Statistical Homework 2

Miglioranza Ettore

This analysis examines the relationship between diabetes progression one year after the initial baseline observations and a set of ten clinical indicators in a sample of 442 patients. The response variable (**progr**) measures the degree of diabetes progression, with higher values indicating worse health outcomes. The explanatory variables include: **age** (in years), **sex** (coded as 1 for male and 2 for female), **BMI** (body mass index), **BP** (blood pressure), **TC** (total cholesterol), **LDL** (low-density lipoprotein cholesterol), **HDL** (high-density lipoprotein cholesterol), **TCH** (the ratio of total cholesterol to **HDL**), **TG** (log-transformed triglycerides), and **GC** (glucose concentration). In the following analysis, we aim to compare the goodness of fit of different models with regard to the branch of decision tree models. In particular, we want to compare the results of a regression decision tree, an ensemble of multiple regression decision trees, also called random forest, and finally a boosted regression decision tree. To ensure reproducibility and robustness, we use cross-validation for fine-tuning and selection of hyper-parameters.

Decision Tree

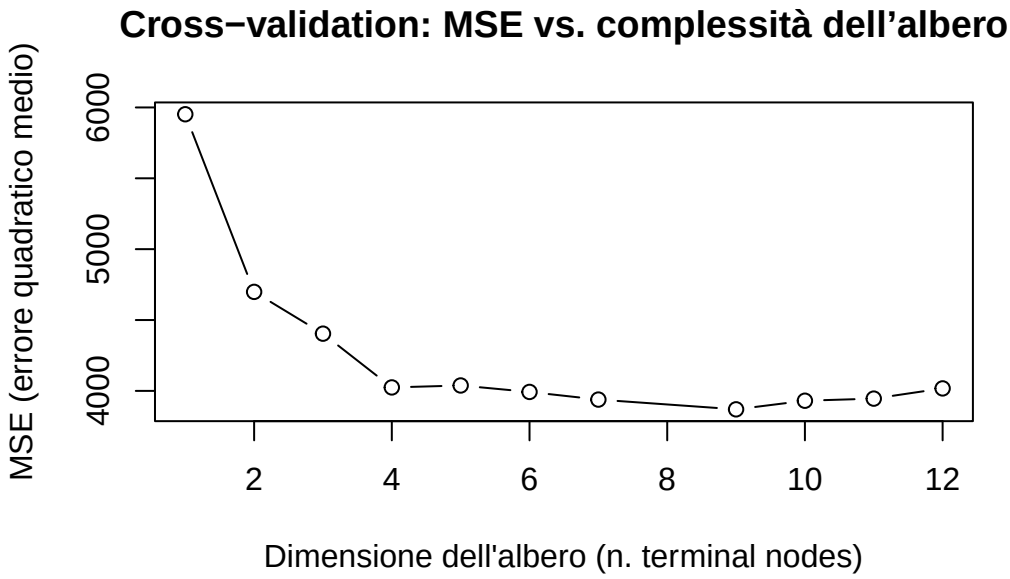
After this brief look at the distribution of response variables, we begin training our first model on the data. The first model will be a regression decision tree. In R's *tree* package there is no maximum depth set by default, so the model stops performing division operations and therefore generates a leaf whenever the node (leaf) has less than a certain number of observations, controlled by the *minsize* parameter (default is 10). An interesting thing to note is that by default, the R package does not use MSE (mean square error) to decide which predictor is best suited to perform the split. Instead, the default split function is based on minimizing the sum of residual squares (RSS). However, after training the model on all the data, the output of the model shows the most informative chosen predictors 'TG' 'BMI' 'HDL' 'GC' 'BP', the number of terminal nodes, which is 12, and the residual mean deviance, which is 2674.

```r
set.seed(123)
tree_model <- tree(progr ~ ., data = db)
summary(tree_model)
```

```
Regression tree:
tree(formula = progr ~ ., data = db)
Variables actually used in tree construction:
[1] "TG"  "BMI" "HDL" "GC"  "BP"
Number of terminal nodes:  12
Residual mean deviance:  2674 = 1150000 / 430
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-140.900  -35.830   -4.805    0.000   33.540  154.100
```

We adjust the number of leaves using cross-validation procedure to determine the optimal tree size. The goal is to identify the best number of leaves k, that minimizes model complexity by removing unnecessary splits, therefore achieving a good balance between bias and variance. This time, we compute the Mean Squared Error (**MSE**) for each tree of varying complexity and use it as the criterion to fine-tune k.

```r
set.seed(123)
# Cross-validation for leaves number
cv_tree <- cv.tree(tree_model)
# Number of observations in the dataset
n <- nrow(db)
# MSE for each tree with different complexity
mse_values <- cv_tree$dev / n
# Plot with MSE
plot(cv_tree$size, mse_values, type = "b",
     xlab = "Dimensione dell'albero (n. terminal nodes)",
     ylab = "MSE (errore quadratico medio)",
     main = "Cross-validation: MSE vs. complessità dell'albero")
```



