Cancer de mama

Por definir

Jose Felix Rojas Cabeza

2022 / April / 05

Table of Contents

[Data 1](#_Toc100080839)

[Data collection 1](#_Toc100080840)

[Data exploration and preparation 3](#_Toc100080841)

[Correlogram 4](#_Toc100080842)

[Creation training and test datasets for 10-fold crossvalidation 5](#_Toc100080843)

[KNN 6](#_Toc100080844)

[Confusion matrix KNN 6](#_Toc100080845)

[SVM 7](#_Toc100080846)

[Confusion matrix SVM 7](#_Toc100080847)

[RF 8](#_Toc100080848)

[Matriz de Confusión RF 9](#_Toc100080849)

[TABLA 1 9](#_Toc100080850)

[CURVAS ROC 11](#_Toc100080851)

# Data

## Data collection

> bc\_full <- read.csv(params$data1, stringsAsFactors = FALSE)  
> # dim(bc\_full)  
> # head(bc\_full[1:4])

> dim(bc\_full)

[1] 569 32

> head(bc\_full)

id radius\_mean texture\_mean perimeter\_mean area\_mean smoothness\_mean  
1 842302 17.99 10.38 122.80 1001.0 0.11840  
2 842517 20.57 17.77 132.90 1326.0 0.08474  
3 84300903 19.69 21.25 130.00 1203.0 0.10960  
4 84348301 11.42 20.38 77.58 386.1 0.14250  
5 84358402 20.29 14.34 135.10 1297.0 0.10030  
6 843786 12.45 15.70 82.57 477.1 0.12780  
 compactness\_mean concavity\_mean concave\_points\_mean symmetry\_mean  
1 0.27760 0.3001 0.14710 0.2419  
2 0.07864 0.0869 0.07017 0.1812  
3 0.15990 0.1974 0.12790 0.2069  
4 0.28390 0.2414 0.10520 0.2597  
5 0.13280 0.1980 0.10430 0.1809  
6 0.17000 0.1578 0.08089 0.2087  
 fractal\_dimension\_mean radius\_se texture\_se perimeter\_se area\_se  
1 0.07871 1.0950 0.9053 8.589 153.40  
2 0.05667 0.5435 0.7339 3.398 74.08  
3 0.05999 0.7456 0.7869 4.585 94.03  
4 0.09744 0.4956 1.1560 3.445 27.23  
5 0.05883 0.7572 0.7813 5.438 94.44  
6 0.07613 0.3345 0.8902 2.217 27.19  
 smoothness\_se compactness\_se concavity\_se concave\_points\_se symmetry\_se  
1 0.006399 0.04904 0.05373 0.01587 0.03003  
2 0.005225 0.01308 0.01860 0.01340 0.01389  
3 0.006150 0.04006 0.03832 0.02058 0.02250  
4 0.009110 0.07458 0.05661 0.01867 0.05963  
5 0.011490 0.02461 0.05688 0.01885 0.01756  
6 0.007510 0.03345 0.03672 0.01137 0.02165  
 fractal\_dimension\_se radius\_worst texture\_worst perimeter\_worst area\_worst  
1 0.006193 25.38 17.33 184.60 2019.0  
2 0.003532 24.99 23.41 158.80 1956.0  
3 0.004571 23.57 25.53 152.50 1709.0  
4 0.009208 14.91 26.50 98.87 567.7  
5 0.005115 22.54 16.67 152.20 1575.0  
6 0.005082 15.47 23.75 103.40 741.6  
 smoothness\_worst compactness\_worst concavity\_worst concave\_points\_worst  
1 0.1622 0.6656 0.7119 0.2654  
2 0.1238 0.1866 0.2416 0.1860  
3 0.1444 0.4245 0.4504 0.2430  
4 0.2098 0.8663 0.6869 0.2575  
5 0.1374 0.2050 0.4000 0.1625  
6 0.1791 0.5249 0.5355 0.1741  
 symmetry\_worst fractal\_dimension\_worst diagnosis  
1 0.4601 0.11890 M  
2 0.2750 0.08902 M  
3 0.3613 0.08758 M  
4 0.6638 0.17300 M  
5 0.2364 0.07678 M  
6 0.3985 0.12440 M

