Verifica Tree Based Models

Guarino Renata

2024-06-01

# Analisi Comparativa dei Modelli Basati su Alberi per la Previsione del Cancro al Seno

# -----------------------------------------------  
# Il cancro al seno è una delle principali cause di mortalità tra le donne in tutto il mondo. Comprendere i fattori che influenzano la presenza o l'assenza di questa malattia è cruciale per lo sviluppo di strategie di prevenzione e trattamento efficaci.   
#   
# Nel seguente studio, verrà esaminato un dataset contenente informazioni cliniche e biologiche su pazienti con e senza cancro al seno. Attraverso l'analisi dei dati, sarà esplorata l'importanza di variabili come età, BMI, livelli di glucosio, insulina e altri parametri metabolici nel processo decisionale dei modelli.   
#   
#   
# Gli strumenti utilizzati includono gli alberi di classificazione (CART) e le random forest.In particolare, sarà valutata la capacità predittiva dei modelli, saranno identificate le variabili più influenti e sarà analizzata la loro relazione con la presenza del cancro al seno.  
# L'obiettivo di questo studio è fornire una migliore comprensione dei fattori che possono contribuire allo sviluppo del cancro al seno e fornire informazioni utili per la prevenzione e il trattamento della malattia.  
  
# -----------------------------------------------  
  
# Carica pacchetti necessari  
library(readr) # Per la lettura dei dati  
library(dplyr) # Per la manipolazione dei dati  
library(caTools) # Per la suddivisione dei dati  
library(rpart) # Per la costruzione di alberi di classificazione  
library(rpart.plot) # Per migliorare la visualizzazione degli alberi di classificazione  
library(pROC) # Per la creazione di curve ROC  
library(caret) # Per la validazione incrociata e la valutazione dei modelli  
library(ROSE) # Per il bilanciamento dei dati  
library(tree) # Per la costruzione di alberi di classificazione  
library(randomForest) # Per la costruzione di modelli di random forest  
library(ggplot2) # Per la creazione di grafici  
library(partykit) # Per la creazione di altri grafici  
  
  
# Visualizza il dataset  
# Leggi il file CSV e assegnalo alla variabile   
BK <- read\_csv("dataR2.csv")  
  
head(BK) # Visualizza le prime righe del dataset

## # A tibble: 6 × 10  
## Age BMI Glucose Insulin HOMA Leptin Adiponectin Resistin MCP.1  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 48 23.5 70 2.71 0.467 8.81 9.70 8.00 417.  
## 2 83 20.7 92 3.12 0.707 8.84 5.43 4.06 469.  
## 3 82 23.1 91 4.50 1.01 17.9 22.4 9.28 555.  
## 4 68 21.4 77 3.23 0.613 9.88 7.17 12.8 928.  
## 5 86 21.1 92 3.55 0.805 6.70 4.82 10.6 774.  
## 6 49 22.9 92 3.23 0.732 6.83 13.7 10.3 530.  
## # ℹ 1 more variable: Classification <dbl>

str(BK) # Visualizza la struttura del dataframe

## spc\_tbl\_ [116 × 10] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ Age : num [1:116] 48 83 82 68 86 49 89 76 73 75 ...  
## $ BMI : num [1:116] 23.5 20.7 23.1 21.4 21.1 ...  
## $ Glucose : num [1:116] 70 92 91 77 92 92 77 118 97 83 ...  
## $ Insulin : num [1:116] 2.71 3.12 4.5 3.23 3.55 ...  
## $ HOMA : num [1:116] 0.467 0.707 1.01 0.613 0.805 ...  
## $ Leptin : num [1:116] 8.81 8.84 17.94 9.88 6.7 ...  
## $ Adiponectin : num [1:116] 9.7 5.43 22.43 7.17 4.82 ...  
## $ Resistin : num [1:116] 8 4.06 9.28 12.77 10.58 ...  
## $ MCP.1 : num [1:116] 417 469 555 928 774 ...  
## $ Classification: num [1:116] 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Age = col\_double(),  
## .. BMI = col\_double(),  
## .. Glucose = col\_double(),  
## .. Insulin = col\_double(),  
## .. HOMA = col\_double(),  
## .. Leptin = col\_double(),  
## .. Adiponectin = col\_double(),  
## .. Resistin = col\_double(),  
## .. MCP.1 = col\_double(),  
## .. Classification = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

tail(BK) # Visualizza le ultime sei righe del dataset

## # A tibble: 6 × 10  
## Age BMI Glucose Insulin HOMA Leptin Adiponectin Resistin MCP.1  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 54 36.0 119 11.9 3.50 89.3 8.01 5.06 218.   
## 2 45 26.8 92 3.33 0.756 54.7 12.1 11.0 268.   
## 3 62 26.8 100 4.53 1.12 12.4 21.4 7.32 330.   
## 4 65 32.0 97 5.73 1.37 61.5 22.5 10.3 314.   
## 5 72 25.6 82 2.82 0.570 25.0 33.8 3.27 392.   
## 6 86 27.2 138 19.9 6.78 90.3 14.1 4.35 90.1  
## # ℹ 1 more variable: Classification <dbl>

# Numero di livelli nella variabile "Classification"  
num\_levels <- length(unique(BK$Classification))  
print(num\_levels)

## [1] 2

# Visualizza i valori univoci della variabile "Classification"  
unique\_values <- unique(BK$Classification)  
print(unique\_values)

## [1] 1 2

print(BK$Classification)

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [38] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [75] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2  
## [112] 2 2 2 2 2

# Ottieni il numero di variabili  
num\_variables <- ncol(BK)  
cat("Numero di variabili:", num\_variables, "\n")

## Numero di variabili: 10

# Ottieni i tipi di variabili  
variable\_types <- sapply(BK, class)  
print(variable\_types)

## Age BMI Glucose Insulin HOMA   
## "numeric" "numeric" "numeric" "numeric" "numeric"   
## Leptin Adiponectin Resistin MCP.1 Classification   
## "numeric" "numeric" "numeric" "numeric" "numeric"

# Il Datase è costitutito da 10 variabili numeriche:  
  
# Age: età;  
# BMI: indice di massa corporea;  
# Glucose: livelli di glucosio nel sangue;  
# Insulin: quantità di insulina somministrata;  
# Homa (Homeostasis Model Assessment): metodo di calcolo dell'insulino-resistenza (HOMA-IR (Insulin Resistance) = [Insulina a digiuno (μU/mL) \* Glicemia a digiuno (mmol/L)] / 22.5);  
# Leptin: ormone prodotto principalmente dalle cellule adipose (aumenta il bilancio energetico inibendo la fame);  
# Adiponectin: proteina prodotta dalle cellule adipose che modula una serie di processi metabolici, tra cui la regolazione del glucosio e la degradazione degli acidi grassi;  
# Resistin: ormone prodotto dalle cellule adipose e dai macrofagi (può indurre ad una insulino-resistenza)  
# MCP.1 (Monocyte Chemoattractant Protein-1): svolge un ruolo cruciale nel reclutamento e nell'attivazione dei monociti, dei macrofagi e delle cellule T al sito di infiammazione  
# Classification: 1 - Sotto controllo medico, 2 – Paziente

# Link dataset: https://archive.ics.uci.edu/dataset/451/breast+cancer+coimbra  
  
  
# Missing Data  
  
# Controlla se ci sono valori mancanti nel dataframe "Diabetes"  
missing\_values <- colSums(is.na(BK))  
  
# Visualizza il numero di valori mancanti per ogni colonna  
print(missing\_values)

## Age BMI Glucose Insulin HOMA   
## 0 0 0 0 0   
## Leptin Adiponectin Resistin MCP.1 Classification   
## 0 0 0 0 0

# Non ci sono valori mancanti sul dataset  
  
  
  
# -----------------------------------------------  
# Calcolo delle percentuali di "Yes" e "No"  
# -----------------------------------------------  
#  
# Le percentuali di "Yes" e "No" nella variabile target vengono calcolate per valutare l'equilibrio del dataset dopo il bilanciamento delle classi.  
#  
# Questo passaggio è importante per verificare se l'oversampling ha prodotto una distribuzione bilanciata delle classi nel dataset.  
#  
# Le percentuali vengono calcolate dividendo il numero di osservazioni corrispondenti alla classe "Yes" o "No" per il numero totale di osservazioni nel dataset bilanciato e moltiplicando il risultato per 100.  
#  
# Le percentuali calcolate vengono quindi stampate utilizzando la funzione `cat` per fornire un resoconto chiaro della distribuzione delle classi nel dataset.  
#  
# -----------------------------------------------  
  
# Calcola le percentuali di "Yes" e "No" nella variabile target  
percent\_no <- mean(BK$Classification == "1") \* 100  
percent\_yes <- mean(BK$Classification == "2") \* 100  
  
# Stampare le percentuali  
cat("Percentuale di '1':", percent\_yes, "%\n")

## Percentuale di '1': 55.17241 %

cat("Percentuale di '2':", percent\_no, "%\n")

## Percentuale di '2': 44.82759 %

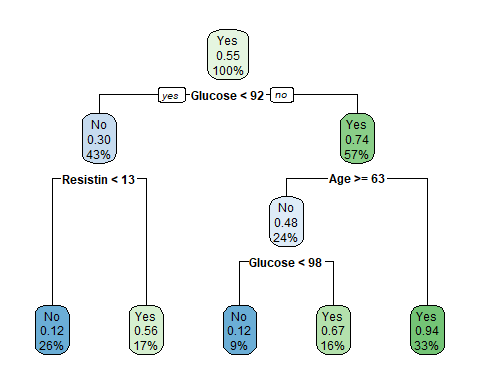
# Trasformazione della variabile target in factor  
  
# Etichetta i valori "1" e "2" come "No" e "Yes"  
BK$Classification <- factor(BK$Classification, levels = c("1", "2"), labels = c("No", "Yes"))  
  
# Verifica della trasformazione  
str(BK$Classification)

## Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...

