figure 14: ranking of features importance according to bagging

The features importance ranking found by the mean decrease in Accuracy and in Gini index are different. Only 3 of them appear among the top 5 variables according to both indices. They are '*cp', 'thalach'* and *'slope'*.

The recall rate found using the validation approach is instead 91.93%.
One again, 10-fold cross-validation is used to assess the model performance. The average recall rate of bagging is 81.88%, which is lower than the average tree recall (83.52%). It can be concluded that on this dataset bagging didn't improve the tree prediction.
Even though the idea behind Bagging theoretically works, namely obtaining independence by bootstrapping the original dataset, problems may arise in certain contexts. This happens when some predictors are stronger than others and as a consequence they are included nearly always in the first splits of all the trees. The different trees are thus correlated, so the decrease in variance is not as high as expected.

RANDOM FOREST

To deal with situations in which bagging is not suitable, random forest performs a further randomization by subsampling the features to be used at each node split, providing a stronger basis for independence. In this way, even if there are stronger predictors, there are more chances for weaker ones to be chosen as nodes splitting variables. Unlike bagging and boosting which can be applied to all types of models, random forest is built using only trees, as suggested by its name.
Cross-validation is usually used to choose the optimal number of predictors, but as a rule of thumb the square root of the total number of variables is generally used. Random forests perform particularly well when there are many correlated features. As with bagging, the number of trees used is not important, as long as there are enough of them.
A further parameter which can be tuned is the tree depth, which is the maximum distance from a terminal node to the initial node, in order to control the speed of computation for large datasets. However, since the dimensionality of data is low, it is not necessary and the focus will be only on tuning the number of features to split the nodes of the tree model, being the most important hyperparameter.

Instead of starting by fixing a number of subsampled predictors equal to $\sqrt{11} \approx 4$, the optimal parameter is directly found using the 10 cross-validation approach on the training set. Afterwards, the performance is calculated on the validation set. The recall rate is 93.55%

The optimal number of subsampled predictors at each node is 3.

