Aprendizagem Estatística em Altas Dimensões [MAE0501/MAE5904/IBI5904]

Ícaro Maia Santos de Castro[1]
Rayssa de Carvalho Roberto[2]
Rodrigo Aoyama Nakahara[3]
Rodrigo Araujo[4]
Vitor Hugo Vieira de Lima[5]

Novembro de 2020

Sumário

[1]Número USP: 11866921
[2]Número USP: 10940828
[3]Número USP: 3510922
[4]Número USP: 9299208
[5]Número USP: 10263886

Leitura dos Dados

```
diabetes <- read_csv("diabetes.csv")
```

```
## Parsed with column specification:
## cols(
##   Pregnancies = col_double(),
##   Glucose = col_double(),
##   BloodPressure = col_double(),
##   SkinThickness = col_double(),
##   Insulin = col_double(),
##   BMI = col_double(),
##   DiabetesPedigreeFunction = col_double(),
##   Age = col_double(),
##   Outcome = col_double()
## )
```

```
colnames(diabetes)[9] <- "diabetes"
diabetes$diabetes <- as.factor(diabetes$diabetes)
levels(diabetes$diabetes) <- c("No","Yes")

# Missings
diabetes[, 2:6][diabetes[, 2:6] == 0] <- NA

head(diabetes) %>% kable(caption="Dados.")
```

Tabela 1: Dados.

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	diabetes
6	148	72	35	NA	33,6	0,63	50	Yes
1	85	66	29	NA	26,6	0,35	31	No
8	183	64	NA	NA	23,3	0,67	32	Yes
1	89	66	23	94	28,1	0,17	21	No
0	137	40	35	168	43,1	2,29	33	Yes
5	116	74	NA	NA	25,6	0,20	30	No

Análise Descritiva

```
# OBS: análise descritiva com base no conjunto de treinamento para evitar data snooping

library(dplyr)
library(caret)
library(lattice)
library(ggplot2)
library(GGally)
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 4.0.2
```

```
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following objects are masked from 'package:psych':
##
##     alpha, rescale
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.0.2
```

```
## Type 'citation("pROC")' for a citation.
```

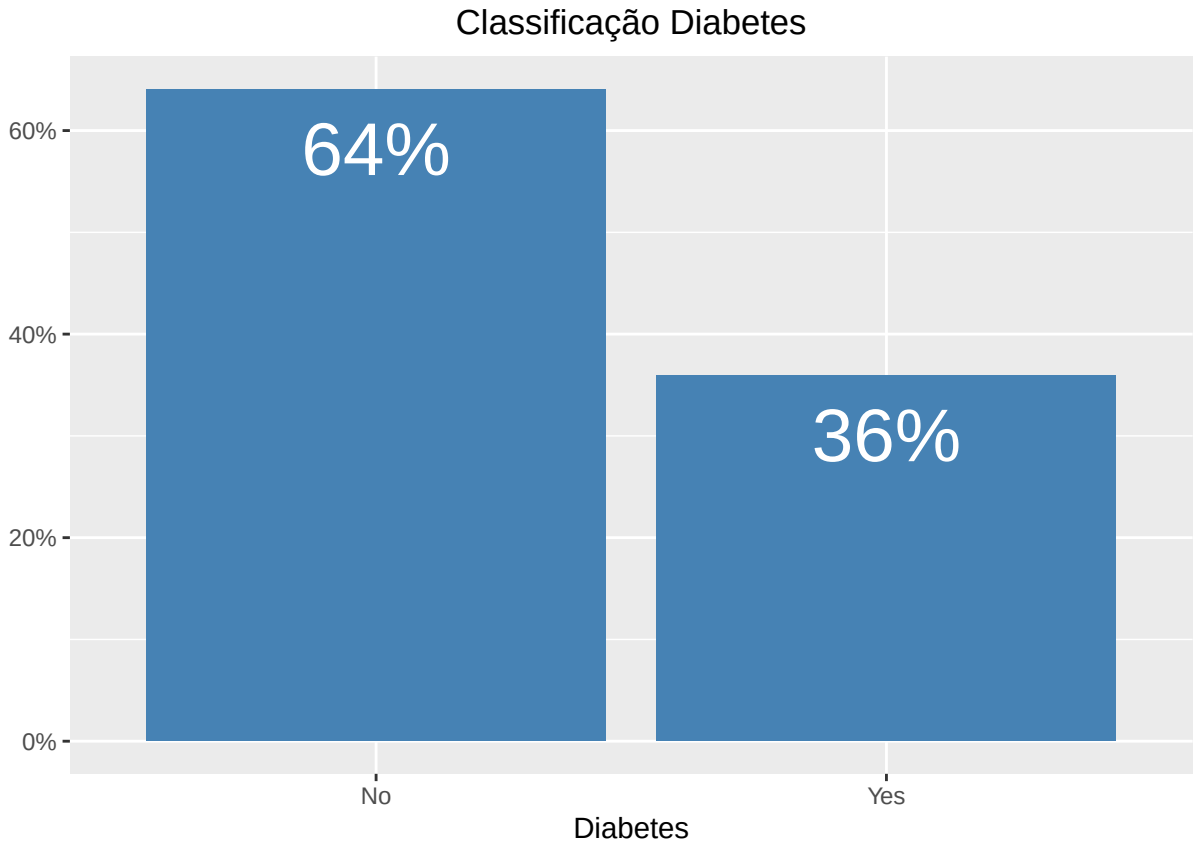
```
##
## Attaching package: 'pROC'
```

```
## The following object is masked from 'package:colorspace':
##
##     coords
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
# Distribuição da variável resposta

train %>% count(Diabetes = factor(diabetes)) %>% mutate(pct = prop.table(n)) %>%
  ggplot(aes(x = Diabetes, y = pct, fill = pct, label = scales::percent(pct))) +
```

```
  geom_col(position = 'dodge', fill="steelblue") +
  labs(title = "Classificação Diabetes", x = "Diabetes", y = "") +
  geom_text(aes(label=scales::percent(pct) ), vjust=1.6, color="white", size=10) +
  scale_y_continuous(labels = scales::percent) +
  theme(plot.title = element_text(hjust = 0.5), legend.title = element_blank())
```