# Verifica della distribuzione delle classi dopo la trasformazione  
table(BK$Classification)

##   
## No Yes   
## 52 64

# ---------------------- PROCESSING ------------------------------------  
# Imposta il seme del generatore di numeri casuali per renderlo riproducibile  
set.seed(123)  
  
# Suddivide il dataset in training set e testing set  
train\_index <- createDataPartition(BK$Classification, p = 0.8, list = FALSE, times = 1)  
train <- BK[train\_index, ]  
test <- BK[-train\_index, ]  
  
# albero di prova  
Default\_Tree <- rpart(Classification ~ ., data = train, method = "class")  
rpart.plot(Default\_Tree)  
  
  
  
######## CART ##############  
  
# Creo il primo albero con rpart  
  
# Crea un albero con il dataset training inserendo cp = 0.0001  
set.seed(123)  
First\_Tree <- rpart(Classification ~ ., data = train, method = "class", cp = 0.0001)  
rpart.plot(First\_Tree)



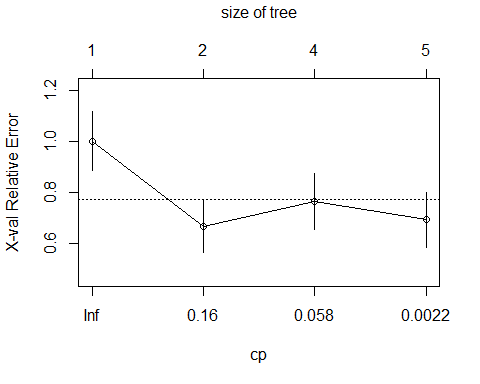
# Analizzando il 100% delle osservazioni, il primo albero evidenzia che livelli elevati delle prime tre variabili significative - Glucose, Age, Resistin - influenzano l'incidenza della malattia nei pazienti affetti da carcinoma mammario.  
  
summary(First\_Tree)

## Call:  
## rpart(formula = Classification ~ ., data = train, method = "class",   
## cp = 1e-04)  
## n= 94   
##   
## CP nsplit rel error xerror xstd  
## 1 0.38095238 0 1.0000000 1.0000000 0.1147660  
## 2 0.07142857 1 0.6190476 0.6666667 0.1055693  
## 3 0.04761905 3 0.4761905 0.7619048 0.1093850  
## 4 0.00010000 4 0.4285714 0.6904762 0.1066209  
##   
## Variable importance  
## Glucose Age HOMA Insulin Resistin Leptin   
## 27 18 13 11 9 8   
## MCP.1 BMI Adiponectin   
## 5 5 4   
##   
## Node number 1: 94 observations, complexity param=0.3809524  
## predicted class=Yes expected loss=0.4468085 P(node) =1  
## class counts: 42 52  
## probabilities: 0.447 0.553   
## left son=2 (40 obs) right son=3 (54 obs)  
## Primary splits:  
## Glucose < 91.5 to the left, improve=8.927344, (0 missing)  
## Age < 39 to the left, improve=6.844153, (0 missing)  
## HOMA < 2.169985 to the left, improve=5.601717, (0 missing)  
## Insulin < 10.285 to the left, improve=4.312241, (0 missing)  
## BMI < 33.67744 to the right, improve=3.778799, (0 missing)  
## Surrogate splits:  
## HOMA < 1.094619 to the left, agree=0.702, adj=0.300, (0 split)  
## Age < 39 to the left, agree=0.691, adj=0.275, (0 split)  
## Insulin < 5.403 to the left, agree=0.660, adj=0.200, (0 split)  
## Adiponectin < 22.12789 to the right, agree=0.628, adj=0.125, (0 split)  
## MCP.1 < 526.405 to the right, agree=0.628, adj=0.125, (0 split)  
##   
## Node number 2: 40 observations, complexity param=0.04761905  
## predicted class=No expected loss=0.3 P(node) =0.4255319  
## class counts: 28 12  
## probabilities: 0.700 0.300   
## left son=4 (24 obs) right son=5 (16 obs)  
## Primary splits:  
## Resistin < 13.24805 to the left, improve=3.675000, (0 missing)  
## Age < 44 to the left, improve=3.466667, (0 missing)  
## Insulin < 3.4445 to the right, improve=2.912554, (0 missing)  
## BMI < 31.01992 to the right, improve=2.400000, (0 missing)  
## HOMA < 0.809264 to the right, improve=1.828213, (0 missing)  
## Surrogate splits:  
## BMI < 27.90689 to the left, agree=0.800, adj=0.500, (0 split)  
## Insulin < 6.3485 to the left, agree=0.775, adj=0.438, (0 split)  
## HOMA < 1.416853 to the left, agree=0.775, adj=0.438, (0 split)  
## Leptin < 29.01205 to the left, agree=0.750, adj=0.375, (0 split)  
## Glucose < 86.5 to the left, agree=0.700, adj=0.250, (0 split)  
##   
## Node number 3: 54 observations, complexity param=0.07142857  
## predicted class=Yes expected loss=0.2592593 P(node) =0.5744681  
## class counts: 14 40  
## probabilities: 0.259 0.741   
## left son=6 (23 obs) right son=7 (31 obs)  
## Primary splits:  
## Age < 63 to the right, improve=5.520544, (0 missing)  
## BMI < 32.48096 to the right, improve=2.849832, (0 missing)  
## MCP.1 < 297.372 to the left, improve=2.695628, (0 missing)  
## Glucose < 118.5 to the left, improve=1.649832, (0 missing)  
## Leptin < 9.6597 to the left, improve=1.567488, (0 missing)  
## Surrogate splits:  
## Leptin < 47.14355 to the right, agree=0.685, adj=0.261, (0 split)  
## MCP.1 < 317.9845 to the left, agree=0.685, adj=0.261, (0 split)  
## Adiponectin < 5.470395 to the left, agree=0.630, adj=0.130, (0 split)  
## Insulin < 9.008 to the left, agree=0.611, adj=0.087, (0 split)  
## Resistin < 4.558425 to the left, agree=0.611, adj=0.087, (0 split)  
##   
## Node number 4: 24 observations  
## predicted class=No expected loss=0.125 P(node) =0.2553191  
## class counts: 21 3  
## probabilities: 0.875 0.125   
##   
## Node number 5: 16 observations  
## predicted class=Yes expected loss=0.4375 P(node) =0.1702128  
## class counts: 7 9  
## probabilities: 0.438 0.562   
##   
## Node number 6: 23 observations, complexity param=0.07142857  
## predicted class=No expected loss=0.4782609 P(node) =0.2446809  
## class counts: 12 11  
## probabilities: 0.522 0.478   
## left son=12 (8 obs) right son=13 (15 obs)  
## Primary splits:  
## Glucose < 97.5 to the left, improve=3.061594, (0 missing)  
## MCP.1 < 297.372 to the left, improve=2.614625, (0 missing)  
## Resistin < 12.7652 to the left, improve=1.811594, (0 missing)  
## Insulin < 5.673 to the left, improve=1.278261, (0 missing)  
## HOMA < 2.222398 to the left, improve=1.049689, (0 missing)  
## Surrogate splits:  
## HOMA < 1.374439 to the left, agree=0.870, adj=0.625, (0 split)  
## Insulin < 5.533 to the left, agree=0.826, adj=0.500, (0 split)  
## Leptin < 8.7747 to the left, agree=0.739, adj=0.250, (0 split)  
## Age < 65.5 to the left, agree=0.696, adj=0.125, (0 split)  
## BMI < 22.11111 to the left, agree=0.696, adj=0.125, (0 split)  
##   
## Node number 7: 31 observations  
## predicted class=Yes expected loss=0.06451613 P(node) =0.3297872  
## class counts: 2 29  
## probabilities: 0.065 0.935   
##   
## Node number 12: 8 observations  
## predicted class=No expected loss=0.125 P(node) =0.08510638  
## class counts: 7 1  
## probabilities: 0.875 0.125   
##   
## Node number 13: 15 observations  
## predicted class=Yes expected loss=0.3333333 P(node) =0.1595745  
## class counts: 5 10  
## probabilities: 0.333 0.667

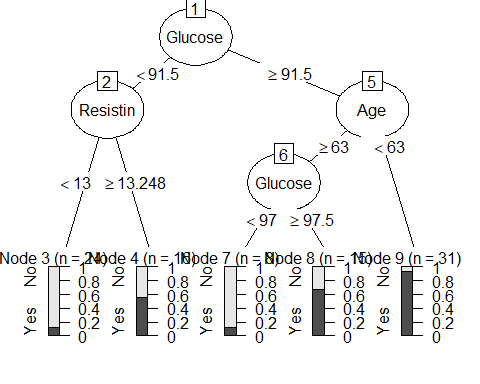
printcp(First\_Tree)