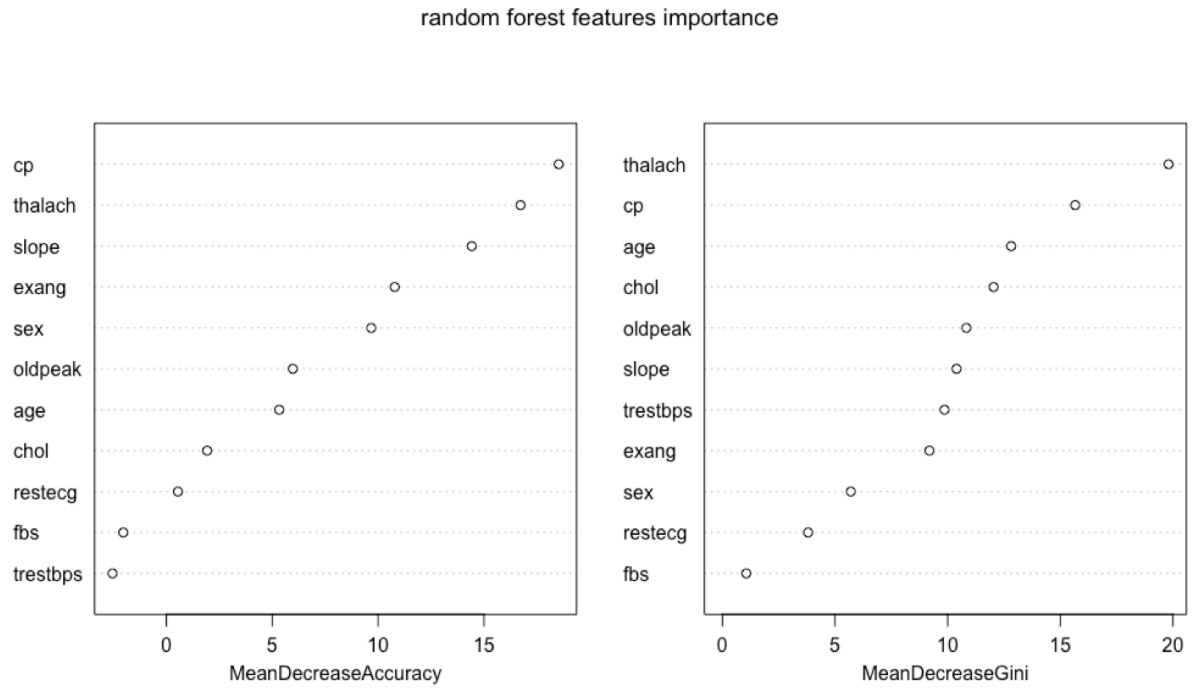

Figure 15: ranking of features importance according to random forest

Regarding the variables' importance ranking, only three of them are considered to be important by both indices. Moreover, they are the same found with bagging, which are *'thalach', 'cp'* and *'slope'.*

Nested cross-validation can be used to assess the average performance of the model on the dataset. The recall rate for random forest is on average 86.18%
On this dataset random forest works better than bagging, improving the recall rate by approximately 5%. Probably this is due to the fact that there are stronger predictors creating some correlation within trees in bagging.

BOOSTING

The last ensemble method used to analyze the dataset is boosting. Like bagging it can be applied to different methods. However, it uses a different approach compared to Bagging and Random forest, since it is a sequential algorithm. It reaches the independence by sequentially growing trees using the information from the previous fitted ones.
There are different types of boosting methods, such that Gradient boosting and AdaBoost (adaptive boosting). Gradient boosting adjusts the model by fitting the residuals of the previous model, using them as the response variables. After, the residuals are updated and fitted again. Each model takes a step in the direction that minimizes the prediction error, namely its gradient.
Gradient Boosting relies on a loss function, which is used to compute the gradient. The loss function assesses how good the prediction made by the model is. Any differentiable loss function can be used with boosting, such as mean squared error for regression or logarithmic loss for classification.

AdaBoost works by changing the weights of the observations at each step. The weights of the misclassified instances are increased and those of the correctly predicted instances are decreased, so that the new fitted model takes more into account the formers than the latter. A sequence of weak classifier is produced, one classifier at each step, and their prediction is combined using a weighted majority vote. The best classifiers have higher influence on the final result. AdaBoost is the basic model used for binary classification

The main disadvantage of boosting is that it takes more time to fit the final model, being a sequential algorithm. On the other hand, it outperforms the other methods, since slow learners tend to perform better.
As opposed to bagging and random forest, boosting is very likely to overfit. That's why is recommended to use some constraints, such as tuning the number of trees and their depth. Tree's depth refers to the number of edges in order to reach the final nodes. Often stumps are used so that complexity is reduced. Other alternatives are to control for the minimum number of observations per split or for the minimum improvement of the loss.
Another parameter which has to be tuned is the shrinkage parameter that controls the learning rate of boosting by determining the contribution of each tree on the final outcome. When it is small, the boosting learns slowly because the impact of each tree is shrunk. The process takes more time since it requires a higher number of trees, but the final performance is improved. Typical values for the tuning parameter are 0.01 and 0.001.

Gradient boosting is applied first.
There are three parameters that need to be tuned: tree depth, number of trees and learning rate. Moreover, the minimum number of observations per node is fixed at 10. The learning rate can be set by default equal to 0.001, leading to a higher number of trees required. This can be done since the dataset is small, so computing many trees doesn't require much time. The method used in a 5-fold cross-validation repeated three times.
The final parameters used by the model are 4600 trees, 1 as maximum tree depth, a minimum number of observations per node of 10 and a shrinkage parameter of 0.001.
The recall rate obtained is 93.55%, while the average recall rate obtained through nested cross validation is 88%, which is quite good.

Next, the model is fitted using a fast implementation of AdaBoost. The hyperparameter to be tuned is the number of iterations before stopping through a 5-fold nested cross-validation
The optimal number of iterations found by AdaBoost.M1 is 100.
The recall rate from the validation set is 90.32%, while that from the nested cross validation is 80.58%. AdaBoost was not able to improve estimates as opposed to Gradient boosting (88%).

MODELS NOT CONSIDERED FOR THE ANALYSIS

To perform this analysis, some widely used models such as non-linear models, principal component regression (PCR), partial least square (PLS) and neural networks have been excluded a priori.