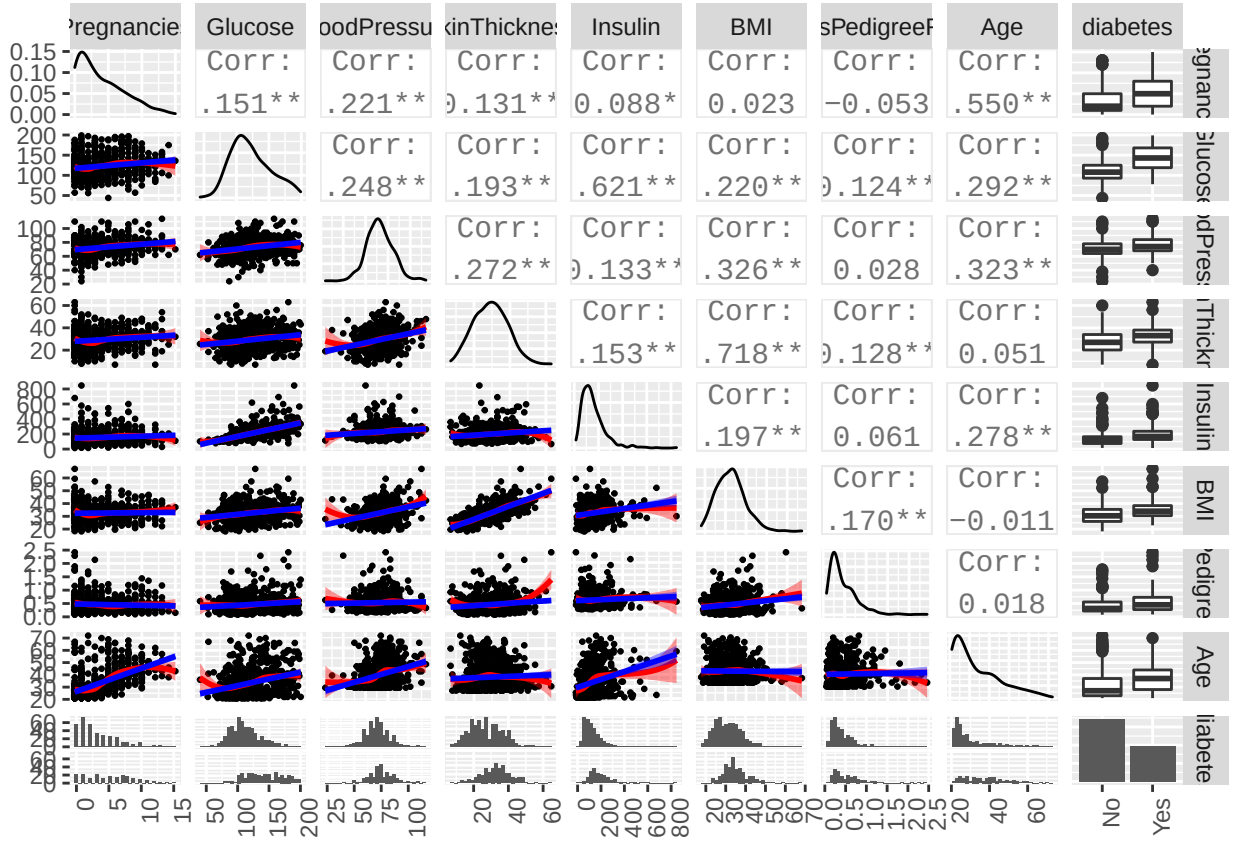

```
# Matriz completa com dispersão, densidades, correlações, retas de regressão (com IC) e LOESS (com IC)

      # Função auxiliar para curvas de regressão linear e LOESS nos gráficos abaixo
      curvas <- function(data, mapping, ...){
        p <- ggplot(data = data, mapping = mapping) +
          geom_point(size = 0.5) +
          geom_smooth(method = loess, fill = "red", color = "red", ...) +
          geom_smooth(method = lm, fill = "blue", color = "blue", ...)
        p
      }

ggpairs(train, columns = 1:9, lower = list(continuous = curvas)) + # Obs: pode demorar para montar o gr
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## `geom_smooth()` using formula 'y ~ x'
```

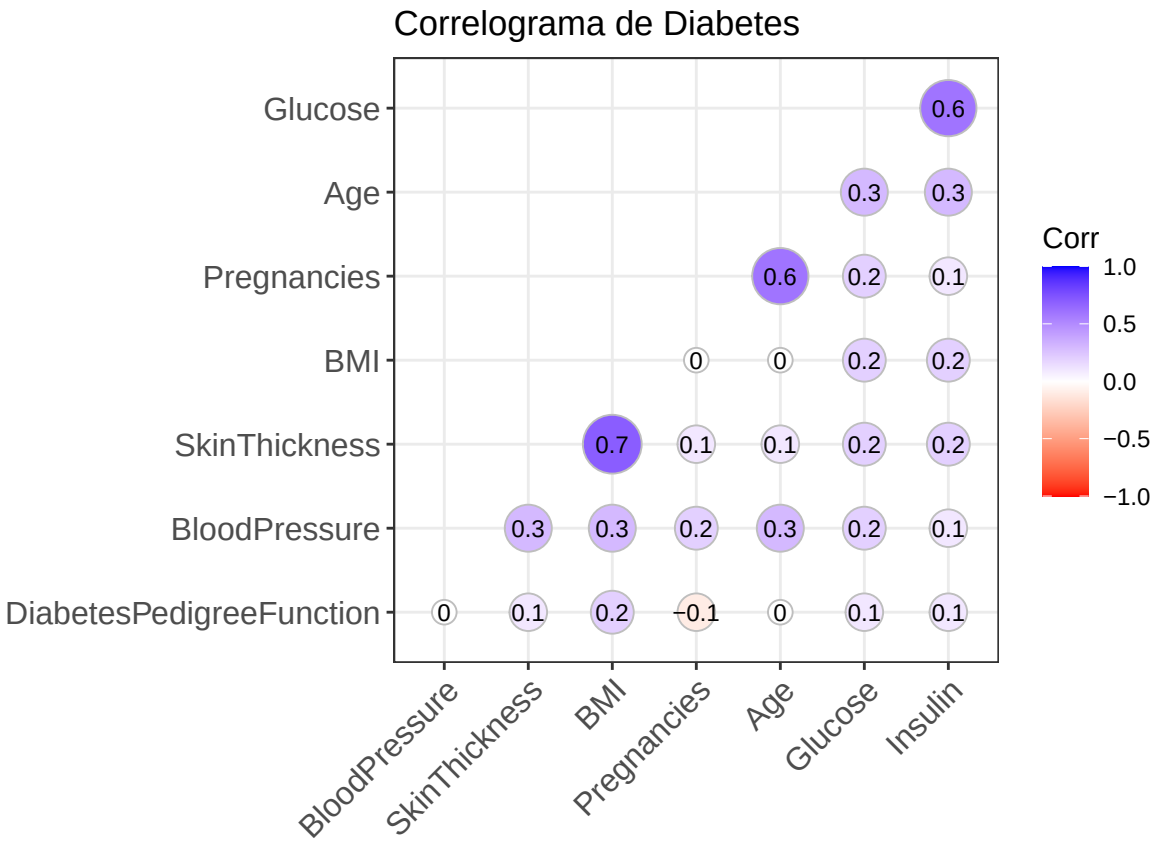
```
## `geom_smooth()` using formula 'y ~ x'
```

```
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
## ‘geom_smooth()‘ using formula ’y ~ x’
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

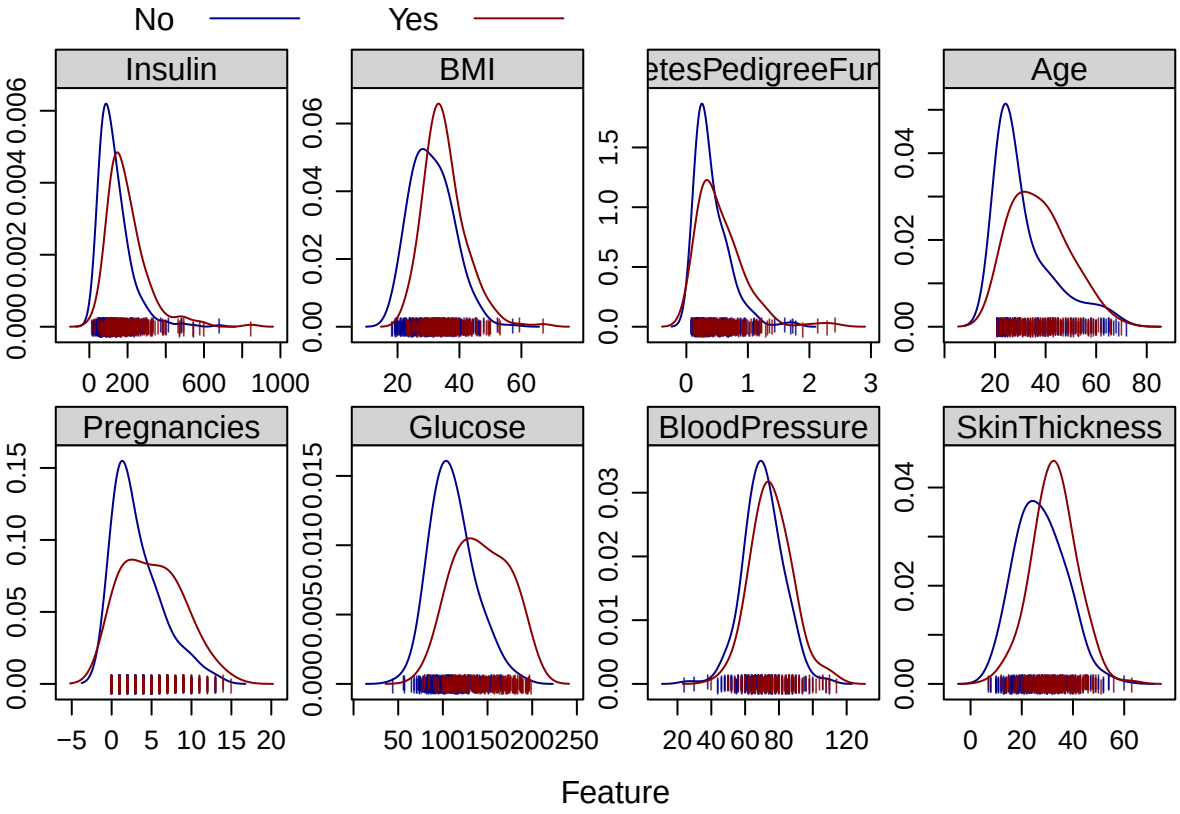



