

Machine Learning for Computational Biology

Nikolay Oskolkov, Lund University, NBIS SciLifeLab, Sweden
AFI course, 16.02.2026

Session 4: Decision Tree-based Machine Learning



@NikolayOskolkov



github.com/NikolayOskolkov

Topics we'll cover in this session:

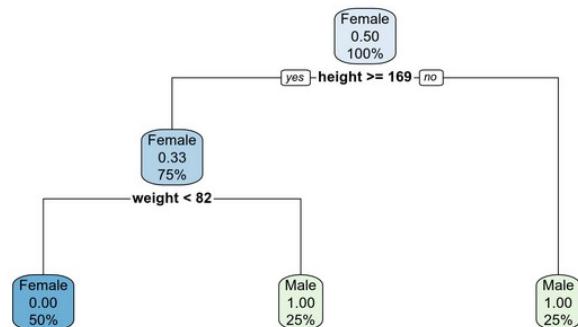
- 1) Decision tree cost function: Gini index
- 2) Computing optimal split via Gini index
- 3) Getting subset for further splitting
- 4) Code decision tree from scratch and validate model
- 5) Difference between decision tree and Random Forest

Decision tree from scratch in R: problem formulation

```
1 X<-data.frame(height=c(183,167,178,171),weight=c(78,73,85,67))
2 y<-as.factor(c("Female", "Male", "Male", "Female"))
3 data.frame(X, sex = y)
```

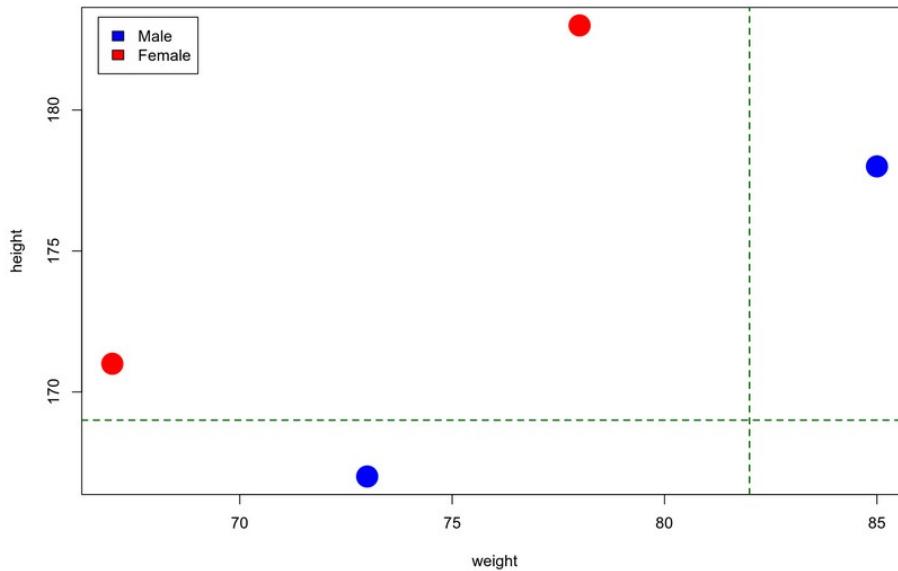
height	weight	sex
183	78	Female
167	73	Male
178	85	Male
171	67	Female

```
1 library("rpart"); library("rpart.plot")
2 fit<-rpart(y~height+weight,data=X,method="class",minsplit=1)
3 rpart.plot(fit)
```