The non-linear models are a family that comprehends a variety of methods used to deal with nonlinear behaviors, which are not captured by basic linear models such as linear or logistic regression. The simplest ones are polynomial regression and step functions, considered as an extension of linear models. The former adds to the original coefficients of the standard linear model the same coefficients but raised to a power, providing more flexibility in fitting the data. Usually, degrees greater than 3 or 4 are not used so to avoid overfitting.
The latter model fits a piecewise constant function, defining some cutpoints based on natural breakpoints in the data, provided they exists.
In addition, more sophisticated methods can be found, such as regression splines and generalized additive models (GAM). Regression spline provides an improvement to step functions by introducing piecewise polynomial regression instead of a linear one. Here the cutpoints where the function changes are called knots. With K knots, $K + 1$ polynomials are fitted. Moreover, splines impose a constraint at the knots so that the function is continuous in derivatives up to degree $d - 1$, where d is the degree of the spline. This allows the function to be very smooth.
A problem that may arise is the fact that variance tends to be quite high in the boundary regions. To solve it, further constraints can be included thereby the function is linear at the boundaries. This model is known as natural spline.
Finally, GAM is a general model, thanks to which it is possible to fit different non-linear models to each variable and then consider their effect additively.
The reason why they have not been considered for this analysis is for the sake of simplicity and time management, since non-linear models might be difficult to tune without having hints

on the shape of data. Moreover, given the large number of features and the presence of mixed type data, it is not clear how to analyze them graphically in order to identify non-linear patterns. To avoid undertaking a possibly long process of trial and error, other more immediate models have been used.

Further interesting methods are PCR and PLS, which work by reducing data dimensionality. PCR is the application of principal component analysis (PCA), that is an unsupervised learning method, in order to derive a set of features called principal components expressed as linear combination of the original ones.
The first component is built so that it captures the highest possible variance of data, then the next component explains the remaining highest possible variance, constrained to be orthonormal to the first and so on. In the end the number of features obtained is equal to the initial one, however only the first principal components are useful in explaining the data, so they are used to fit a new low dimensional linear regression.
The drawback of this method is that it doesn't take into account the effect of the response Y. It may end up by including in the principal components the variables that better explain data variability, although they are not related to the response variable.
PLS adapts itself better to the problem at hand, as it makes use of the label Y in building new features. Then the process is the same as with PCR
Provided that is possible to apply PCR and PLS to binary and categorical variables, it is not recommended since these methods have been originally thought to be applied on continuous variables, relying on data variance. Due to the fact that the Heart Disease Dataset contains both binary and categorical features, in order to apply PCR and PLS the variables should be transformed somehow. Moreover, they may lose their meaning and interpretability.

Finally, the last model that has been excluded are neural networks. Neural networks' basic computational unit is the neuron, which operates similarly to a linear or logistic regression. It works by taking some values as input and returning a single output obtained by applying a so-called activation function. This has the function of introducing some non-linearity in an otherwise linear model.
The main strength of this method is that of combining such simple elements in order to create a more complex structure. The network is composed by an input layer, an output layer and possibly one or more hidden layers in the case of feed-forward neural networks. Each layer is in turn made of one or more neurons, which take as input the output of the neurons in the previous layer to which it is linked and produce an output that is passed to the nodes in the next layer. Neurons are connected through weights, which role is that of scaling the inputs. A special type of neuron called bias neuron, which transmits a fixed value, can be used if needed. The learning process occurs by changing both weights and biases to obtain the best possible prediction. In order to assess the goodness of the prediction, it is important to choose the appropriate loss function
Along with the choice of the architecture of the network, meaning the number of layers, neurons and connections, it is fundamental the type of activation function chosen. Among the most popular ones there are the ReLU, the Sigmoid and the Hard Than.
After having defined the architecture, the backpropagation algorithm is applied in order to train the network. In the case of perceptron, which is the network with no hidden layers, weights and biases are trained by using the stochastic gradient descent method. The algorithm works by taking the gradient of the loss function with respect to the weights and