```r
# Correlações/Correlograma
corr<-round(cor(train[,-9]),1)
ggcorrplot(corr, hc.order = TRUE,
           type = "lower",
           lab = TRUE,
           lab_size = 3,
           method="circle",
           colors = c("red", "white", "blue"),
           title="Correlograma de Diabetes",
           ggtheme=theme_bw)
```

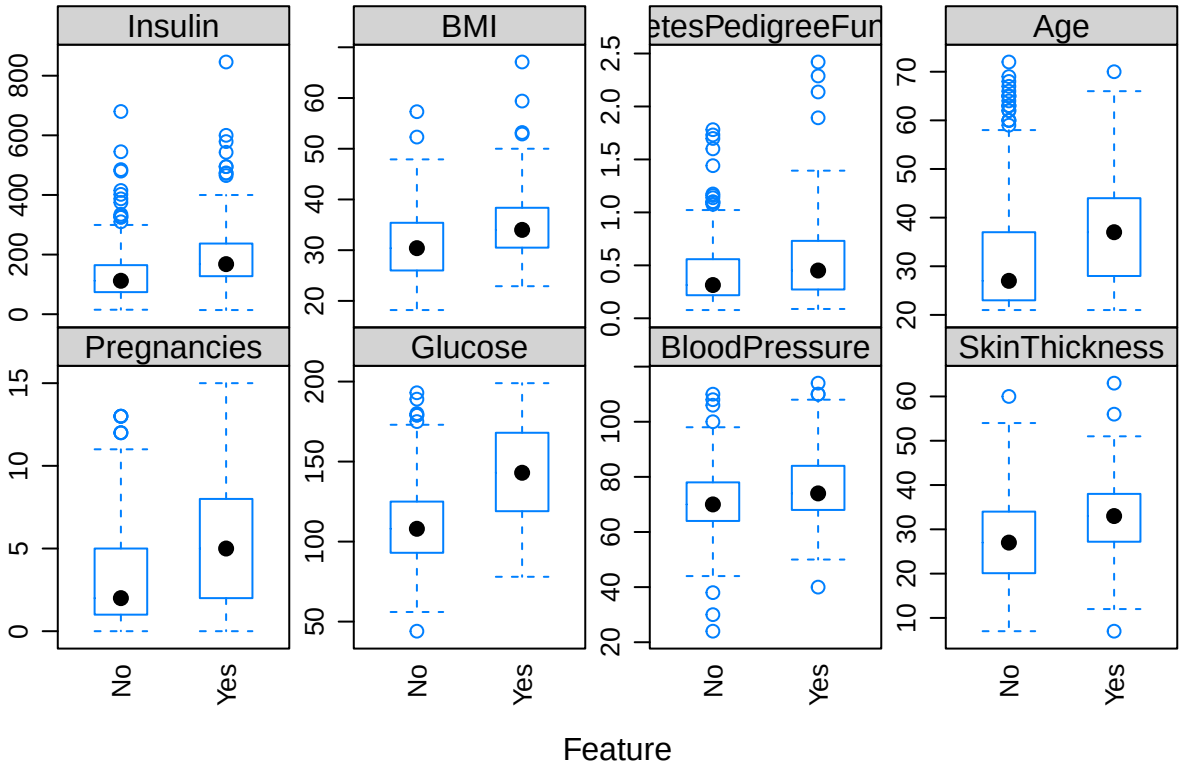



```
# Distribuição da diabetes Yes/No por feature

featurePlot(x = train[, 1:8], # Densidades alisadas
            y = train$diabetes,
            plot = "density",
            scales = list(x = list(relation="free"), y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
            layout = c(4, 2),
            auto.key = list(columns = 4),
            par.settings = list(strip.background=list(col="lightgrey"),
            superpose.line = list(col = c("darkblue","darkred")),
            superpose.symbol = list(col = c("darkblue","darkred")) ))
```

```
## Warning in draw.key(simpleKey(...), draw = FALSE): not enough rows for columns
```




```
featurePlot(x = train[, 1:8], # Boxplots
            y = train$diabetes,
            plot = "box",
            scales = list(y = list(relation="free"), x = list(rot = 90)),
            layout = c(4,2 ),
            auto.key = list(columns = 4),
            par.settings = list(strip.background=list(col="lightgrey")))
```



Funções auxiliares

```
library(dplyr)
library(ggraph)
```

```
## Warning: package 'ggraph' was built under R version 4.0.3
```

```
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.0.2
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tidyr':
##
##     crossing