We prune our initial tree model with the fine tuned hyper-parameter k number of leaves and we show the output again to assess the structure and performance of the model.

```r
set.seed(123)
best_size <- cv_tree$size[which.min(cv_tree$dev)]
pruned_tree <- prune.tree(tree_model, best = best_size)
summary(pruned_tree)
```

```
Regression tree:
snip.tree(tree = tree_model, nodes = c(4L, 28L, 26L))
Variables actually used in tree construction:
[1] "TG"  "BMI" "GC"  "BP"
Number of terminal nodes:  9
Residual mean deviance:  2864 = 1240000 / 433
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-140.900  -37.310   -6.325    0.000   34.150  156.700
```

The final pruned tree identified four key predictors: **TG**, **BMI**, **GC**, and **BP**. **TG** was selected for the initial split, highlighting it as the most influential variable in minimizing the residual sum of squares. **BMI** appeared multiple times across different branches, emphasizing its

critical role in capturing non-linear effects on disease progression. **GC** and **BP** were utilized in lower-level splits, indicating that they help fine-tune predictors within specific subgroups. Interestingly, **HDL**, which was included in the full, unpruned tree, was eliminated during the pruning process, suggesting its relatively minor contribution compared to the retained variables.

Random Forest

To enhance the performance of a single regression decision tree, it is common to deploy an ensemble of different simpler decision trees, also called 'weak learners', and to average their prediction. This procedure is called Random Forest in the case of decision trees, but it can be applied to all kind of models. We test here the Random Forest technique and we keep a record of how much each predictor contributes to making accurate predictions.

```
set.seed(123)

rf_default <- randomForest(progr ~ ., data = db, importance = TRUE)
print(rf_default)
```

```
Call:
 randomForest(formula = progr ~ ., data = db, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 3229.008
                    % Var explained: 45.55
```

The MSE of the Random Forest model is initially higher than that of the fine-tuned decision tree. However, before drawing any conclusions, it is necessary to fine-tune the Random Forest model. We do this by exploring different values of the *ntree* parameter, which controls the total number of trees grown in the forest. For each value of *ntree*, we measure the Out-of-Bag (OOB) MSE, which is the average error calculated across all trees, based on the predictions made on their respective out-of-bag validation sets.

```
set.seed(123)

# Try different numbers of trees
ntree_vals <- seq(10, 1000, by = 10)
oob_errors_ntree <- sapply(ntree_vals, function(n) {
```

```
  rf <- randomForest(progr ~ ., data = db, ntree = n, importance = TRUE)
  return(rf$mse[n])  # get the OOB error after growing 'n' trees
})
best_ntree <- ntree_vals[which.min(oob_errors_ntree)]
cat("Miglior ntree:", best_ntree, "\n")
```

```
Miglior ntree: 670
```

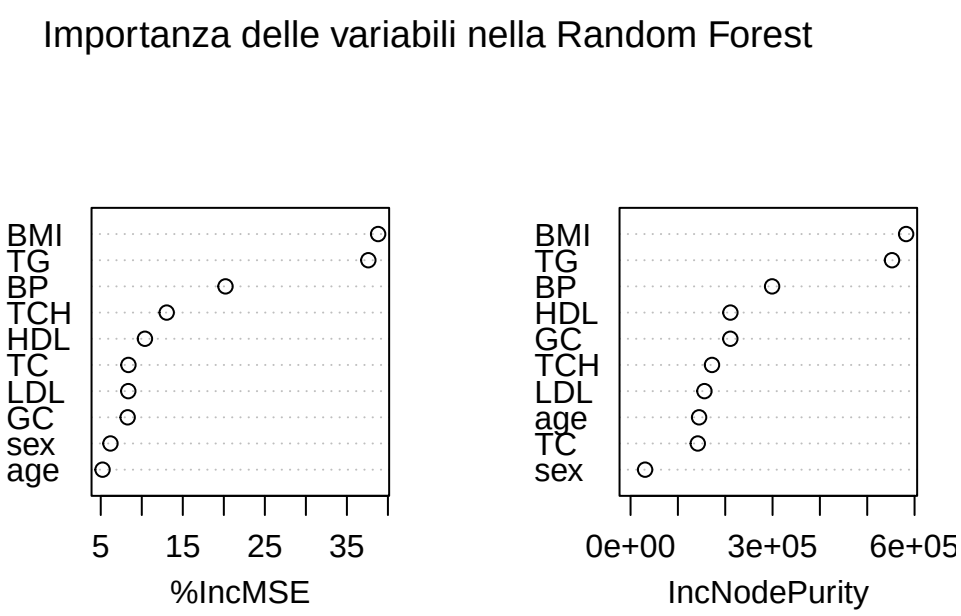
Now that we have performed a complete search for the best number of trees, which turned out to be 670, we can train a better-tuned Forest random model.

```
set.seed(123)
rf_best <- randomForest(progr ~ ., data = db, ntree = best_ntree, importance = T)
print(rf_best)
```

```
Call:
 randomForest(formula = progr ~ ., data = db, ntree = best_ntree,      importance = T)
               Type of random forest: regression
                     Number of trees: 670
No. of variables tried at each split: 3

          Mean of squared residuals: 3222.578
                    % Var explained: 45.66
```