##   
## Classification tree:  
## rpart(formula = Classification ~ ., data = train, method = "class",   
## cp = 1e-04)  
##   
## Variables actually used in tree construction:  
## [1] Age Glucose Resistin  
##   
## Root node error: 42/94 = 0.44681  
##   
## n= 94   
##   
## CP nsplit rel error xerror xstd  
## 1 0.380952 0 1.00000 1.00000 0.11477  
## 2 0.071429 1 0.61905 0.66667 0.10557  
## 3 0.047619 3 0.47619 0.76190 0.10938  
## 4 0.000100 4 0.42857 0.69048 0.10662

plotcp(First\_Tree)



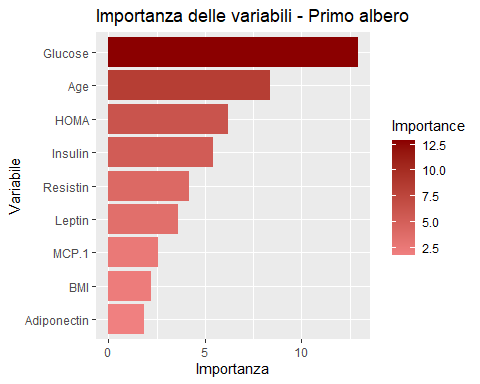
# Altro plot per la classificazione del primo albero  
plot(as.party(First\_Tree), cex=0.2)



# È evidente sin da subito che la risposta è influenzata da livelli elevati di Glucosio, seguiti da una fascia d'età più giovane.  
  
# Visualizza le variabili importanti  
First\_Tree$variable.importance

## Glucose Age HOMA Insulin Resistin Leptin   
## 12.907689 8.358263 6.199512 5.404126 4.155047 3.583666   
## MCP.1 BMI Adiponectin   
## 2.556060 2.220199 1.835989

# Estrai l'importanza delle variabili per il primo albero  
importance\_values\_first <- First\_Tree$variable.importance  
names\_first\_tree <- names(importance\_values\_first)  
var\_importance\_df\_first <- data.frame(Variable = names\_first\_tree, Importance = importance\_values\_first)  
  
# Grafico dell'importanza delle variabili per il primo albero  
ggplot(var\_importance\_df\_first, aes(x = reorder(Variable, Importance), y = Importance, fill = Importance)) +  
 geom\_bar(stat = "identity") +  
 coord\_flip() +  
 xlab("Variabile") +  
 ylab("Importanza") +  
 ggtitle("Importanza delle variabili - Primo albero") +  
 scale\_fill\_gradient(low = "lightcoral", high = "darkred")



# Glucosio, età e livelli di HOMA emergono come le variabili più significative.  
  
# Prevedi le classi utilizzando il primo albero  
Predicted\_First\_Tree <- predict(First\_Tree, newdata = test, type = 'class')  
test$Predicted\_First\_Tree <- Predicted\_First\_Tree  
  
# Valuta sul test set  
table(Predicted\_First\_Tree, test$Classification)

##   
## Predicted\_First\_Tree No Yes  
## No 6 1  
## Yes 4 11

# Calcola il tasso di errore di classificazione sul test set  
missclassification\_error\_rate <- mean(Predicted\_First\_Tree != test$Classification)  
print(missclassification\_error\_rate)

## [1] 0.2272727

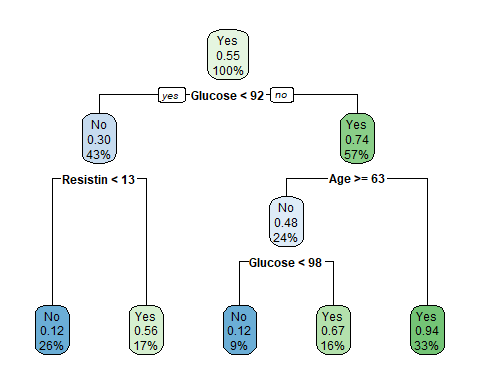
# Crea una matrice di confusione anche attraverso il quale, si può visualizzare l'accuratezza del modello sul test set.  
confusionMatrix(Predicted\_First\_Tree, test$Classification)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6 1  
## Yes 4 11  
##   
## Accuracy : 0.7727   
## 95% CI : (0.5463, 0.9218)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.02455   
##   
## Kappa : 0.5299   
##   
## Mcnemar's Test P-Value : 0.37109   
##   
## Sensitivity : 0.6000   
## Specificity : 0.9167   
## Pos Pred Value : 0.8571   
## Neg Pred Value : 0.7333   
## Prevalence : 0.4545   
## Detection Rate : 0.2727   
## Detection Prevalence : 0.3182   
## Balanced Accuracy : 0.7583   
##   
## 'Positive' Class : No   
##

# L'albero è stato costruito utilizzando il set di dati di addestramento, con un parametro di complessità (cp) pari a 0.0001. La complessità controlla il numero di nodi e la profondità dell'albero. Una complessità più bassa può portare a un albero più complesso, mentre una complessità più alta può portare a un albero più semplice.  
  
# Variabili importanti:  
# Le variabili Glucose, Age, HOMA sono state identificate come le più importanti per la classificazione. Questo significa che sono le variabili che meglio distinguono le classi di output nel dataset.  
  
# Tabella di confusione:  
# La tabella di confusione mostra che su 22 casi, il modello ha correttamente classificato 17 casi. Questo si traduce in un'accuratezza del 77%, indicando che il modello è in grado di predire correttamente la classe di appartenenza nella maggior parte dei casi.  
  
# Accuratezza:  
# L'accuratezza del modello indica la percentuale di casi correttamente classificati rispetto al totale dei casi. Un'accuratezza del 77% indica che il modello ha predetto correttamente la classe di appartenenza nel 77% dei casi.  
  
# Tasso di errore di classificazione:  
# Il tasso di errore di classificazione rappresenta la percentuale di casi predetti in modo errato rispetto al totale dei casi. Un tasso di errore del 22,72% indica che il modello ha predetto erroneamente la classe di appartenenza nel 22,72% dei casi.  
  
# Valore Kappa:  
# Il valore Kappa è una misura della concordanza tra le predizioni del modello e i veri valori. Un valore Kappa del 53% indica una buona concordanza tra le predizioni del modello e i veri valori.  
  
# Sensibilità e Specificità:  
# La sensibilità rappresenta la capacità del modello di identificare correttamente i veri positivi, mentre la specificità rappresenta la capacità del modello di identificare correttamente i veri negativi. Una sensibilità del 60% indica che il modello identifica correttamente i pazienti affetti dalla malattia nel 60% dei casi, mentre una specificità del 92% indica che il modello identifica correttamente i pazienti non affetti dalla malattia nel 92% dei casi.  
  
# Complessivamente, i risultati indicano che l'albero di classificazione ha una buona capacità di predire la classe di appartenenza.  
  
  
  
  
# Secondo alberto con rpart  
  
"Poiché nel primo albero ho utilizzato un valore di cp pari a 0.01, ho deciso di esplorare ulteriormente la complessità dell'albero riducendo il valore di cp a 0 nel secondo albero.   
In questo modo, sto esaminando un modello con una maggiore complessità, consentendo all'albero di adattarsi meglio ai dati di addestramento senza la necessità di potatura. Questa scelta mi permette di esplorare diverse configurazioni dell'albero e valutare se un modello più complesso possa portare a una migliore capacità predittiva senza dover ricorrere alla potatura."

## [1] "Poiché nel primo albero ho utilizzato un valore di cp pari a 0.01, ho deciso di esplorare ulteriormente la complessità dell'albero riducendo il valore di cp a 0 nel secondo albero. \nIn questo modo, sto esaminando un modello con una maggiore complessità, consentendo all'albero di adattarsi meglio ai dati di addestramento senza la necessità di potatura. Questa scelta mi permette di esplorare diverse configurazioni dell'albero e valutare se un modello più complesso possa portare a una migliore capacità predittiva senza dover ricorrere alla potatura."