## The following object is masked from 'package:tibble':
##
##     as_data_frame


## The following objects are masked from 'package:stats':
##
##     decompose, spectrum


## The following object is masked from 'package:base':
##
##     union
```

```r
plotaroc <- function(rocobj, titulo = "Curva ROC"){
  # Função que plota as curvas roc para os modelos ajustados
  b <- which.max(rocobj$sensitivities + rocobj$specificities)
  best <- round(c(rocobj$thresholds[b],rocobj$specificities[b],rocobj$sensitivities[b]), 3)

  pROC::ggroc(rocobj, col = "red", alpha = 0.5, size = 0.5) +
    theme_gray() +
    ggtitle(titulo) +
    geom_abline(intercept = 1, slope=1, linetype = "dashed") +
    labs(x="Especificidade", y = "Sensibilidade")  +
    geom_point(data = tibble(Sensibilidade = best[2],
                             Especificidade = best[3]),
               mapping = aes(x=Sensibilidade, y=Especificidade),
               col = "black") +
    geom_text(mapping =  aes(x = best[2] - 0.15,
                             y = best[3] - 0.05),
              label = paste( best[1], "(", best[2], ",", best[3], ")")) +
    geom_text(mapping = aes(x = 0.5,
                            y = 0.01),
              label = paste("AUC: ", round(rocobj$auc,3)))
}

tree_func <- function(final_model,
                      tree_num) {

  # get tree by index
  tree <- randomForest::getTree(final_model,
                                k = tree_num,
                                labelVar = TRUE) %>%
    tibble::rownames_to_column() %>%
    # make leaf split points to NA, so the 0s won't get plotted
    mutate(`split point` = ifelse(is.na(prediction), `split point`, NA))

  # prepare data frame for graph
  graph_frame <- data.frame(from = rep(tree$rowname, 2),
                            to = c(tree$`left daughter`, tree$`right daughter`))
```

```
  # convert to graph and delete the last node that we don't want to plot
  graph <- graph_from_data_frame(graph_frame) %>%
    delete_vertices("0")

  # set node labels
  V(graph)$node_label <- gsub("_", " ", as.character(tree$`split var`))
  V(graph)$leaf_label <- as.character(tree$prediction)
  V(graph)$split <- as.character(round(tree$`split point`, digits = 2))

  # plot
  plot <- ggraph(graph, 'dendrogram') +
    theme_bw() +
    geom_edge_link() +
    geom_node_point() +
    geom_node_text(aes(label = node_label), na.rm = TRUE, repel = TRUE) +
    geom_node_label(aes(label = split), vjust = 2, na.rm = TRUE, fill = "white") +
    geom_node_label(aes(label = leaf_label, fill = leaf_label), na.rm = TRUE,
                    repel = TRUE, colour = "white", fontface = "bold", show.legend = FALSE) +
    theme(panel.grid.minor = element_blank(),
          panel.grid.major = element_blank(),
          panel.background = element_blank(),
          plot.background = element_rect(fill = "white"),
          panel.border = element_blank(),
          axis.line = element_blank(),
          axis.text.x = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks = element_blank(),
          axis.title.x = element_blank(),
          axis.title.y = element_blank(),
          plot.title = element_text(size = 14))

  return(plot)
}
```

Análise Discriminante

Modelagem

```
# Obs: possibilidades de modelos de AD: rda, lda, pda (acurácias iguais) e qda (pior)

train.control <- caret::trainControl(method = "cv", number = 15, classProbs =  TRUE) # Cross-validation

## ADL com dados imputados
set.seed(23)
modeloAD1 <- caret::train(diabetes ~ ., data = train, trControl = train.control, method = "lda")
print(modeloAD1)
```

```
## Linear Discriminant Analysis
```

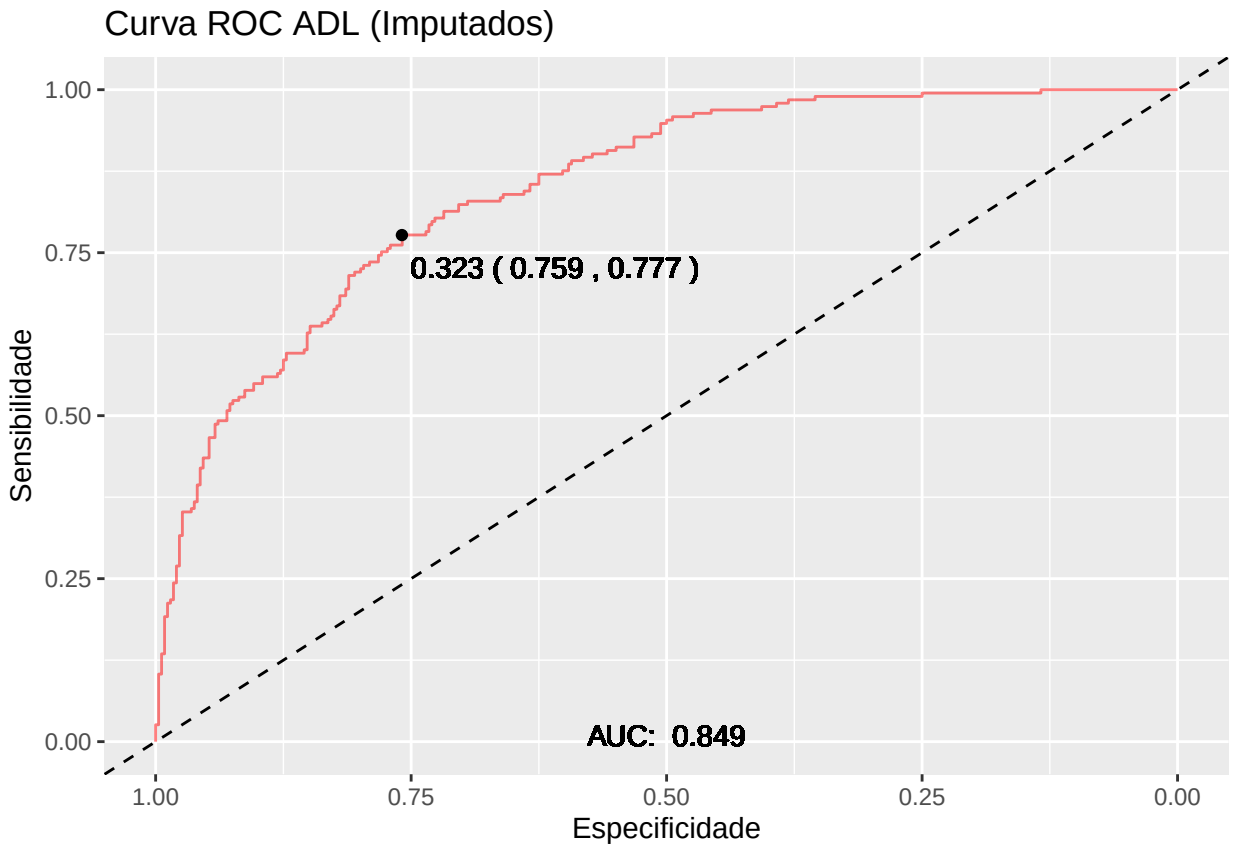
```
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (15 fold)
## Summary of sample sizes: 501, 501, 501, 501, 501, 501, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7636508  0.4650271
```