## Data exploration and preparation

> # 1234567890123456789012345678901234567890123456789012345678901234567890  
> bc <- bc\_full[,-1]   
>   
> # Figura 1. Izq  
> #str(bc)  
> table(is.na(bc))

FALSE   
17639

> # Benign / Malignant  
> bc$diagnosis <- factor(bc$diagnosis, levels= c("B", "M"), labels=c("Benign", "Malignant"))  
> table(bc$diagnosis)

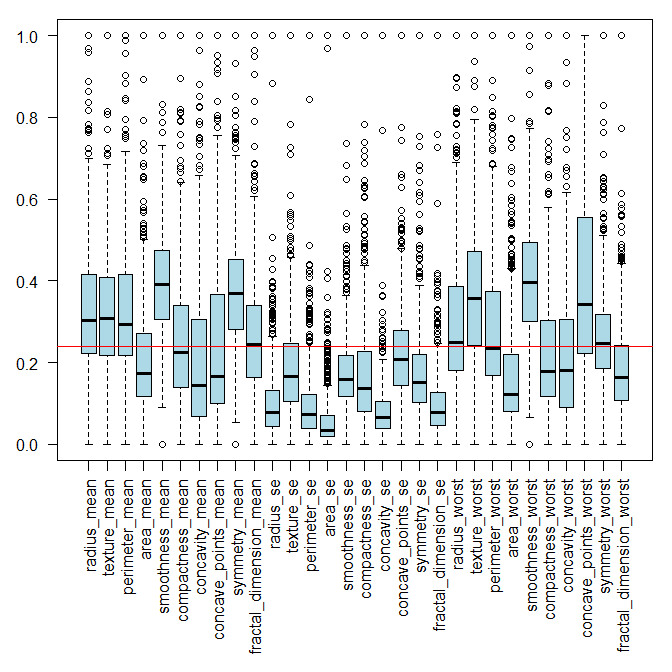
Benign Malignant   
 357 212

> table(is.na(bc))

FALSE   
17639

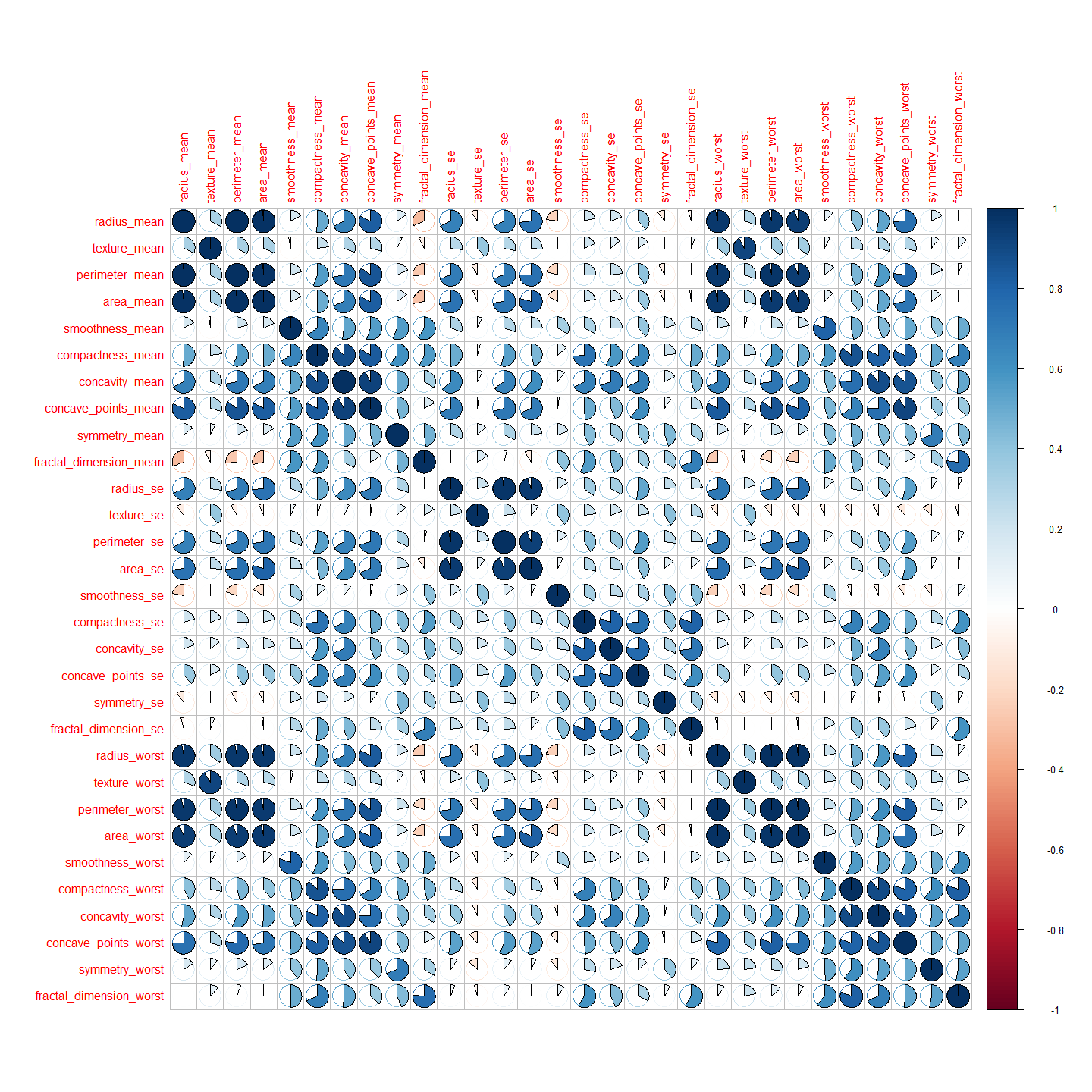
> normalize <- function(x) {  
+ return ((x - min(x)) / (max(x) - min(x)))  
+ }  
> bc\_n <- as.data.frame(lapply(bc[1:(round(dim[2],0)-1)], normalize))