# Creo il secondo albero con cp = 0  
set.seed(123)  
Second\_Tree <- rpart(Classification ~ ., data = train, method = "class", cp = 0)  
rpart.plot(Second\_Tree)



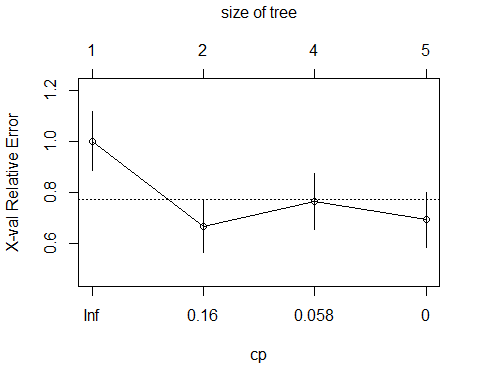
# Confermando quanto discusso in precedenza, il secondo grafico sottolinea l'importanza del glucosio nel distinguere tra pazienti affetti dalla condizione e soggetti sotto controllo medico  
  
summary(Second\_Tree)

## Call:  
## rpart(formula = Classification ~ ., data = train, method = "class",   
## cp = 0)  
## n= 94   
##   
## CP nsplit rel error xerror xstd  
## 1 0.38095238 0 1.0000000 1.0000000 0.1147660  
## 2 0.07142857 1 0.6190476 0.6666667 0.1055693  
## 3 0.04761905 3 0.4761905 0.7619048 0.1093850  
## 4 0.00000000 4 0.4285714 0.6904762 0.1066209  
##   
## Variable importance  
## Glucose Age HOMA Insulin Resistin Leptin   
## 27 18 13 11 9 8   
## MCP.1 BMI Adiponectin   
## 5 5 4   
##   
## Node number 1: 94 observations, complexity param=0.3809524  
## predicted class=Yes expected loss=0.4468085 P(node) =1  
## class counts: 42 52  
## probabilities: 0.447 0.553   
## left son=2 (40 obs) right son=3 (54 obs)  
## Primary splits:  
## Glucose < 91.5 to the left, improve=8.927344, (0 missing)  
## Age < 39 to the left, improve=6.844153, (0 missing)  
## HOMA < 2.169985 to the left, improve=5.601717, (0 missing)  
## Insulin < 10.285 to the left, improve=4.312241, (0 missing)  
## BMI < 33.67744 to the right, improve=3.778799, (0 missing)  
## Surrogate splits:  
## HOMA < 1.094619 to the left, agree=0.702, adj=0.300, (0 split)  
## Age < 39 to the left, agree=0.691, adj=0.275, (0 split)  
## Insulin < 5.403 to the left, agree=0.660, adj=0.200, (0 split)  
## Adiponectin < 22.12789 to the right, agree=0.628, adj=0.125, (0 split)  
## MCP.1 < 526.405 to the right, agree=0.628, adj=0.125, (0 split)  
##   
## Node number 2: 40 observations, complexity param=0.04761905  
## predicted class=No expected loss=0.3 P(node) =0.4255319  
## class counts: 28 12  
## probabilities: 0.700 0.300   
## left son=4 (24 obs) right son=5 (16 obs)  
## Primary splits:  
## Resistin < 13.24805 to the left, improve=3.675000, (0 missing)  
## Age < 44 to the left, improve=3.466667, (0 missing)  
## Insulin < 3.4445 to the right, improve=2.912554, (0 missing)  
## BMI < 31.01992 to the right, improve=2.400000, (0 missing)  
## HOMA < 0.809264 to the right, improve=1.828213, (0 missing)  
## Surrogate splits:  
## BMI < 27.90689 to the left, agree=0.800, adj=0.500, (0 split)  
## Insulin < 6.3485 to the left, agree=0.775, adj=0.438, (0 split)  
## HOMA < 1.416853 to the left, agree=0.775, adj=0.438, (0 split)  
## Leptin < 29.01205 to the left, agree=0.750, adj=0.375, (0 split)  
## Glucose < 86.5 to the left, agree=0.700, adj=0.250, (0 split)  
##   
## Node number 3: 54 observations, complexity param=0.07142857  
## predicted class=Yes expected loss=0.2592593 P(node) =0.5744681  
## class counts: 14 40  
## probabilities: 0.259 0.741   
## left son=6 (23 obs) right son=7 (31 obs)  
## Primary splits:  
## Age < 63 to the right, improve=5.520544, (0 missing)  
## BMI < 32.48096 to the right, improve=2.849832, (0 missing)  
## MCP.1 < 297.372 to the left, improve=2.695628, (0 missing)  
## Glucose < 118.5 to the left, improve=1.649832, (0 missing)  
## Leptin < 9.6597 to the left, improve=1.567488, (0 missing)  
## Surrogate splits:  
## Leptin < 47.14355 to the right, agree=0.685, adj=0.261, (0 split)  
## MCP.1 < 317.9845 to the left, agree=0.685, adj=0.261, (0 split)  
## Adiponectin < 5.470395 to the left, agree=0.630, adj=0.130, (0 split)  
## Insulin < 9.008 to the left, agree=0.611, adj=0.087, (0 split)  
## Resistin < 4.558425 to the left, agree=0.611, adj=0.087, (0 split)  
##   
## Node number 4: 24 observations  
## predicted class=No expected loss=0.125 P(node) =0.2553191  
## class counts: 21 3  
## probabilities: 0.875 0.125   
##   
## Node number 5: 16 observations  
## predicted class=Yes expected loss=0.4375 P(node) =0.1702128  
## class counts: 7 9  
## probabilities: 0.438 0.562   
##   
## Node number 6: 23 observations, complexity param=0.07142857  
## predicted class=No expected loss=0.4782609 P(node) =0.2446809  
## class counts: 12 11  
## probabilities: 0.522 0.478   
## left son=12 (8 obs) right son=13 (15 obs)  
## Primary splits:  
## Glucose < 97.5 to the left, improve=3.061594, (0 missing)  
## MCP.1 < 297.372 to the left, improve=2.614625, (0 missing)  
## Resistin < 12.7652 to the left, improve=1.811594, (0 missing)  
## Insulin < 5.673 to the left, improve=1.278261, (0 missing)  
## HOMA < 2.222398 to the left, improve=1.049689, (0 missing)  
## Surrogate splits:  
## HOMA < 1.374439 to the left, agree=0.870, adj=0.625, (0 split)  
## Insulin < 5.533 to the left, agree=0.826, adj=0.500, (0 split)  
## Leptin < 8.7747 to the left, agree=0.739, adj=0.250, (0 split)  
## Age < 65.5 to the left, agree=0.696, adj=0.125, (0 split)  
## BMI < 22.11111 to the left, agree=0.696, adj=0.125, (0 split)  
##   
## Node number 7: 31 observations  
## predicted class=Yes expected loss=0.06451613 P(node) =0.3297872  
## class counts: 2 29  
## probabilities: 0.065 0.935   
##   
## Node number 12: 8 observations  
## predicted class=No expected loss=0.125 P(node) =0.08510638  
## class counts: 7 1  
## probabilities: 0.875 0.125   
##   
## Node number 13: 15 observations  
## predicted class=Yes expected loss=0.3333333 P(node) =0.1595745  
## class counts: 5 10  
## probabilities: 0.333 0.667

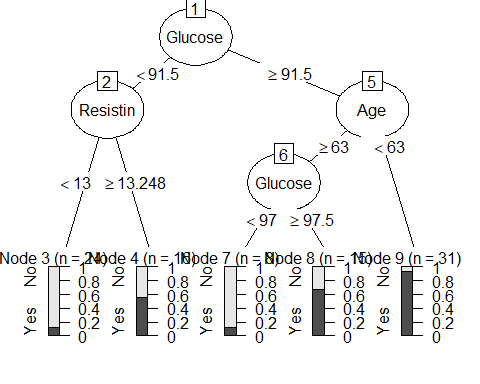
printcp(Second\_Tree)

##   
## Classification tree:  
## rpart(formula = Classification ~ ., data = train, method = "class",   
## cp = 0)  
##   
## Variables actually used in tree construction:  
## [1] Age Glucose Resistin  
##   
## Root node error: 42/94 = 0.44681  
##   
## n= 94   
##   
## CP nsplit rel error xerror xstd  
## 1 0.380952 0 1.00000 1.00000 0.11477  
## 2 0.071429 1 0.61905 0.66667 0.10557  
## 3 0.047619 3 0.47619 0.76190 0.10938  
## 4 0.000000 4 0.42857 0.69048 0.10662

plotcp(Second\_Tree)



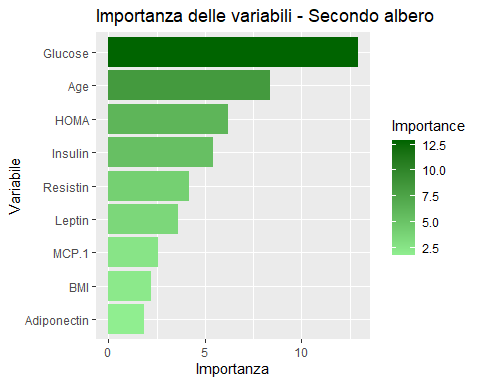
# Altro plot per la classificazione del secondo albero  
plot(as.party(Second\_Tree), cex=0.2)



# Visualizza le variabili importanti  
Second\_Tree$variable.importance

## Glucose Age HOMA Insulin Resistin Leptin   
## 12.907689 8.358263 6.199512 5.404126 4.155047 3.583666   
## MCP.1 BMI Adiponectin   
## 2.556060 2.220199 1.835989

# Estrai l'importanza delle variabili per il secondo albero  
importance\_values\_second <- Second\_Tree$variable.importance  
names\_second\_tree <- names(importance\_values\_second)  
var\_importance\_df\_second <- data.frame(Variable = names\_second\_tree, Importance = importance\_values\_second)  
  
# Grafico dell'importanza delle variabili per il secondo albero  
ggplot(var\_importance\_df\_second, aes(x = reorder(Variable, Importance), y = Importance, fill = Importance)) +  
 geom\_bar(stat = "identity") +  
 coord\_flip() +  
 xlab("Variabile") +  
 ylab("Importanza") +  
 ggtitle("Importanza delle variabili - Secondo albero") +  
 scale\_fill\_gradient(low = "lightgreen", high = "darkgreen")