```
rocAD1 <- roc(response = train$diabetes, predictor = predict(modeloAD1, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocAD1, titulo = "Curva ROC ADL (Imputados)")
```



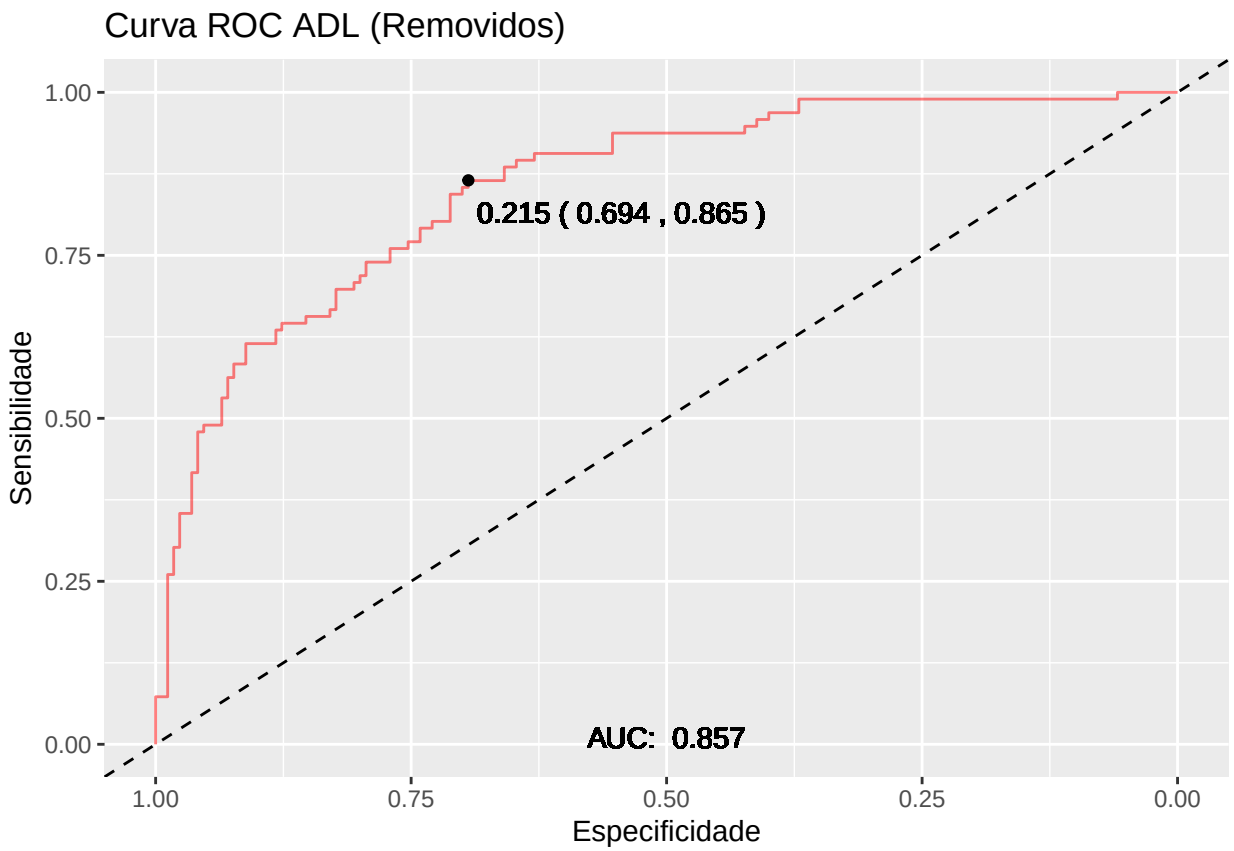
```r
## ADL sem missing
set.seed(23)
modeloAD2 <- caret::train(diabetes ~ ., data = train_without_NAs, trControl = train.control, method = "
print(modeloAD2)
```

```
## Linear Discriminant Analysis
##
## 266 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (15 fold)
## Summary of sample sizes: 248, 248, 249, 248, 247, 249, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7784199  0.5053595
```

```r
rocAD2 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloAD2, train_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```r
plotaroc(rocAD2, titulo = "Curva ROC ADL (Removidos)")
```




```
## AD Flexível com dados imputados
set.seed(23)
modeloAD3 <- caret::train(diabetes ~ ., data = train, trControl = train.control, method = "fda")
```

```
## Loading required package: earth
```

```
## Warning: package 'earth' was built under R version 4.0.3
```

```
## Loading required package: plotmo
```

```
## Warning: package 'plotmo' was built under R version 4.0.3
```

```
## Loading required package: plotrix
```

```
##
## Attaching package: 'plotrix'
```

```
## The following object is masked from 'package:scales':
##
##     rescale
```

```
## The following object is masked from 'package:psych':
##
##     rescale
```

```
## Loading required package: TeachingDemos
```

```
## Warning: package 'TeachingDemos' was built under R version 4.0.3
```

```
##
## Attaching package: 'TeachingDemos'
```

```
## The following objects are masked from 'package:Hmisc':
##
##     cnvrt.coords, subplot
```

```
print(modeloAD3)
```

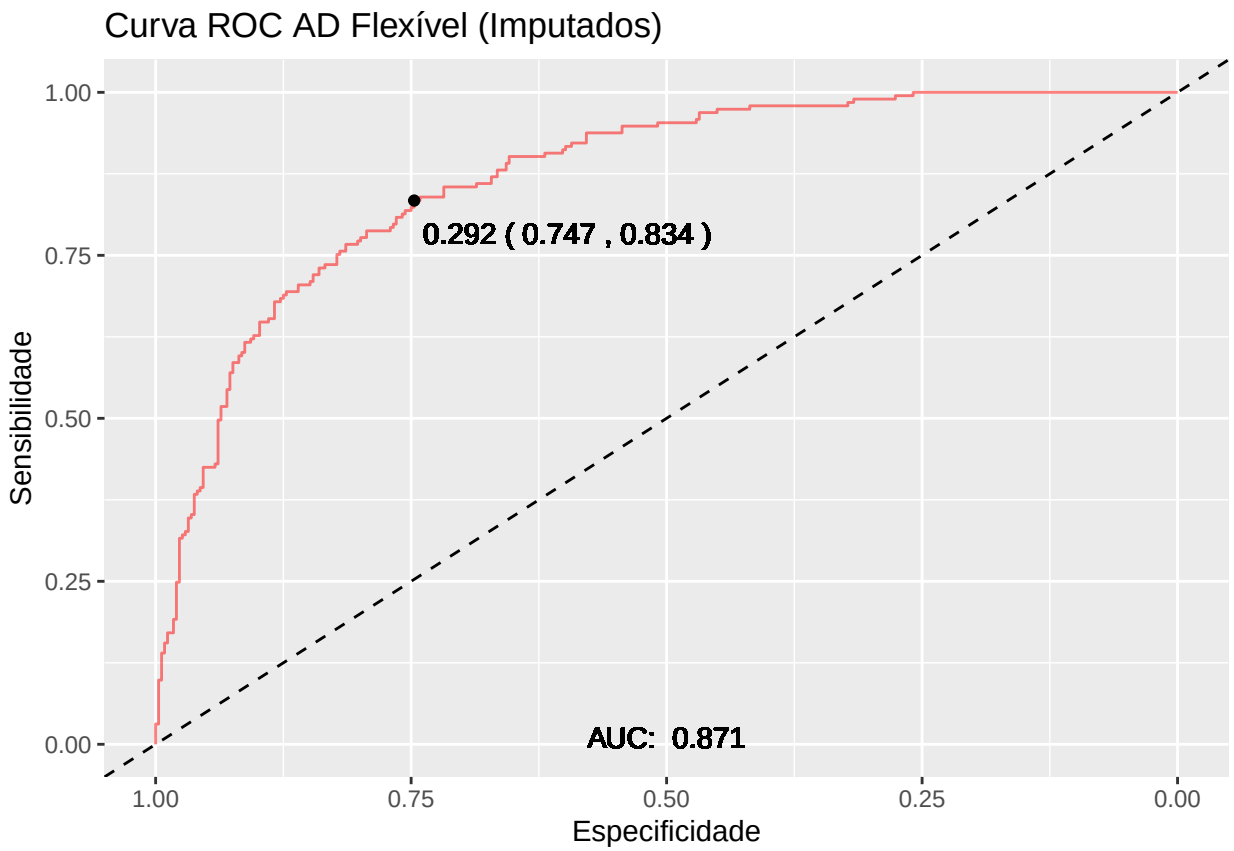
```
## Flexible Discriminant Analysis
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (15 fold)
## Summary of sample sizes: 501, 501, 501, 501, 501, 501, ...
## Resampling results across tuning parameters:
##
##   nprune  Accuracy   Kappa
##    2      0.7450794  0.4129759
##    8      0.7823280  0.5204850
##   15      0.7823280  0.5192155
##
## Tuning parameter 'degree' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 1 and nprune = 8.
```