The results clearly show that tuning the ntree parameter helped stabilize the Random Forest model, improving performance. In particular, even after fine-tuning, the MSE remains higher than that of the fine-tuned decision tree. This difference is mainly attributable to the way in which error is evaluated: the Random Forest reports Mean Quadratic Residuals based on out-of-bag (OOB) predictions, i.e. predictions made on subsets of data not seen by individual trees during training, effectively providing a measure of internal validation. In contrast, the decision tree was trained and evaluated on the entire dataset without an explicit validation step, which probably resulted in an optimistic (lower) error estimate. The Random Forest model explains about 45.66% of the variance of the target variable (progr), a moderate result, capturing almost half of the variability. Furthermore, the model benefited from the growth of 670 trees: although slightly larger than the typical default of 500 trees, it was not dramatically larger. After reaching about 670 trees, the OOB error stabilized, indicating that the addition of more trees would not lead to further substantial improvements.

```
      %IncMSE IncNodePurity
age  5.220810     144745.99
sex  6.182869      30670.99
BMI 38.804910     582163.79
BP  20.204136     299009.82
TC   8.370771     141964.79
LDL  8.355604     156060.69
HDL 10.384045     210924.72
TCH 13.035813     172027.10
TG  37.612116     552324.47
GC   8.278500     210749.80
```

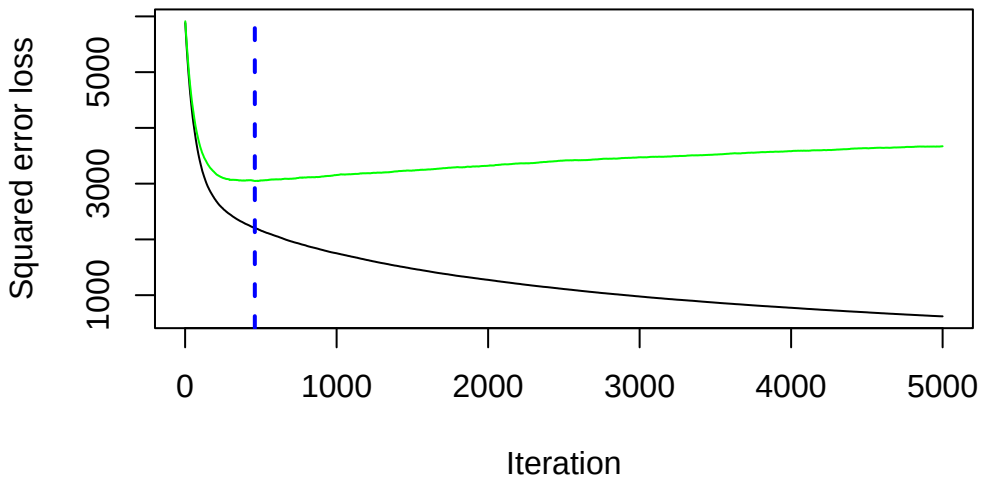


The provided plots highlight the importance of each variable in predicting the response. As shown, the most influential predictors are **BMI**, **TG** and **BP**, indicating that they are the primary drivers of the model's predictions, both clinically and practically. A clear gap can be observed between these top variables and the rest, suggesting that a small number of predictors dominate the model's behavior. Additionally, variables with a %IncMSE very close to zero, such as sex and age, might be considered as noise or irrelevant features within the predictive context.

Boosted Regression Tree

We trained a Gradient Boosting Machine (GBM) model to predict the response variable progr using all available predictors in the dataset db. The GBM trains many small trees sequentially, each improving the errors made by the previous one. The model was configured for regression tasks by specifying a Gaussian distribution. We set the number of trees to 5000, with an interaction depth of 3, allowing the model to capture up to three-way interactions between predictors. A learning rate (shrinkage) of 0.01 was chosen to ensure gradual, stable learning and to reduce the risk of overfitting. Additionally, 5-fold cross-validation was employed to monitor performance and assist in selecting the optimal number of trees, which turned out to be 460. The model automatically used all available CPU cores for computation (n.cores = NULL), and verbose output was disabled to maintain a clean log. A random seed was set for reproducibility.