> # Figura 1. der  
> par(mar=c(11,3,1,1))  
> means\_n <- as.numeric((lapply(bc\_n, mean)))  
> boxplot(bc\_n[1:30], las=2, col="lightblue")  
> # , main="Figura 2. Distribucion de las variables del dataset normalizadas"  
> abline(h=mean(means\_n), col="red")



# Correlogram

> library(corrplot)  
> par(mar=c(11,3,1,1))  
> C <- cor(round(as.matrix(bc\_n[1:30]), 4))  
> corrplot(C, method = "pie")



# Creation training and test datasets for 10-fold crossvalidation

> # 2021-07-19  
> library(caret)  
> bc\_n\_full <- cbind(bc\_n, bc$diagnosis)  
>   
> names(bc\_n\_full)[31] <- "diagnosis"  
> dim(bc\_n\_full)

[1] 569 31

> set.seed(params$seed\_train)  
> inTrain <- createDataPartition( y = 1:nrow(bc\_n\_full),  
+ p = params$ptrain,  
+ list = FALSE )  
> # KNN, SVM, RF  
> train <- bc\_n\_full[inTrain, ]  
> test <- bc\_n\_full[-inTrain, ]  
> table(train[,31])

Benign Malignant   
 241 140

> table(test[,31])

Benign Malignant   
 116 72

> #ANN  
> #Data Partition  
> set.seed(params$seed\_alg)  
> inTrain\_ann <- createDataPartition(y=bc\_n\_full$diagnosis, p=round(params$ptrain,4), list=FALSE)  
> # Normalized dataset  
> ann\_data\_n <- cbind(bc\_n, bc$diagnosis)  
> names(ann\_data\_n)[31] <- "diagnosis"  
> train\_ann <- ann\_data\_n[inTrain\_ann, ]  
> test\_ann <- ann\_data\_n[-inTrain\_ann, ]  
> #train.set <- dataset[inTrain,]  
> #test.set <- dataset[-inTrain,]  
> ( c(nrow(train)/nrow(test), nrow(train\_ann)/nrow(test\_ann)) )

[1] 2.026596 2.026596

> # should be around 2

# KNN

> set.seed(params$seed\_alg)  
> start\_time <- Sys.time()  
> ctrl <- trainControl(method="repeatedcv",repeats = 10, number = 10,   
+ classProbs = T, savePredictions = T)   
> model\_knn <- train(diagnosis ~ ., data = train, method = "knn",   
+ trControl = ctrl, preProcess = c("center","scale"),  
+ tuneLength = 20)  
> end\_time <- Sys.time()  
> (knn\_time <- end\_time - start\_time)