```
rocAD3 <- roc(response = train$diabetes, predictor = predict(modeloAD3, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocAD3, titulo = "Curva ROC AD Flexível (Imputados)")
```



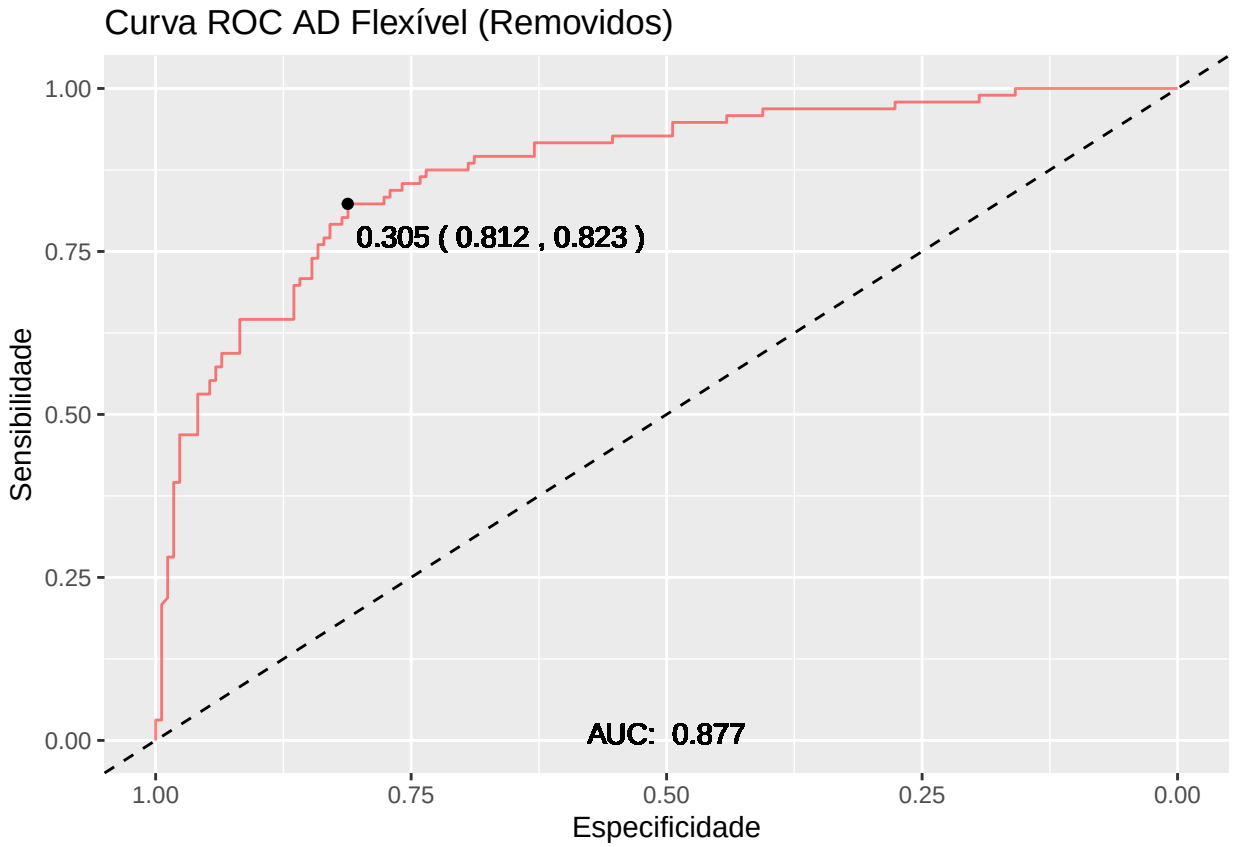

```
## AD Flexível sem missing
set.seed(23)
modeloAD4 <- caret::train(diabetes ~ ., data = train_without_NAs, trControl = train.control, method = "
print(modeloAD4)
```

```
## Flexible Discriminant Analysis
##
## 266 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (15 fold)
## Summary of sample sizes: 248, 248, 249, 248, 247, 249, ...
## Resampling results across tuning parameters:
##
##   nprune  Accuracy   Kappa
##    2      0.7481137  0.4233678
##    8      0.7516455  0.4422724
##   15      0.7446279  0.4240573
##
## Tuning parameter 'degree' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were degree = 1 and nprune = 8.
```

```
rocAD4 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloAD4, train_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD4, titulo = "Curva ROC AD Flexível (Removidos)")
```