Predicted\_Second\_Tree <- predict(Second\_Tree, newdata = test, type = 'class')  
test$Predicted\_Second\_Tree <- Predicted\_Second\_Tree  
  
# Valutazione sul test set  
table(Predicted\_Second\_Tree, test$Classification)

##   
## Predicted\_Second\_Tree No Yes  
## No 6 1  
## Yes 4 11

# Calcola il tasso di errore di classificazione sul test set  
missclassification\_error\_rate\_second <- mean(Predicted\_Second\_Tree != test$Classification)  
print(missclassification\_error\_rate\_second)

## [1] 0.2272727

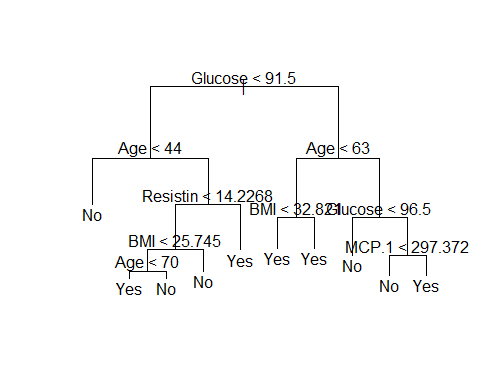
# Crea una matrice di confusione anche attraverso il quale, si può visualizzare l'accuratezza del modello sul test set.  
confusionMatrix(Predicted\_Second\_Tree, test$Classification)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6 1  
## Yes 4 11  
##   
## Accuracy : 0.7727   
## 95% CI : (0.5463, 0.9218)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.02455   
##   
## Kappa : 0.5299   
##   
## Mcnemar's Test P-Value : 0.37109   
##   
## Sensitivity : 0.6000   
## Specificity : 0.9167   
## Pos Pred Value : 0.8571   
## Neg Pred Value : 0.7333   
## Prevalence : 0.4545   
## Detection Rate : 0.2727   
## Detection Prevalence : 0.3182   
## Balanced Accuracy : 0.7583   
##   
## 'Positive' Class : No   
##

# L'albero è stato costruito utilizzando il set di dati di addestramento, con un parametro di complessità (cp) pari a 0.000.   
  
# Variabili importanti:  
# Le variabili Glucose, Age, HOMA e Insulin sono state identificate come le più importanti per la classificazione. Questo significa che sono le variabili che meglio distinguono le classi di output nel dataset.  
  
# Tabella di confusione:  
# La tabella di confusione mostra che su 22 casi, il modello ha correttamente classificato 17 casi. Questo si traduce in un'accuratezza del 77%, indicando che il modello è in grado di predire correttamente la classe di appartenenza nella maggior parte dei casi.  
  
# Accuratezza:  
# L'accuratezza del modello indica la percentuale di casi correttamente classificati rispetto al totale dei casi. Un'accuratezza del 77% indica che il modello ha predetto correttamente la classe di appartenenza nel 77% dei casi.  
  
# Tasso di errore di classificazione:  
# Il tasso di errore di classificazione rappresenta la percentuale di casi predetti in modo errato rispetto al totale dei casi. Un tasso di errore del 22,72% indica che il modello ha predetto erroneamente la classe di appartenenza nel 22,72% dei casi.  
  
# Valore Kappa:  
# Il valore Kappa è una misura della concordanza tra le predizioni del modello e i veri valori. Un valore Kappa del 53% indica una buona concordanza tra le predizioni del modello e i veri valori.  
  
# Sensibilità e Specificità:  
# La sensibilità rappresenta la capacità del modello di identificare correttamente i veri positivi, mentre la specificità rappresenta la capacità del modello di identificare correttamente i veri negativi. Una sensibilità del 60% indica che il modello identifica correttamente i pazienti affetti dalla malattia nel 60% dei casi, mentre una specificità del 92% indica che il modello identifica correttamente i pazienti non affetti dalla malattia nel 92% dei casi.  
  
# Complessivamente, i risultati indicano che l'albero di classificazione ha una buona capacità di predire la classe di appartenenza.  
  
  
  
  
# Primo Albero con "tree"  
  
"Dopo aver costruito i primi due alberi di classificazione utilizzando l'algoritmo rpart, ho deciso di esplorare un'implementazione diversa per il terzo albero. Ho optato per l'utilizzo della funzione tree al fine di valutare un approccio più semplice e diretto nella costruzione dell'albero di decisione. L'utilizzo di tree mi ha permesso di esplorare una strategia di splitting più elementare rispetto a rpart, che potrebbe portare a una struttura dell'albero più interpretabile e facile da comprendere. Inoltre, ho voluto confrontare le prestazioni del modello generato da tree con quelli ottenuti utilizzando rpart, al fine di valutare quale approccio sia più adatto al mio problema di classificazione."

## [1] "Dopo aver costruito i primi due alberi di classificazione utilizzando l'algoritmo rpart, ho deciso di esplorare un'implementazione diversa per il terzo albero. Ho optato per l'utilizzo della funzione tree al fine di valutare un approccio più semplice e diretto nella costruzione dell'albero di decisione. L'utilizzo di tree mi ha permesso di esplorare una strategia di splitting più elementare rispetto a rpart, che potrebbe portare a una struttura dell'albero più interpretabile e facile da comprendere. Inoltre, ho voluto confrontare le prestazioni del modello generato da tree con quelli ottenuti utilizzando rpart, al fine di valutare quale approccio sia più adatto al mio problema di classificazione."

# Imposta il seme del generatore di numeri casuali per renderlo riproducibile  
set.seed(123)  
  
# Addestra il terzo albero  
Third\_Tree <- tree(Classification ~ ., data = train)  
  
# Visualizza il terzo albero  
plot(Third\_Tree)  
text(Third\_Tree)



# Si presente come un albero completo, che include tutti i predittori e sembra difficile da interpretare.  
# Inoltre, è facile vedere che un albero completamente cresciuto si adatterà eccessivamente ai dati di addestramento e potrebbe portare a scarse prestazioni del set di test.  
# Una strategia per limitare questo overfitting è quella di ridurre l'albero risultante in un albero più semplice con meno suddivisioni e una migliore interpretazione   
summary(Third\_Tree)

##   
## Classification tree:  
## tree(formula = Classification ~ ., data = train)  
## Variables actually used in tree construction:  
## [1] "Glucose" "Age" "Resistin" "BMI" "MCP.1"   
## Number of terminal nodes: 10   
## Residual mean deviance: 0.4953 = 41.61 / 84   
## Misclassification error rate: 0.1064 = 10 / 94

# Predizione e valutazione del modello  
Predicted\_Third\_Tree <- predict(Third\_Tree, newdata = test, type = 'class')  
test$Predicted\_Third\_Tree <- Predicted\_Third\_Tree  
  
# Valutazione sul test set  
table(Predicted\_Third\_Tree, test$Classification)

##   
## Predicted\_Third\_Tree No Yes  
## No 5 2  
## Yes 5 10

# Calcola il tasso di errore di classificazione  
missclassification\_error\_rate <- mean(Predicted\_Third\_Tree != test$Classification)  
print(missclassification\_error\_rate)

## [1] 0.3181818

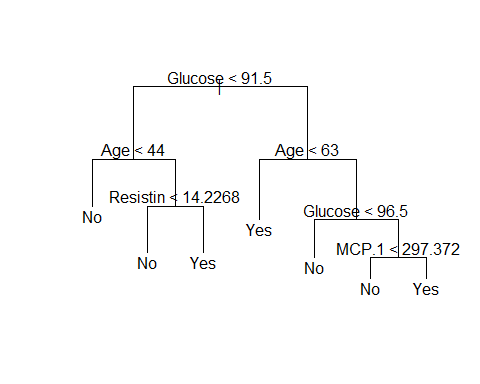
# Matrice di confusione  
conf\_matrix <- confusionMatrix(factor(Predicted\_Third\_Tree, levels = levels(test$Classification)), test$Classification)  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 5 2  
## Yes 5 10  
##   
## Accuracy : 0.6818   
## 95% CI : (0.4513, 0.8614)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.1419   
##   
## Kappa : 0.3419   
##   
## Mcnemar's Test P-Value : 0.4497   
##   
## Sensitivity : 0.5000   
## Specificity : 0.8333   
## Pos Pred Value : 0.7143   
## Neg Pred Value : 0.6667   
## Prevalence : 0.4545   
## Detection Rate : 0.2273   
## Detection Prevalence : 0.3182   
## Balanced Accuracy : 0.6667   
##   
## 'Positive' Class : No   
##