Time difference of 20.97821 secs

## Confusion matrix KNN

> prediction <- predict(model\_knn, test)   
> ct <- table(prediction, test$diagnosis)   
> (cm\_knn <- confusionMatrix(ct, positive="Malignant"))

Confusion Matrix and Statistics  
  
   
prediction Benign Malignant  
 Benign 115 6  
 Malignant 1 66  
   
 Accuracy : 0.9628   
 95% CI : (0.9248, 0.9849)  
 No Information Rate : 0.617   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.9202   
   
 Mcnemar's Test P-Value : 0.1306   
   
 Sensitivity : 0.9167   
 Specificity : 0.9914   
 Pos Pred Value : 0.9851   
 Neg Pred Value : 0.9504   
 Prevalence : 0.3830   
 Detection Rate : 0.3511   
 Detection Prevalence : 0.3564   
 Balanced Accuracy : 0.9540   
   
 'Positive' Class : Malignant

# SVM

> set.seed(params$seed\_alg)  
> start\_time <- Sys.time()  
> model\_svm <- train(diagnosis ~ ., train, method='svmLinear',  
+ trControl= trainControl(method='cv',number=10,repeats = 10,  
+ classProbs = T,  
+ savePredictions = T),  
+ tuneGrid= NULL, trace = FALSE)  
> end\_time <- Sys.time()  
> (svm\_time <- end\_time - start\_time)

Time difference of 1.87234 secs

## Confusion matrix SVM

> prediction <- predict(model\_svm, test)   
> ct <- table(prediction, test$diagnosis)   
> (cm\_svm <- confusionMatrix(ct, positive="Malignant"))

Confusion Matrix and Statistics  
  
   
prediction Benign Malignant  
 Benign 115 5  
 Malignant 1 67  
   
 Accuracy : 0.9681   
 95% CI : (0.9318, 0.9882)  
 No Information Rate : 0.617   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.9318   
   
 Mcnemar's Test P-Value : 0.2207   
   
 Sensitivity : 0.9306   
 Specificity : 0.9914   
 Pos Pred Value : 0.9853   
 Neg Pred Value : 0.9583   
 Prevalence : 0.3830   
 Detection Rate : 0.3564   
 Detection Prevalence : 0.3617   
 Balanced Accuracy : 0.9610   
   
 'Positive' Class : Malignant

# RF

> set.seed(params$seed\_alg)  
> ctrl <- trainControl( method = "repeatedcv",  
+ number = 10,  
+ repeats = 10,  
+ summaryFunction = defaultSummary,  
+ verboseIter = FALSE,  
+ classProbs = TRUE,  
+ savePredictions = TRUE)  
> # repeats=3 )  
> ## Tunegrid for Random Forest  
> # mtry define cuantas variables se seleccionan al azar en cada split. Por  
> # defecto sqrt(n.variables)  
> grid\_rf <- expand.grid(.mtry = c(2,4,8,16))  
> start\_time <- Sys.time()  
> model\_rf <- train (diagnosis ~ .,  
+ data = train,  
+ method ="rf",  
+ trControl=ctrl,  
+ tuneGrid = grid\_rf,  
+ metric="Accuracy",  
+ prePoc = c("center", "scale"),  
+ verbose =FALSE,  
+ trace = FALSE  
+ )  
> end\_time <- Sys.time()  
> (rf\_time <- end\_time - start\_time)

Time difference of 1.12777 mins

## Matriz de Confusión RF

> pred\_rf <- predict (model\_rf, newdata = test)  
> (cm\_rf <- confusionMatrix(data=pred\_rf, test$diagnosis, positive = "Malignant"))

Confusion Matrix and Statistics  
  
 Reference  
Prediction Benign Malignant  
 Benign 114 6  
 Malignant 2 66  
   
 Accuracy : 0.9574   
 95% CI : (0.9179, 0.9815)  
 No Information Rate : 0.617   
 P-Value [Acc > NIR] : <2e-16   
   
 Kappa : 0.909   
   
 Mcnemar's Test P-Value : 0.2888   
   
 Sensitivity : 0.9167   
 Specificity : 0.9828   
 Pos Pred Value : 0.9706   
 Neg Pred Value : 0.9500   
 Prevalence : 0.3830   
 Detection Rate : 0.3511   
 Detection Prevalence : 0.3617   
 Balanced Accuracy : 0.9497   
   