update them in the direction where the gradient is negative, which is the direction in which the loss is minimized. The size of the steps taken at each update is regulated by the learning rate. The optimal values for the weights are reached when they stop changing or the change is too small.
On the other hand, when dealing with a multilayer neural network, the stochastic gradient descent cannot be directly applied, instead the gradient is computed through the backpropagation algorithm. This is based on the chain rule of differential calculus.
The process takes place into two phases, which are the forward and backward phases. During the forward phase, the predicted output returned by using the current weights and biases is compared to the true output labels. Then, the derivative of the loss function with respect to the output is computed. Afterwards, the backward phase consists in calculating the gradient of the loss function with respect to the different weights by exploiting the chain rule of calculus. The weights are thus updated according to the gradients found. The process is repeated until convergence.
The initial value of weights and biases is randomly chosen. Since the initialization point can have a great impact on whether a good or a bad local optimum is reached, it is important to choose it wisely.
The reason why this method has been discarded is due to the small size of the dataset, which neural networks are not particularly suited for. One of their drawbacks is that they are overparametrized models which easily lead to overfitting, especially when the number of observations in the dataset is small. On the contrary, they work quite well with large datasets, being able to capture complex patterns thanks to their large number of parameters.
A simpler neural network can still be applied to a small dataset. However, it takes time to choose the proper architecture and tune it, further it provides no additional advantage over more basic models for this type of analysis. That's why it has been opted for simpler methods

CONCLUSION

The following table summarizes the findings of the analysis on the Heart Disease Dataset.

MODEL	Validation approach Recall Rate (%)	Nested cross-validation Recall Rate (%)
Logistic Regression	95.16	89.51
Lasso	91.94	86.97
Ridge Regression	88.71	83.52
Tree classifier	90.32	84.94
Bagging	91.93	81.88
Random Forest	93.55	86.18
Gradient Bosting	93.55	88.00
AdaBoost	90.32	80.58

In general, all the methods seem to predict the data pretty well, with an average recall rate between 80% and 90%.

By looking at the table, a reduction in prediction when applying the cross-validation approach compared to the validation one can be seen. Usually, the cross-validation methods provide more reliable performance assessments, since they are less affected by the dataset splitting. The two models with the highest performance are logistic regression and gradient boosting, while the two which performed worst are AdaBoost and Bagging. Despite being considered a weak learner, tree classifier showed a good predictive capability when dealing with new data. Thus, given its nice graphical interpretability, it may be a proper candidate to be used on this dataset.
However, since one of the main goals of this analysis is to obtain the best possible prediction, the choice would be between logistic regression and gradient boosting. In terms of model complexity and required computational time, logistic regression seems the natural best choice. In addition, it provides the opportunity to calculate probabilities of belonging to a certain class for every observation.
Another point of interest, is the fact that bagging wasn't able to improve tree classifier, which already had a good performance, whereas random forest improved it by about 1 percentage point. This enhancement is not enough to justify the loss of interpretability and the increased computational time.

Regarding variables importance, information can be obtained both from subset selection methods, including Lasso, and from tree classifier along with bagging and random forest.
The common features selected by the different subset selection methods and Lasso *are 'age', 'sex', 'cp', 'thalach'* and *'exang*'. It is however difficult to assess the importance of categorical variables such as *'exang', 'sex'* and *'cp'*, since these methods consider the different levels as single variables
On the opposite, tree, bagging and random forest consider the feature instead of the dummy variables created from it. They are in accordance regarding the most important features chosen, which are *'cp', 'thalach'* and *'slope'*. All of them are present also among the variables used by the previous methods. It can be concluded that the features contributing the most in explaining the response, namely whether a patient has or not heart diseases, are *'cp'* (chest pain type), *'thalach'* (maximum heart rate achieved) and *'slope'* (slope of the peak exercise ST segment).

The most relevant findings emerged from this study are summarized as follows:

1. All the supervised learning models applied to the Heart Disease Dataset delivered a pretty good predictive performance;
2. The best model in terms of performance, model complexity and computational time is the logistic regression;
3. Tree classifier performed well on the dataset, despite being considered a weak learner
4. Ensemble methods, with the exception of gradient boosting, didn't significantly improve the predictive power of tree classifier;
5. The most significative variables in explaining the target variable (whether a patient has heart diseases or not) seems to be *'cp'* (chest pain type), *'thalach'* (maximum heart rate achieved) and *'slope'* (slope of the peak exercise ST segment).