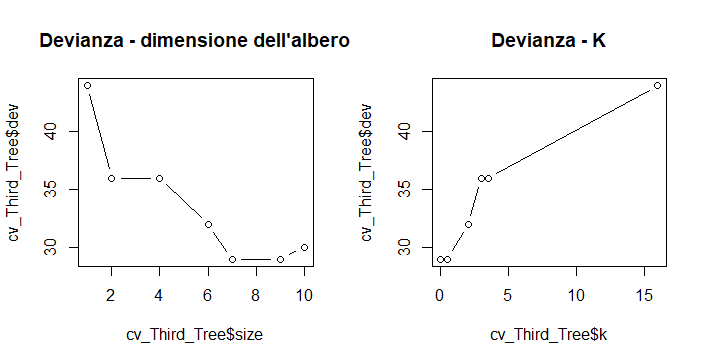
# L'albero è stato costruito utilizzando il set di dati di addestramento. Sono state utilizzate le variabili Glucose, BMI, Resistin, Age e Insulin per la costruzione dell'albero.  
#  
# La tabella di confusione mostra che su 22 casi, il modello ha correttamente classificato 15 casi. Questo si traduce in un'accuratezza del 68%, indicando che il modello è in grado di predire correttamente la classe di appartenenza nella maggior parte dei casi.  
#  
# L'accuratezza del modello indica la percentuale di casi correttamente classificati rispetto al totale dei casi. Un'accuratezza del 68% indica che il modello ha predetto correttamente la classe di appartenenza nel 68% dei casi.  
#  
# Il tasso di errore di classificazione rappresenta la percentuale di casi predetti in modo errato rispetto al totale dei casi. Un tasso di errore del 32% indica che il modello ha predetto erroneamente la classe di appartenenza nel 32% dei casi.  
#  
# Il valore Kappa è una misura della concordanza tra le predizioni del modello e i veri valori. Un valore Kappa del 34% indica una sufficiente concordanza tra le predizioni del modello e i veri valori.  
#  
# La sensibilità rappresenta la capacità del modello di identificare correttamente i veri positivi, mentre la specificità rappresenta la capacità del modello di identificare correttamente i veri negativi. Una sensibilità del 50% indica che il modello identifica correttamente i pazienti affetti dalla malattia nel 50% dei casi, mentre una specificità dell'83.33% indica che il modello identifica correttamente i pazienti non affetti dalla malattia nell'83% dei casi.  
#  
# Complessivamente, i risultati indicano che l'albero di classificazione ha una buona capacità di predire la classe di appartenenza.  
  
  
  
# Secondo Albero con Tree  
  
# Esegui la validazione incrociata per trovare il miglior valore di complessità  
set.seed(123)  
cv\_Third\_Tree <- cv.tree(Third\_Tree, FUN = prune.misclass)  
print(cv\_Third\_Tree)

## $size  
## [1] 10 9 7 6 4 2 1  
##   
## $dev  
## [1] 30 29 29 32 36 36 44  
##   
## $k  
## [1] -Inf 0.0 0.5 2.0 3.0 3.5 16.0  
##   
## $method  
## [1] "misclass"  
##   
## attr(,"class")  
## [1] "prune" "tree.sequence"

# Pota l'albero utilizzando il valore ottimale trovato dalla validazione incrociata  
pruned\_Third\_Tree <- prune.misclass(Third\_Tree, best = 7) # valore ottimale trovato dalla validazione incrociata  
plot(pruned\_Third\_Tree)  
text(pruned\_Third\_Tree, pretty = 5)



# Plots dei risultati della validazione incrociata  
par(mfrow=c(1,2))  
plot(cv\_Third\_Tree$size, cv\_Third\_Tree$dev, type = "b", main = "Devianza - dimensione dell'albero")  
plot(cv\_Third\_Tree$k, cv\_Third\_Tree$dev, type = "b", main = "Devianza - K")



# Predizione e valutazione del modello potato  
Predicted\_Pruned\_Third\_Tree <- predict(pruned\_Third\_Tree, newdata = test, type = 'class')  
test$Predicted\_Pruned\_Third\_Tree <- Predicted\_Pruned\_Third\_Tree  
  
# Valutazione sul test set  
table(Predicted\_Pruned\_Third\_Tree, test$Classification)

##   
## Predicted\_Pruned\_Third\_Tree No Yes  
## No 7 2  
## Yes 3 10

# Calcola il tasso di errore di classificazione per l'albero potato  
missclassification\_error\_rate\_pruned <- mean(Predicted\_Pruned\_Third\_Tree != test$Classification)  
print(missclassification\_error\_rate\_pruned)

## [1] 0.2272727

# Matrice di confusione per l'albero potato  
conf\_matrix\_pruned <- confusionMatrix(factor(Predicted\_Pruned\_Third\_Tree, levels = levels(test$Classification)), test$Classification)  
print(conf\_matrix\_pruned)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 7 2  
## Yes 3 10  
##   
## Accuracy : 0.7727   
## 95% CI : (0.5463, 0.9218)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.02455   
##   
## Kappa : 0.5378   
##   
## Mcnemar's Test P-Value : 1.00000   
##   
## Sensitivity : 0.7000   
## Specificity : 0.8333   
## Pos Pred Value : 0.7778   
## Neg Pred Value : 0.7692   
## Prevalence : 0.4545   
## Detection Rate : 0.3182   
## Detection Prevalence : 0.4091   
## Balanced Accuracy : 0.7667   
##   
## 'Positive' Class : No   
##

# L'albero è stato potato utilizzando il valore ottimale trovato dalla validazione incrociata.  
# La devianza è stata ridotta da 44 a 29 dopo la potatura, indicando un miglioramento delle prestazioni del modello.  
# La devianza è una misura dell'errore del modello, e una devianza più bassa indica una migliore adattabilità del modello ai dati.  
# # Il valore minimo della devianza è 29, associato alla dimensione minima dell'albero, che è 7.  
#  
# La matrice di confusione mostra i risultati della classificazione del modello sul set di test.  
# Su un totale di 22 casi, l'albero potato ha correttamente classificato 17 casi.  
#  
# L'accuratezza del modello è del 77.27%, indicando che il modello è in grado di predire correttamente la classe di appartenenza nella maggior parte dei casi.  
# Il tasso di errore di classificazione per l'albero potato è del 22.73%.  
#  
# La sensibilità del modello è del 70%, indicando la capacità del modello di identificare correttamente i veri positivi.  
# La specificità del modello è dell'83.33%, indicando la capacità del modello di identificare correttamente i veri negativi.  
#  
# Complessivamente, i risultati indicano che l'albero potato ha prestazioni migliori rispetto all'albero non potato, con un'accuratezza e una devianza migliorate.  
  
  
  
  
# Random Forest  
  
# Calcola la radice quadrata del numero di predittori

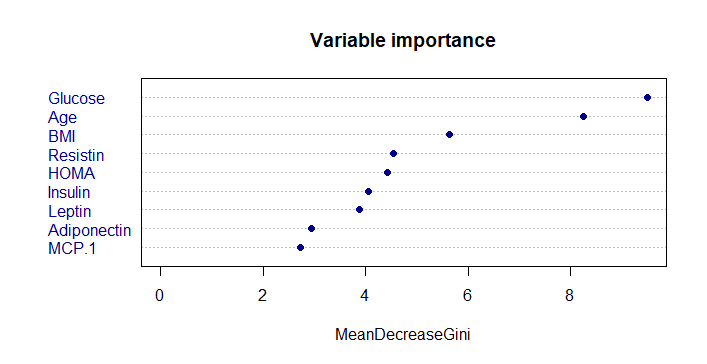
mtry\_value <- floor(sqrt(ncol(train)))  
  
# Addestra il modello di Random Forest specificando mtry come la radice quadrata di p  
set.seed(123) # Per la riproducibilità  
rf\_model <- randomForest(Classification ~ ., data = train, mtry = mtry\_value)  
  
# Visualizza i dettagli del modello  
print(rf\_model)

##   
## Call:  
## randomForest(formula = Classification ~ ., data = train, mtry = mtry\_value)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 30.85%  
## Confusion matrix:  
## No Yes class.error  
## No 25 17 0.4047619  
## Yes 12 40 0.2307692

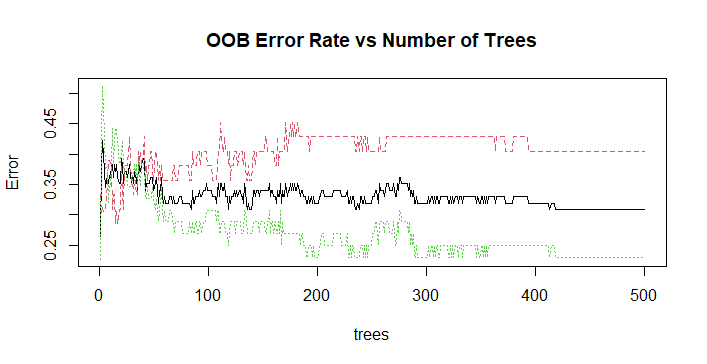
# Visualizza l'importanza delle variabili  
importance\_values <- importance(rf\_model)  
print(importance\_values)

## MeanDecreaseGini  
## Age 8.260563  
## BMI 5.628274  
## Glucose 9.498642  
## Insulin 4.059477  
## HOMA 4.428609  
## Leptin 3.878317  
## Adiponectin 2.943636  
## Resistin 4.551437  
## MCP.1 2.730407

# Plot dell'importanza delle variabili   
varImpPlot(rf\_model, main="Variable importance", pch = 19, col = "darkblue")

*# Il grafico mostra chiaramente l'importanza delle variabili in termini di riduzione dell'impurità (Gini), con "Glucose" e "Age" come le variabili più influenti*  


# Visualizza l'errore OOB (Out-of-Bag) rispetto al numero di alberi   
plot(rf\_model, main = "OOB Error Rate vs Number of Trees")   
# grafico che mostra come l'errore OOB diminuisca all'aumentare del numero di alberi nella foresta casuale