```
## AD Quadrática com dados imputados
set.seed(23)
modeloAD5 <- caret::train(diabetes ~ ., data = train, trControl = train.control, method = "qda")
print(modeloAD5)
```

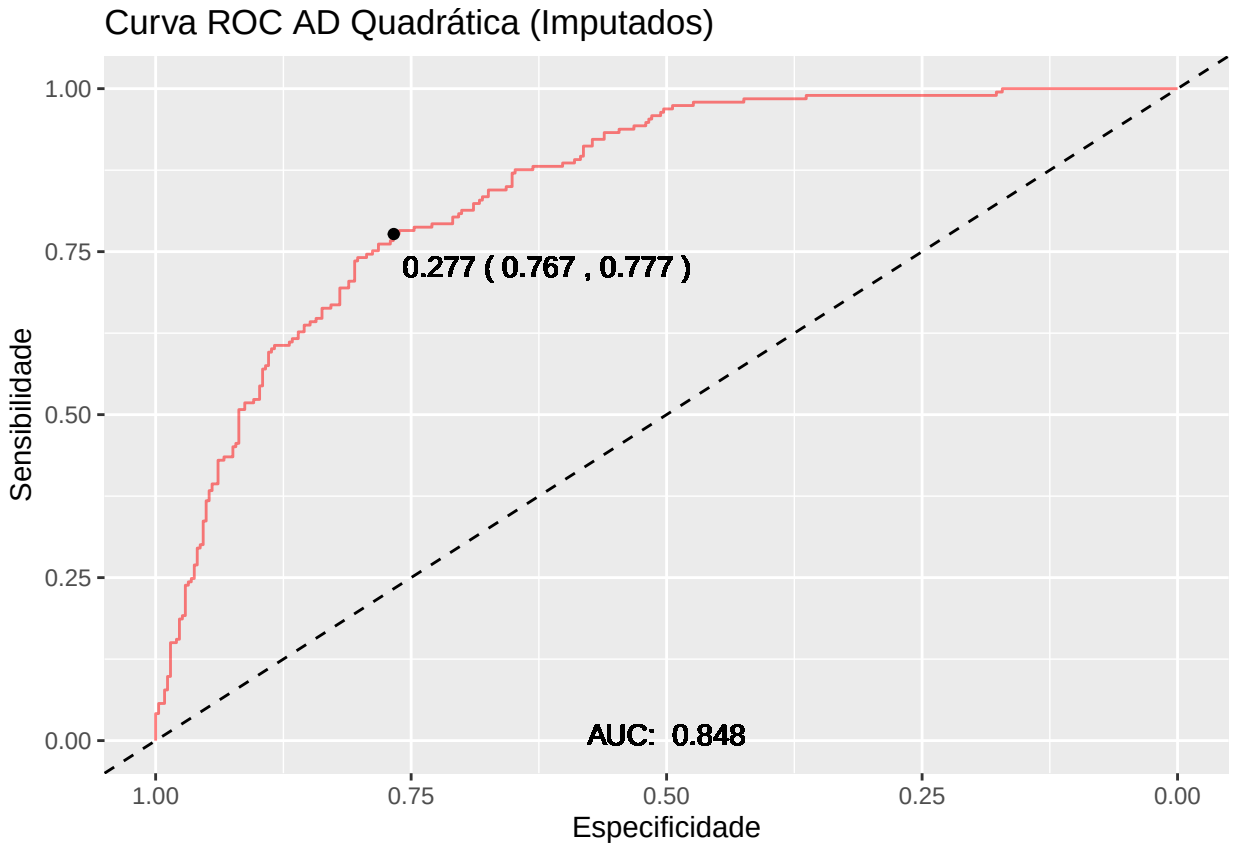
```
## Quadratic Discriminant Analysis
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (15 fold)
## Summary of sample sizes: 501, 501, 501, 501, 501, 501, ...
## Resampling results:
##
```

```
##   Accuracy   Kappa
##   0.7431217  0.4201363
```

```
rocAD5 <- roc(response = train$diabetes, predictor = predict(modeloAD5, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD5, titulo = "Curva ROC AD Quadrática (Imputados)")
```




```
## AD Quadrática sem missing
set.seed(23)
modeloAD6 <- caret::train(diabetes ~ ., data = train_without_NAs, trControl = train.control, method = "
print(modeloAD6)
```

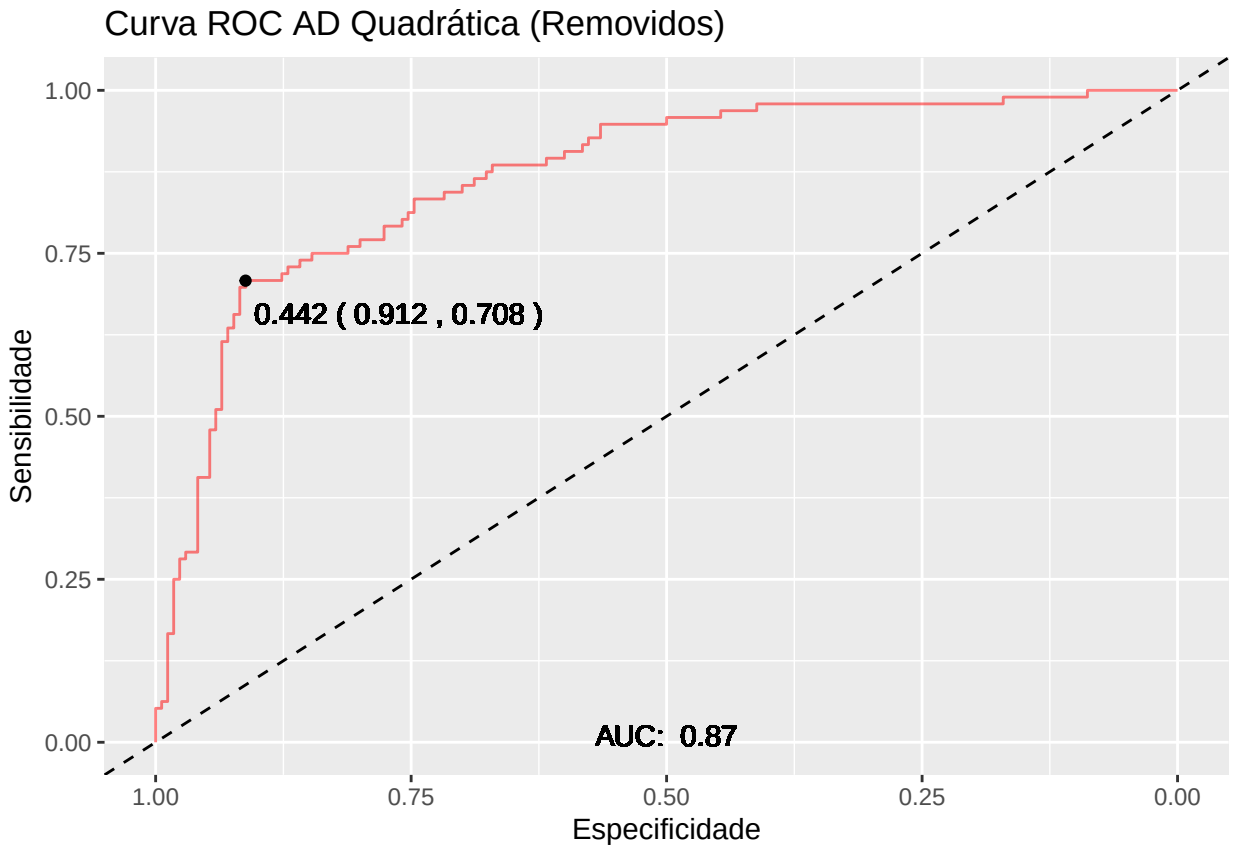
```
## Quadratic Discriminant Analysis
##
## 266 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (15 fold)
```

```
## Summary of sample sizes: 248, 248, 249, 248, 247, 249, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7860681  0.5158632
```