 'Positive' Class : Malignant

# TABLA 1

> print("KNN")

[1] "KNN"

> cm\_knn$overall[1]

Accuracy   
0.962766

> cm\_knn$overall[2]

Kappa   
0.920165

> cm\_knn$byClass

Sensitivity Specificity Pos Pred Value   
 0.9166667 0.9913793 0.9850746   
 Neg Pred Value Precision Recall   
 0.9504132 0.9850746 0.9166667   
 F1 Prevalence Detection Rate   
 0.9496403 0.3829787 0.3510638   
Detection Prevalence Balanced Accuracy   
 0.3563830 0.9540230

> print("SVM")

[1] "SVM"

> cm\_svm$overall[1]

Accuracy   
0.9680851

> cm\_svm$overall[2]

Kappa   
0.9317522

> cm\_svm$byClass

Sensitivity Specificity Pos Pred Value   
 0.9305556 0.9913793 0.9852941   
 Neg Pred Value Precision Recall   
 0.9583333 0.9852941 0.9305556   
 F1 Prevalence Detection Rate   
 0.9571429 0.3829787 0.3563830   
Detection Prevalence Balanced Accuracy   
 0.3617021 0.9609674

> print("RF")

[1] "RF"

> cm\_rf$overall[1]

Accuracy   
0.9574468

> cm\_rf$overall[2]

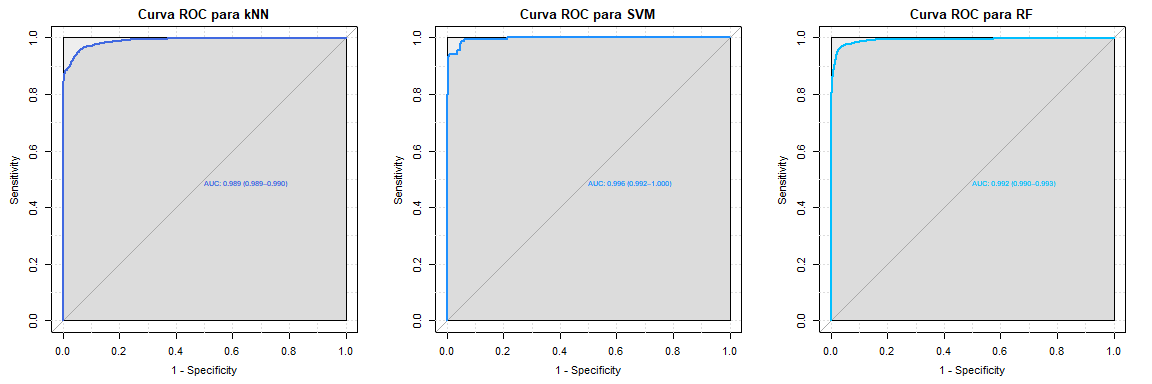
Kappa   
0.9090029

> cm\_rf$byClass

Sensitivity Specificity Pos Pred Value   
 0.9166667 0.9827586 0.9705882   
 Neg Pred Value Precision Recall   
 0.9500000 0.9705882 0.9166667   
 F1 Prevalence Detection Rate   
 0.9428571 0.3829787 0.3510638   
Detection Prevalence Balanced Accuracy   
 0.3617021 0.9497126

# CURVAS ROC

> library(pROC)  
> # "royalblue","dodgerblue", "deepskyblue", "lightskyblue"  
> par(mfrow=c(1,3))  
> plot.roc(model\_knn$pred$obs,  
+ model\_knn$pred$Malignant,  
+ smoothed = TRUE,  
+ legacy.axes=TRUE,  
+ # arguments for ci  
+ ci=TRUE, ci.alpha=0.9, stratified=FALSE,  
+ # arguments for plot  
+ #plot=TRUE,   
+ auc.polygon=TRUE,   
+ max.auc.polygon=TRUE, grid=TRUE,  
+ print.auc=TRUE, show.thres=TRUE,   
+ col="royalblue", main = "Curva ROC para kNN")  
> plot.roc(model\_svm$pred$obs,  
+ model\_svm$pred$Malignant,  
+ smoothed = TRUE,  
+ legacy.axes=TRUE,  
+ # arguments for ci  
+ ci=TRUE, ci.alpha=0.9, stratified=FALSE,  
+ # arguments for plot  
+ #plot=TRUE,   
+ auc.polygon=TRUE,   
+ max.auc.polygon=TRUE, grid=TRUE,  
+ print.auc=TRUE, show.thres=TRUE,  
+ col="dodgerblue", main = "Curva ROC para SVM")  
> plot.roc(model\_rf$pred$obs,  
+ model\_rf$pred$Malignant,  
+ smoothed = TRUE,  
+ legacy.axes=TRUE,  
+ # arguments for ci  
+ ci=TRUE, ci.alpha=0.9, stratified=FALSE,  
+ # arguments for plot  
+ #plot=TRUE,   
+ auc.polygon=TRUE,   
+ max.auc.polygon=TRUE, grid=TRUE,  
+ print.auc=TRUE, show.thres=TRUE,  
+ col="deepskyblue", main = "Curva ROC para RF")