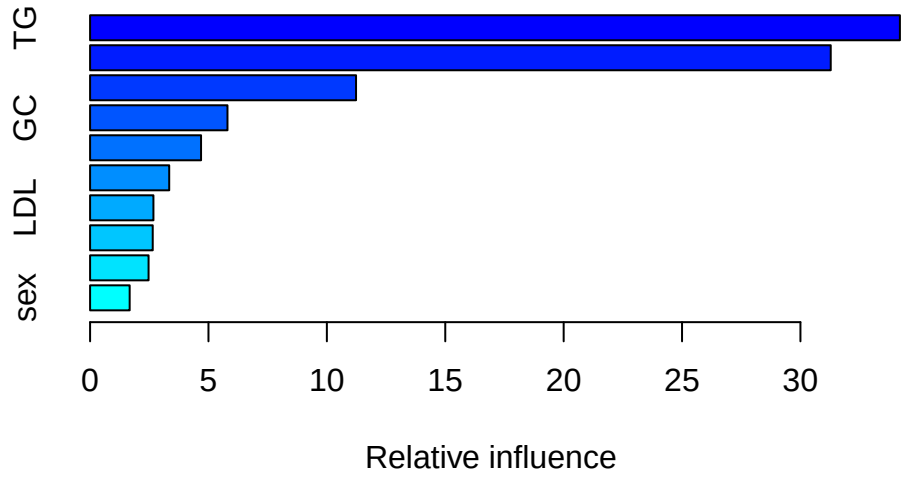
```
set.seed(123)

gbm_model <- gbm(
  formula = progr ~ .,
  data = db,
  distribution = "gaussian",
  n.trees = 5000,
  interaction.depth = 3,
  shrinkage = 0.01,
  cv.folds = 5,
  n.cores = NULL,
  verbose = FALSE
)
```




```
Numero ottimale di boosting iterations: 460
```

The provided line chart illustrates the training and validation loss against the number of trees (iterations). It can be clearly observed that the validation error reaches its minimum at the point indicated on the chart, representing the optimal number of boosting iterations to avoid overfitting.

```
summary(gbm_model, n.trees = best_iter)
```




```
    var   rel.inf
TG   TG 34.200436
BMI BMI 31.278439
BP   BP 11.231238
GC   GC  5.802120
HDL HDL  4.686531
age age  3.341381
LDL LDL  2.674631
TC   TC  2.645068
TCH TCH  2.469377
sex sex  1.670779
```

The bar chart displays the relative influence of each variable in the Gradient Boosting Machine (GBM) model. The results clearly indicate that **TG** and **BMI** are the two most influential predictors, with relative influence values of approximately 34.2% and 31.3%, respectively. This suggests that these two variables play a dominant role in the model's predictive capability. **BP** follows, though with substantially lower influence (around 11.2%), highlighting a considerable gap between the top predictors and the rest. Other variables such as **GC**, **HDL** and **age** contribute modestly, while features like **LDL**, **TC**, **TCH**, and **sex** exhibit very low relative influence, suggesting they have limited impact on the model's performance. Overall, the model appears to rely heavily on a small subset of predictors, with **TG** and **BMI** being the key drivers of predictions.

Comparison between models with CV

In this final section we compare the predictive performance of three different regression models - a pruned decision tree, a random forest and a gradient boosting machine - in predicting diabetes progression. The comparison is performed using a 5-fold cross-validation procedure: the dataset is divided into five parts and, for each fold, the models are trained on 80 per cent of the data and tested on the remaining 20 per cent. Each model is tuned within the training set (e.g. pruning of the decision tree, selection of the optimal number of trees for the random forest and boosting). The mean square error (MSE) is calculated on the test set for each model. Finally, the code reports the MSE values between the folds and calculates the mean MSE for each model, allowing a fair and unbiased comparison of their prediction accuracy.

```
Mean MSE:

Decision Tree: 4117.6

Random Forest: 3257.6

Boosting: 3095.75
```

Conclusion

In this study, three regression models — a cost-complexity pruned decision tree, a random forest, and a gradient boosting model — were developed to predict the progression of diabetes based on clinical predictors. Each model was tuned using internal validation strategies appropriate to its structure: cross-validation pruning for the decision tree, out-of-bag error minimization for the random forest, and cross-validation for boosting. Performance was evaluated through a 5-fold cross-validation procedure, re-optimizing model complexity at each fold. The decision tree achieved the highest average test MSE (4117.6), reflecting its lower flexibility and higher variance. The random forest improved prediction accuracy (3257.6) by reducing variance through aggregation, while boosting achieved the best performance (3095.75) by sequentially minimizing both bias and variance. Importantly, across all models, TG, BMI, and BP consistently emerged as the strongest predictors, reinforcing their central role in explaining disease progression. These results confirm that ensemble methods, and boosting in particular, offer superior predictive capabilities compared to simpler models, although they require more careful tuning and are less interpretable.