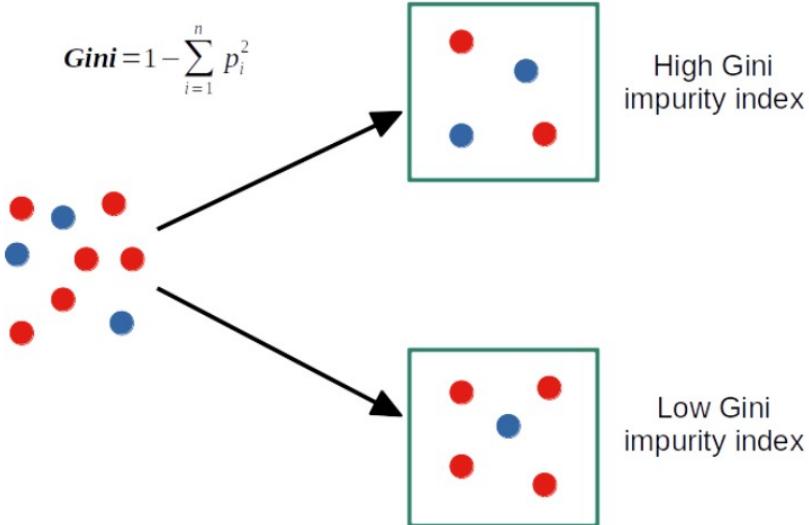


- Let us visualize what the classifier has learnt

```
1 color <- c("red", "blue", "blue", "red")
2 plot(height ~ weight, data = X, col = color, pch = 19, cex = 3)
3 legend("topleft",c("Male","Female"),fill=c("blue","red"),inset=.02)
4
5 abline(h = 169, lty = 2, col = "darkgreen", lwd = 1.5)
6 abline(v = 82, lty = 2, col = "darkgreen", lwd = 1.5)
```



Decision tree from scratch in R: Gini index and split



```
1 gini <- function(x)
2 {
3   return(1 - sum((table(x) / length(x))^2))
4 }
5 gini(c(1, 0, 1, 0))
```

```
[1] 0.5
```

```
1 gini(c(1, 1, 0, 1, 1))
```

```
[1] 0.32
```

```
1 get_best_split <- function(X, y)
2 {
3   mean_gini <- vector(); spl_vals <- vector(); spl_names <- vector()
4   for(j in colnames(X)) # for each variable in X data frame
5   {
6     spl <- vector() # vector of potential split candidates
7     sort_X <- X[order(X[, j]), ]; sort_y <- y[order(X[, j])] # sort by variable
8     for(i in 1:(dim(X)[1]-1)) # for each observation of variable in X data frame
9     {
10       spl[i] <- (sort_X[i, j] + sort_X[(i + 1), j]) / 2 # variable consecutive means
11       g1_y <- sort_y[sort_X[, j] > spl[i]] # take labels for group above split
12       g2_y <- sort_y[sort_X[, j] < spl[i]] # take labels for group below split
13       mean_gini <- append(mean_gini, (gini(g1_y) + gini(g2_y))/2) # two groups mean Gini
14       spl_vals <- append(spl_vals, spl[i])
15       spl_names <- append(spl_names, j)
16     }
17   }
18   min_spl_val <- spl_vals[mean_gini == min(mean_gini)][1] # get best split variable
19   min_spl_name <- spl_names[mean_gini == min(mean_gini)][1] # get best split value
20   sort_X <- X[order(X[, min_spl_name]), ] # sort X by best split variable
21   sort_y <- y[order(X[, min_spl_name])] # sort y by best split variable
22   g1_y <- sort_y[sort_X[, min_spl_name] > min_spl_val] # labels above best split
23   g2_y <- sort_y[sort_X[, min_spl_name] < min_spl_val] # labels below best split
24   if(gini(g1_y) == 0){sex <- paste0("Above: ", as.character(g1_y))}
25   else if(gini(g2_y) == 0){sex <- paste0("Below: ", as.character(g2_y))}
26
27   return(list(spl_name = min_spl_name, spl_value = min_spl_val, sex = sex))
28 }
29 get_best_split(X, y)
```

```
$spl_name
[1] "height"
$spl_value
[1] 169
$sex
[1] "Below: Male"
```