> # https://stackoverflow.com/questions/46891681/calculating-auc-from-nnet-model   
> # https://www.r-bloggers.com/2016/11/calculating-auc-the-area-under-a-roc-curve/   
> #plot.roc(test\_ann$diagnosis,   
> # prediction\_ann\_prob$Malignant,   
> # smoothed = TRUE,  
> # legacy.axes=TRUE,  
> # ci=TRUE, ci.alpha=0.9, stratified=FALSE,  
> # auc.polygon=TRUE,   
> # max.auc.polygon=TRUE, grid=TRUE,  
> # print.auc=TRUE, show.thres=TRUE,  
> # col="blue", main = "Curva ROC para RF")

> print(sessionInfo())

R version 3.6.3 (2020-02-29)  
Platform: x86\_64-w64-mingw32/x64 (64-bit)  
Running under: Windows 10 x64 (build 19044)  
  
Matrix products: default  
  
locale:  
[1] LC\_COLLATE=English\_United States.1252   
[2] LC\_CTYPE=English\_United States.1252   
[3] LC\_MONETARY=English\_United States.1252  
[4] LC\_NUMERIC=C   
[5] LC\_TIME=C   
  
attached base packages:  
[1] stats graphics grDevices utils datasets methods base   
  
other attached packages:  
[1] pROC\_1.17.0.1 caret\_6.0-86 lattice\_0.20-38 ggplot2\_3.3.3   
[5] corrplot\_0.84 knitr\_1.31   
  
loaded via a namespace (and not attached):  
 [1] tidyselect\_1.1.0 xfun\_0.22 kernlab\_0.9-29   
 [4] reshape2\_1.4.4 purrr\_0.3.4 splines\_3.6.3   
 [7] colorspace\_2.0-0 vctrs\_0.3.7 generics\_0.1.0   
[10] stats4\_3.6.3 htmltools\_0.5.1.1 yaml\_2.2.1   
[13] utf8\_1.2.1 survival\_3.1-8 prodlim\_2019.11.13   
[16] rlang\_0.4.10 e1071\_1.7-7 ModelMetrics\_1.2.2.2  
[19] pillar\_1.5.1 glue\_1.4.2 withr\_2.4.1   
[22] DBI\_1.1.1 foreach\_1.5.1 lifecycle\_1.0.0   
[25] plyr\_1.8.6 lava\_1.6.9 stringr\_1.4.0   
[28] timeDate\_3043.102 munsell\_0.5.0 gtable\_0.3.0   
[31] recipes\_0.1.15 codetools\_0.2-16 evaluate\_0.14   
[34] class\_7.3-15 fansi\_0.4.2 Rcpp\_1.0.6   
[37] scales\_1.1.1 ipred\_0.9-11 digest\_0.6.27   
[40] stringi\_1.5.3 dplyr\_1.0.5 grid\_3.6.3   
[43] tools\_3.6.3 magrittr\_2.0.1 proxy\_0.4-26   
[46] tibble\_3.1.0 randomForest\_4.6-14 crayon\_1.4.1   
[49] pkgconfig\_2.0.3 ellipsis\_0.3.1 MASS\_7.3-51.5   
[52] Matrix\_1.2-18 data.table\_1.14.0 lubridate\_1.7.10   
[55] gower\_0.2.2 assertthat\_0.2.1 rmarkdown\_2.7   
[58] iterators\_1.0.13 R6\_2.5.0 rpart\_4.1-15   
[61] nnet\_7.3-12 nlme\_3.1-144 compiler\_3.6.3