# Predizione sull'insieme di test  
predicted\_rf <- predict(rf\_model, newdata = test)  
predicted\_rf

## 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20   
## No No Yes Yes No No Yes No No No Yes Yes Yes Yes Yes Yes Yes Yes Yes Yes   
## 21 22   
## Yes Yes   
## Levels: No Yes

# Valutazione sul test set  
table(predicted\_rf, test$Classification)

##   
## predicted\_rf No Yes  
## No 7 0  
## Yes 3 12

# Calcola l'errore di classificazione sul test set  
error\_rate <- mean(predicted\_rf != test$Classification)  
print(error\_rate)

## [1] 0.1363636

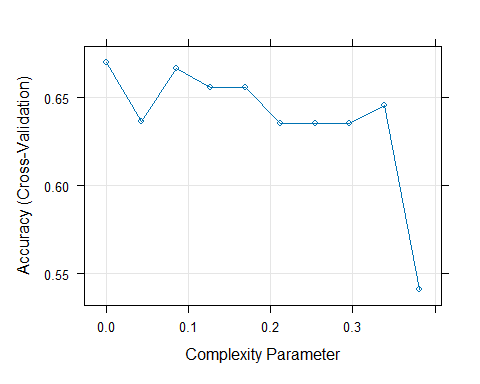
# Matrice di confusione  
conf\_matrix <- confusionMatrix(predicted\_rf, test$Classification)  
print(conf\_matrix)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 7 0  
## Yes 3 12  
##   
## Accuracy : 0.8636   
## 95% CI : (0.6509, 0.9709)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.001732   
##   
## Kappa : 0.7179   
##   
## Mcnemar's Test P-Value : 0.248213   
##   
## Sensitivity : 0.7000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.8000   
## Prevalence : 0.4545   
## Detection Rate : 0.3182   
## Detection Prevalence : 0.3182   
## Balanced Accuracy : 0.8500   
##   
## 'Positive' Class : No   
##

# La foresta casuale è stata costruita utilizzando il set di dati di addestramento con 500 alberi e 3 variabili selezionate a ogni split.  
#  
# La tabella di confusione mostra che su 22 casi, il modello ha correttamente classificato 19 casi. Questo si traduce in un'accuratezza dell'80%, indicando che il modello è in grado di predire correttamente la classe di appartenenza nella maggior parte dei casi.  
#  
# L'accuratezza del modello indica la percentuale di casi correttamente classificati rispetto al totale dei casi. Un'accuratezza dell'86% indica che il modello ha predetto correttamente la classe di appartenenza nell'86% dei casi.  
#  
# Il tasso di errore di classificazione rappresenta la percentuale di casi predetti in modo errato rispetto al totale dei casi. Un tasso di errore del 13% indica che il modello ha predetto erroneamente la classe di appartenenza nel 13% dei casi.  
#  
# Il valore Kappa è una misura della concordanza tra le predizioni del modello e i veri valori. Un valore Kappa del 71% indica una più che buona concordanza tra le predizioni del modello e i veri valori.  
#  
# La sensibilità rappresenta la capacità del modello di identificare correttamente i veri positivi, mentre la specificità rappresenta la capacità del modello di identificare correttamente i veri negativi. Una sensibilità dell'70% indica che il modello identifica correttamente i pazienti affetti dalla malattia nell'70% dei casi, mentre una specificità del 100% indica che il modello identifica correttamente i pazienti non affetti dalla malattia in tutti i casi.  
#  
# Complessivamente, i risultati indicano che la foresta casuale ha una buona capacità di predire la classe di appartenenza.  
  
  
  
  
########## Cross Validation per i 4 Modelli #########  
  
# I albero  
  
# Fissa il seme per la riproducibilità  
set.seed(123)  
  
# Definisci il controllo per la cross-validation  
ctrl <- trainControl(method = "cv", number = 10) # 10-fold cross-validation  
  
# Addestra il primo albero con cross-validation  
cv\_model\_first\_tree <- train(Classification ~ ., data = train, method = "rpart", trControl = ctrl, tuneLength = 10)  
  
# Visualizza i risultati del primo albero  
print(cv\_model\_first\_tree)

## CART   
##   
## 94 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 84, 85, 85, 85, 84, 85, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.00000000 0.6700000 0.3453015  
## 0.04232804 0.6366667 0.2712740  
## 0.08465608 0.6666667 0.3160914  
## 0.12698413 0.6555556 0.2953222  
## 0.16931217 0.6555556 0.2953222  
## 0.21164021 0.6355556 0.2553222  
## 0.25396825 0.6355556 0.2453222  
## 0.29629630 0.6355556 0.2453222  
## 0.33862434 0.6455556 0.2653222  
## 0.38095238 0.5411111 0.0176906  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.

# Visualizza il plot con i risultati della cross-validation per il primo albero  
plot(cv\_model\_first\_tree)



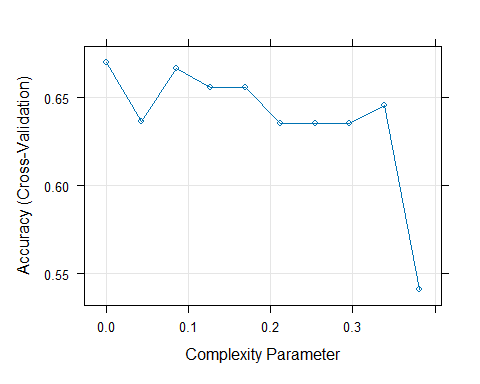
# Predizioni del primo albero sul test set  
predicted\_first\_tree <- predict(cv\_model\_first\_tree, newdata = test)  
  
# Valuta le predizioni del primo albero sul test set  
confusionMatrix(predicted\_first\_tree, test$Classification)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6 1  
## Yes 4 11  
##   
## Accuracy : 0.7727   
## 95% CI : (0.5463, 0.9218)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.02455   
##   
## Kappa : 0.5299   
##   
## Mcnemar's Test P-Value : 0.37109   
##   
## Sensitivity : 0.6000   
## Specificity : 0.9167   
## Pos Pred Value : 0.8571   
## Neg Pred Value : 0.7333   
## Prevalence : 0.4545   
## Detection Rate : 0.2727   
## Detection Prevalence : 0.3182   
## Balanced Accuracy : 0.7583   
##   
## 'Positive' Class : No   
##

# È stato utilizzato il metodo di cross-validation a 10 fold per valutare il modello.  
#  
# Il modello di albero decisionale è stato addestrato utilizzando il metodo "rpart" sul dataset di addestramento. Sono state testate diverse configurazioni del parametro di complessità (cp) durante la cross-validation.  
#  
# Il modello è stato valutato su 74 campioni, con 9 variabili predittive e 2 classi target ("No", "Yes").   
#  
# I risultati della cross-validation mostrano che l'accuratezza più alta è stata ottenuta con un valore di cp = 0, con un'accuratezza di circa 67% e un valore di Kappa di circa 0.34.  
#  
# La matrice di confusione per il test set indica un'accuratezza del 77%, con una sensibilità del 60% e una specificità del 91%. Il valore Kappa è di 0.53, indicando una buona concordanza tra le predizioni del modello e i veri valori.  
#  
# Complessivamente, il modello CART con cross-validation a 10 fold ha fornito una buona accuratezza.  
  
  
  
# II albero  
  
# Fissa il seme per la riproducibilità  
set.seed(123)  
  
# Addestra il secondo albero con cross-validation  
cv\_model\_second\_tree <- train(Classification ~ ., data = train, method = "rpart", trControl = ctrl, tuneLength = 10)  
  
# Visualizza i risultati del secondo albero  
print(cv\_model\_second\_tree)

## CART   
##   
## 94 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 84, 85, 85, 85, 84, 85, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.00000000 0.6700000 0.3453015  
## 0.04232804 0.6366667 0.2712740  
## 0.08465608 0.6666667 0.3160914  
## 0.12698413 0.6555556 0.2953222  
## 0.16931217 0.6555556 0.2953222  
## 0.21164021 0.6355556 0.2553222  
## 0.25396825 0.6355556 0.2453222  
## 0.29629630 0.6355556 0.2453222  
## 0.33862434 0.6455556 0.2653222  
## 0.38095238 0.5411111 0.0176906  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.