Decision tree from scratch in R: code implementation

- After we have found the best split, let us check what group we can split further: `get_new_data` function

```
1 get_new_data <- function(X, y)
2 {
3   spl_name <- get_best_split(X, y)$spl_name
4   spl_val <- get_best_split(X, y)$spl_value
5
6   # Sort X and y by the variable of the best split
7   sort_X <- X[order(X[, spl_name]), ]; sort_y <- y[order(X[, spl_name])]
8
9   # get X and y for the first group of samples above the best split value
10  g1_y <- sort_y[sort_X[, spl_name] > spl_val]
11  g1_X <- sort_X[sort_X[, spl_name] > spl_val,]
12
13  # get X and y for the second group of samples below the best split value
14  g2_y <- sort_y[sort_X[, spl_name] < spl_val]
15  g2_X <- sort_X[sort_X[, spl_name] < spl_val,]
16
17  # return new data (subset of X and y) for a group with Gini index > 0
18  if(gini(g1_y) > 0){return(list(new_X = g1_X, new_y = g1_y))}
19  else if(gini(g2_y) > 0){return(list(new_X = g2_X, new_y = g2_y))}
20  else{return(0)}
21 }
22 get_new_data(X, y)
```

```
$new_X
height weight
4    171     67
3    178     85
1    183     78
```

```
$new_y
[1] Female Male  Female
Levels: Female Male
```

- We can train a decision tree of `max_depth = 2`

```
1 decision_tree <- function(X, y, max_depth = 2)
2 {
3   new_X <- X; new_y <- y
4   df <- data.frame(matrix(ncol = 5, nrow = max_depth))
5   colnames(df) <- c("spl_num", "spl_name", "sign", "spl_val", "label")
6   for(i in 1:max_depth)
7   {
8     best_split_output <- get_best_split(new_X, new_y)
9     sex <- unlist(strsplit(best_split_output$sex,": "))
10    df[i, "spl_num"] <- i
11    df[i, "spl_name"] <- best_split_output$spl_name
12    df[i, "sign"] <- ifelse(sex[1] == "Below", "<", ">")
13    df[i, "spl_val"] <- best_split_output$spl_value
14    df[i, "label"] <- sex[2]
15
16    new_data_output <- get_new_data(new_X, new_y)
17    if(length(new_data_output) != 1)
18    {
19      new_X <- new_data_output$new_X
20      new_y <- new_data_output$new_y
21    }
22    else
23    {
24      print("All terminal nodes have perfect purity")
25      break
26    }
27  }
28  return(df)
29 }
30 decision_tree(X, y)
```

```
[1] "All terminal nodes have perfect purity"
```

spl_num	spl_name	sign	spl_val	label
1	height	<	169.0	Male
2	weight	>	81.5	Male

Decision tree from scratch in R: prediction

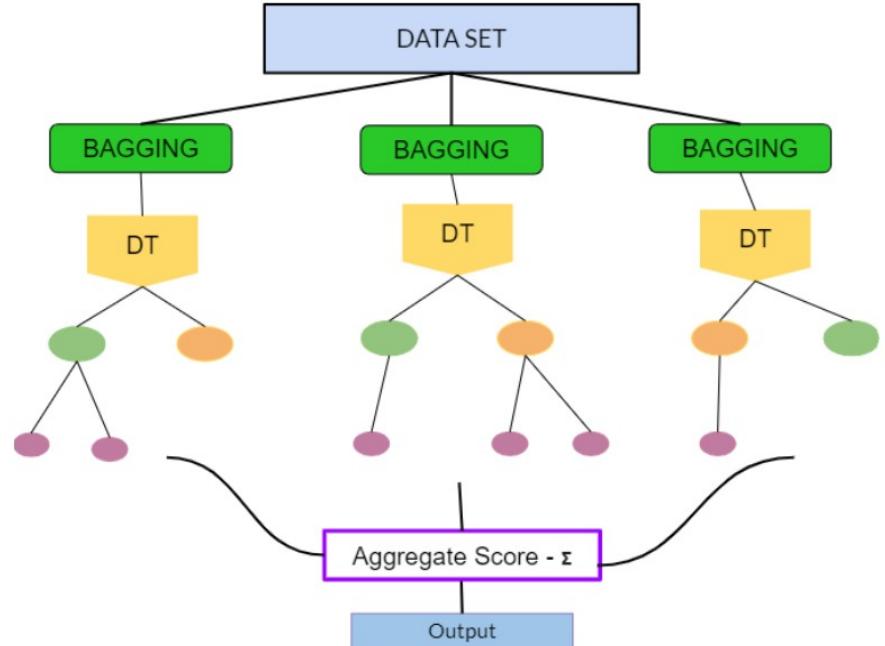
- Finally, after we have trained the decision tree, we can try to make predictions, and check whether we can reconstruct the labels of the data points

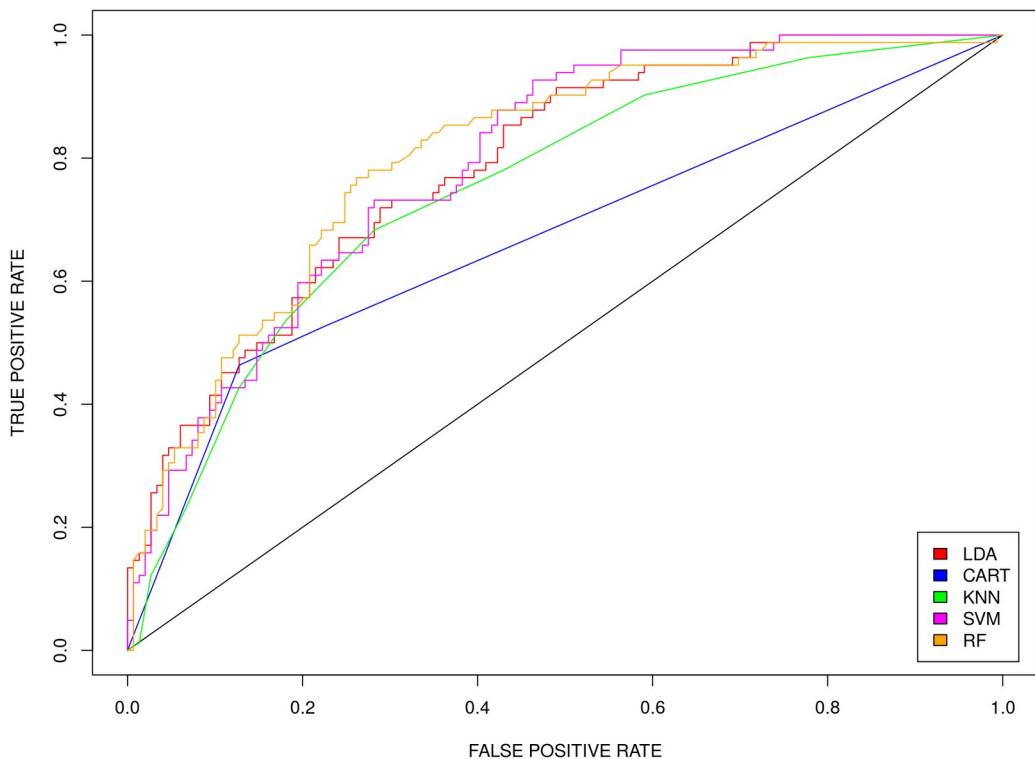
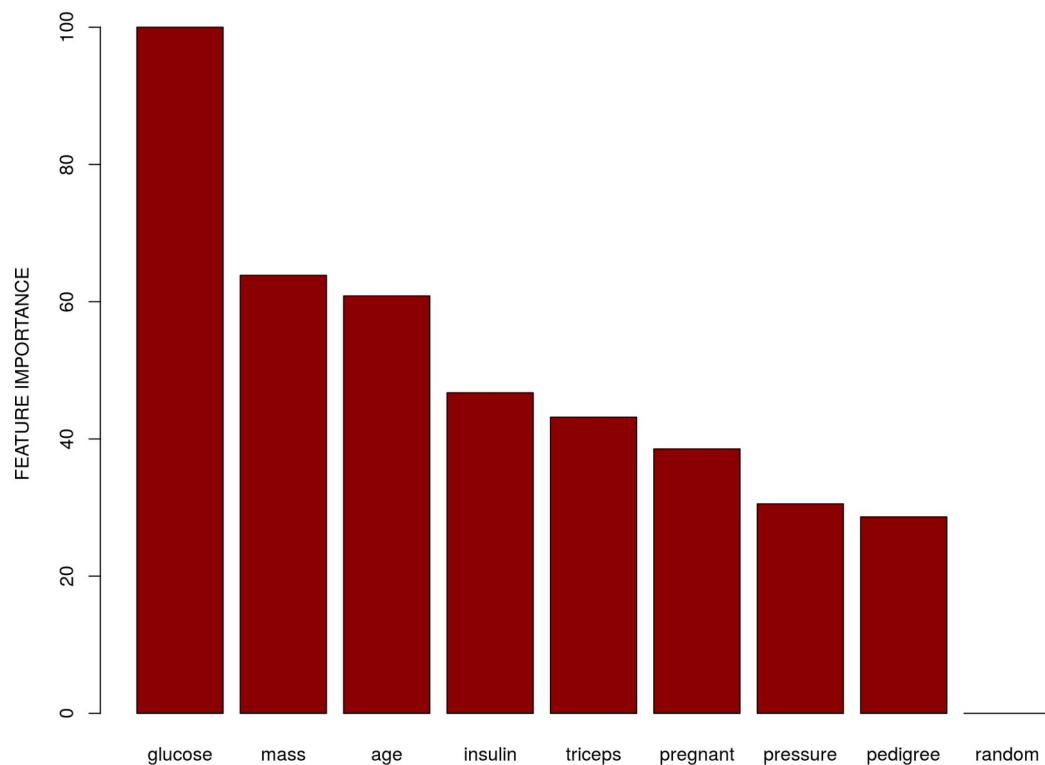
```
1 predict_decision_tree <- function(X, y)
2 {
3   # Train a decision tree
4   t <- decision_tree(X, y, max_depth = 2)
5
6   # Parse the output of decision tree and code it via if, else if and else
7   pred_labs <- vector()
8   for(i in 1:dim(X)[1])
9   {
10     if(eval(parse(text=paste0(X[i,t$spl_name[1]],t$sign[1],t$spl_val[1]))))
11     {
12       pred_labs[i] <- t$label[1]
13     }
14     else if(eval(parse(text=paste0(X[i,t$spl_name[2]],t$sign[2],t$spl_val[2]))))
15     {
16       pred_labs[i] <- t$label[2]
17     }
18     else{pred_labs[i] <- ifelse(t$label[2] == "Male", "Female", "Male")}
19   }
20
21   return(cbind(cbind(X, y), pred_labs))
22 }
23 predict_decision_tree(X, y)
```

```
[1] "All terminal nodes have perfect purity"
```

height	weight	y	pred_labs
183	78	Female	Female
167	73	Male	Male
178	85	Male	Male
171	67	Female	Female

- Random Forest has two key differences:
 - train multiple decision trees (bagging)
 - train trees on fractions of input features





Take home messages of the session:

- 1) Gini index is the internal decision tree cost function
- 2) Computing best split is done for each variable separately
- 3) Figuring out the subset to split further is done via Gini index
- 4) Ensemble learning is the main power of Random Forest which makes it quite different from regular decision trees



*Knut och Alice
Wallenbergs
Stiftelse*



**LUNDS
UNIVERSITET**