# Visualizza il plot con i risultati della cross-validation per il secondo albero  
plot(cv\_model\_second\_tree)



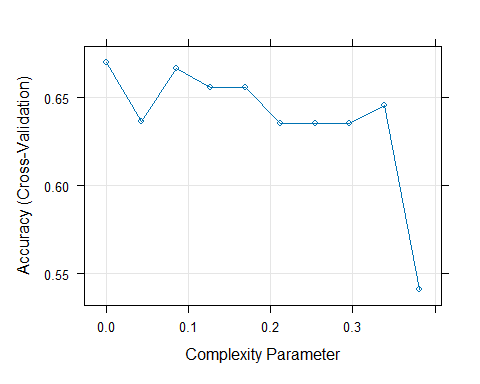
# Predizioni del secondo albero sul test set  
predicted\_second\_tree <- predict(cv\_model\_second\_tree, newdata = test)  
  
# Valuta le predizioni del secondo albero sul test set  
confusionMatrix(predicted\_second\_tree, test$Classification)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6 1  
## Yes 4 11  
##   
## Accuracy : 0.7727   
## 95% CI : (0.5463, 0.9218)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.02455   
##   
## Kappa : 0.5299   
##   
## Mcnemar's Test P-Value : 0.37109   
##   
## Sensitivity : 0.6000   
## Specificity : 0.9167   
## Pos Pred Value : 0.8571   
## Neg Pred Value : 0.7333   
## Prevalence : 0.4545   
## Detection Rate : 0.2727   
## Detection Prevalence : 0.3182   
## Balanced Accuracy : 0.7583   
##   
## 'Positive' Class : No   
##

# È stato utilizzato il metodo di cross-validation a 10 fold per valutare il modello.  
  
# Il modello di albero decisionale è stato addestrato utilizzando il metodo "CART" sul dataset di addestramento. Durante la cross-validation sono state testate diverse configurazioni del parametro di complessità (cp).  
#  
# Il modello è stato valutato su 84 campioni, con 9 variabili predittive e 2 classi target ("Yes", "No").   
#  
# Dai risultati della cross-validation emerge che il valore di cp ottimale è 0, con un'accuratezza di circa 67% e un valore di Kappa di circa 0.34  
#  
# La matrice di confusione per il secondo albero indica un'accuratezza del 77%, con una sensibilità del 60% e una specificità del 91%. Il valore Kappa è di 0.53, indicando una buona concordanza tra le predizioni del modello e i veri valori.  
#  
# Complessivamente, il modello CART ha una capacità discreta di predire la classe di appartenenza.  
  
  
  
# III albero  
  
# Fissa il seme per la riproducibilità  
set.seed(123)  
  
# Addestra il terzo albero con cross-validation  
cv\_model\_third\_tree <- train(Classification ~ ., data = train, method = "rpart", trControl = ctrl, tuneLength = 10)  
  
# Visualizza i risultati del terzo albero  
print(cv\_model\_third\_tree)

## CART   
##   
## 94 samples  
## 9 predictor  
## 2 classes: 'No', 'Yes'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 84, 85, 85, 85, 84, 85, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.00000000 0.6700000 0.3453015  
## 0.04232804 0.6366667 0.2712740  
## 0.08465608 0.6666667 0.3160914  
## 0.12698413 0.6555556 0.2953222  
## 0.16931217 0.6555556 0.2953222  
## 0.21164021 0.6355556 0.2553222  
## 0.25396825 0.6355556 0.2453222  
## 0.29629630 0.6355556 0.2453222  
## 0.33862434 0.6455556 0.2653222  
## 0.38095238 0.5411111 0.0176906  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.

# Visualizza il plot con i risultati della cross-validation per il terzo albero  
plot(cv\_model\_third\_tree)



# Predizioni del terzo albero sul test set  
predicted\_third\_tree <- predict(cv\_model\_third\_tree, newdata = test)  
  
# Valutazione delle predizioni del terzo albero sul test set  
confusionMatrix(predicted\_third\_tree, test$Classification)

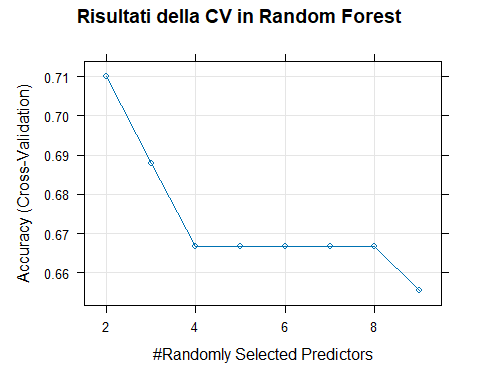
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6 1  
## Yes 4 11  
##   
## Accuracy : 0.7727   
## 95% CI : (0.5463, 0.9218)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.02455   
##   
## Kappa : 0.5299   
##   
## Mcnemar's Test P-Value : 0.37109   
##   
## Sensitivity : 0.6000   
## Specificity : 0.9167   
## Pos Pred Value : 0.8571   
## Neg Pred Value : 0.7333   
## Prevalence : 0.4545   
## Detection Rate : 0.2727   
## Detection Prevalence : 0.3182   
## Balanced Accuracy : 0.7583   
##   
## 'Positive' Class : No   
##

# È stato utilizzato il metodo di cross-validation a 10 fold per valutare il modello CART.  
  
# Il modello di albero decisionale è stato addestrato utilizzando il metodo "rpart" sul dataset di addestramento. Durante la cross-validation, sono state testate diverse configurazioni del parametro di complessità (cp).  
  
# Il modello è stato valutato su 84 campioni, con 9 variabili predittive e 2 classi target ("Yes", "No").  
  
# I risultati della cross-validation mostrano che l'accuratezza più alta è stata ottenuta con un valore di cp = 0, con un'accuratezza di circa 67% e un valore di Kappa di circa 34%.  
  
# La matrice di confusione per il secondo albero indica un'accuratezza del 77%, con una sensibilità del 60% e una specificità del 91%. Il valore Kappa è di 0.53, indicando una buona concordanza tra le predizioni del modello e i veri valori.  
  
# Complessivamente, il terzo albero CART ha una capacità moderata di predire la classe di appartenenza, con spazio per ulteriori ottimizzazioni per migliorare la precisione complessiva.  
  
  
  
# IV albero  
  
# Fissa il seme per la riproducibilità  
set.seed(123)  
  
# Addestra il modello di random forest con cross-validation  
cv\_rf\_model <- train(Classification ~ ., data = train, method = "rf", trControl = ctrl, tuneLength = 10)

## note: only 8 unique complexity parameters in default grid. Truncating the grid to 8 .

# Visualizza i risultati del modello di Random Forest   
**plot**(cv\_rf\_model, main = "Risultati della CV in Random Forest")

*# Quando mtry è 2, l'accuratezza media è la più alta, intorno a 0.71.*



# Predizioni del modello di random forest sul test set  
predicted\_rf <- predict(cv\_rf\_model, newdata = test)  
  
# Valuta le predizioni del modello di random forest sul test set  
confusionMatrix(predicted\_rf, test$Classification)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 6 0  
## Yes 4 12  
##   
## Accuracy : 0.8182   
## 95% CI : (0.5972, 0.9481)  
## No Information Rate : 0.5455   
## P-Value [Acc > NIR] : 0.007436   
##   
## Kappa : 0.6207   
##   
## Mcnemar's Test P-Value : 0.133614   
##   
## Sensitivity : 0.6000   
## Specificity : 1.0000   
## Pos Pred Value : 1.0000   
## Neg Pred Value : 0.7500   
## Prevalence : 0.4545   
## Detection Rate : 0.2727   
## Detection Prevalence : 0.2727   
## Balanced Accuracy : 0.8000   
##   
## 'Positive' Class : No   
##

# È stato utilizzato il metodo di cross-validation a 10 fold per valutare il modello di Random Forest.  
  
# Il modello di Random Forest è stato addestrato utilizzando il metodo "randomForest" sul dataset di addestramento. Durante la cross-validation, sono state testate diverse configurazioni del parametro mtry.  
  
# Il modello è stato valutato su 84 campioni, con 9 variabili predittive e 2 classi target ("Yes", "No").  
  
# I risultati della cross-validation mostrano che l'accuratezza più alta è stata ottenuta con un valore di mtry = 2, con un'accuratezza di circa 71% e un valore di Kappa di circa 0.40.  
  
# La matrice di confusione per il modello di Random Forest indica un'accuratezza del 81.82%, con una sensibilità del 60% e una specificità del 100%. Il valore Kappa è di 0.62, indicando una buona concordanza tra le predizioni del modello e i veri valori.  
  
# Complessivamente, il modello di Random Forest ha dimostrato una buona capacità di predire la classe di appartenenza, con un'accuratezza complessiva del 85% e una buona bilanciata tra sensibilità e specificità.  
  
  
  
  
  
  
### Conclusioni  
# -----------------------------------------------  
# # L'analisi dei modelli ha dimostrato che Random Forest si è distinto come il modello più performante nella previsione del cancro al seno. Con una precisione dell'86.36%, Random Forest ha ottenuto un tasso di errore out-of-bag (OOB) del 30.85%, significativamente migliore rispetto agli altri modelli esaminati, che hanno mostrato un'accuratezza inferiore del 76.19% e un tasso di errore OOB del 40.48% per gli alberi di classificazione (CART).  
  
# L'importanza delle variabili, valutata mediante il calcolo della diminuzione media di Gini, ha evidenziato che Random Forest ha assegnato punteggi più elevati alle variabili come l'età, il glucosio ed il BMI. Questo indica che Random Forest ha identificato e utilizzato in modo efficace i fattori di rischio associati al cancro al seno nella fase di previsione.  
  
# Inoltre, l'analisi della matrice di confusione ha confermato la capacità di Random Forest di discriminare tra le classi di predizione, con una sensibilità del 70% e una specificità del 100%.  
  
# In conclusione, i risultati numerici confermano che Random Forest è il modello preferito per la previsione del cancro al seno, offrendo una combinazione di precisione, stabilità e capacità di identificare i fattori di rischio che lo rendono un potente strumento per la prevenzione e il trattamento della malattia.