```
rocAD6 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloAD6, train_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD6, titulo = "Curva ROC AD Quadrática (Removidos)")
```



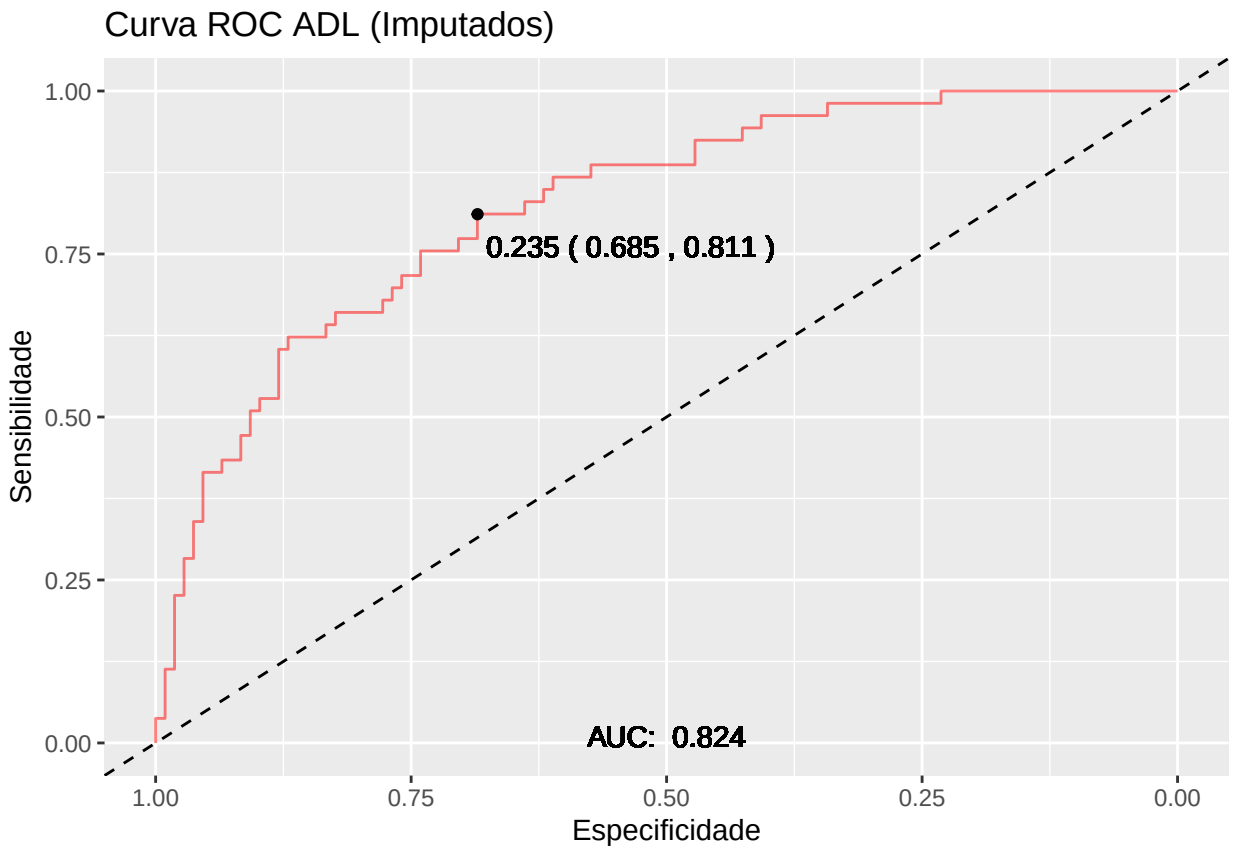
Testes

```
## ADL com dados imputados
rocAD1 <- roc(response = test$diabetes, predictor = predict(modeloAD1, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```r
plotaroc(rocAD1, titulo = "Curva ROC ADL (Imputados)")
```




```r
predictAD1 <- predict(modeloAD1, newdata = test)
confusionMatrix(predictAD1, test$diabetes)
```

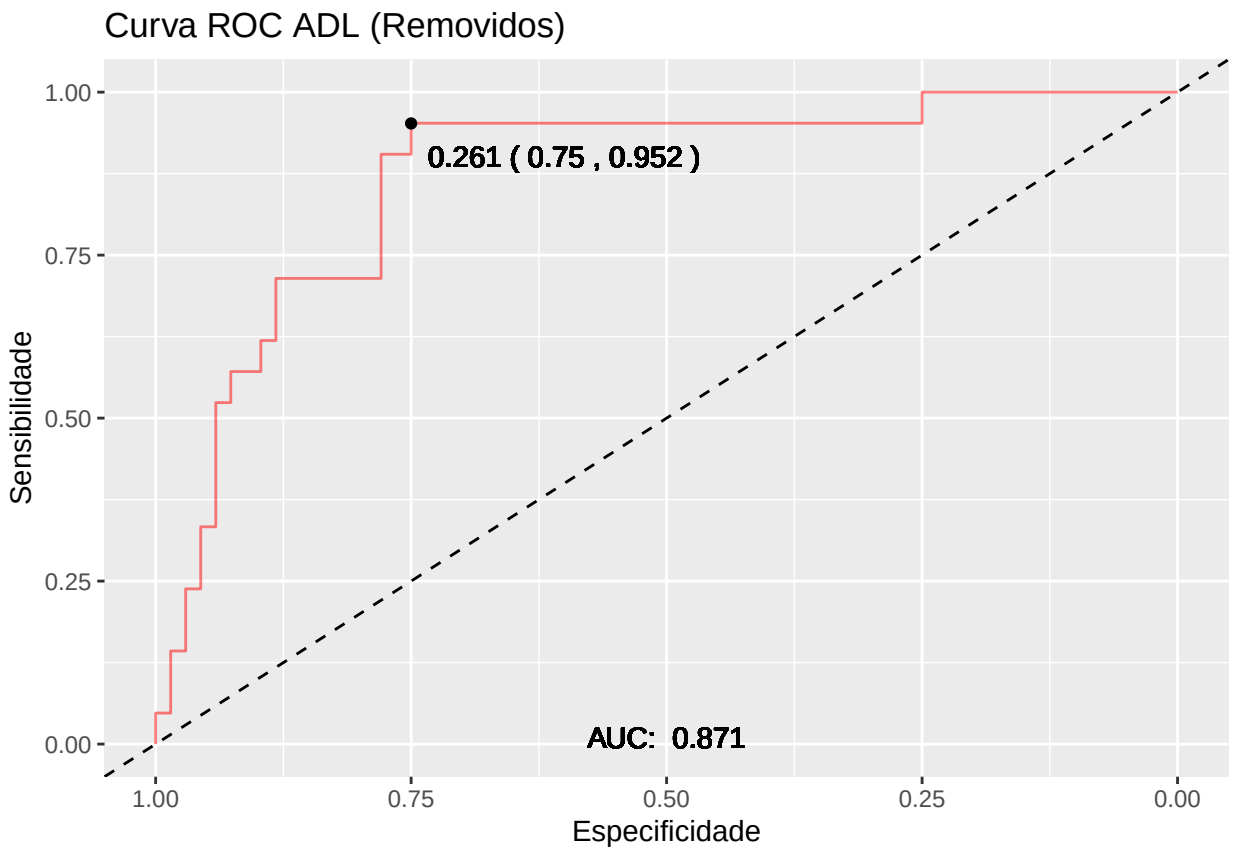
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  99  28
##        Yes  9  25
##
##                Accuracy : 0.7702
##                  95% CI : (0.6974, 0.8327)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.003815
##
##                   Kappa : 0.4274
##
##  Mcnemar's Test P-Value : 0.003085
##
##             Sensitivity : 0.9167
##             Specificity : 0.4717
##          Pos Pred Value : 0.7795
```

```
##          Neg Pred Value : 0.7353
##              Prevalence : 0.6708
##          Detection Rate : 0.6149
##    Detection Prevalence : 0.7888
##       Balanced Accuracy : 0.6942
##
##        'Positive' Class : No
##
```

```
## ADL sem missing
rocAD2 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloAD2, test_without_NAs, ty
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD2, titulo = "Curva ROC ADL (Removidos)")
```




```
predictAD2 <- predict(modeloAD2, newdata = test_without_NAs)
confusionMatrix(predictAD2, test_without_NAs$diabetes)
```

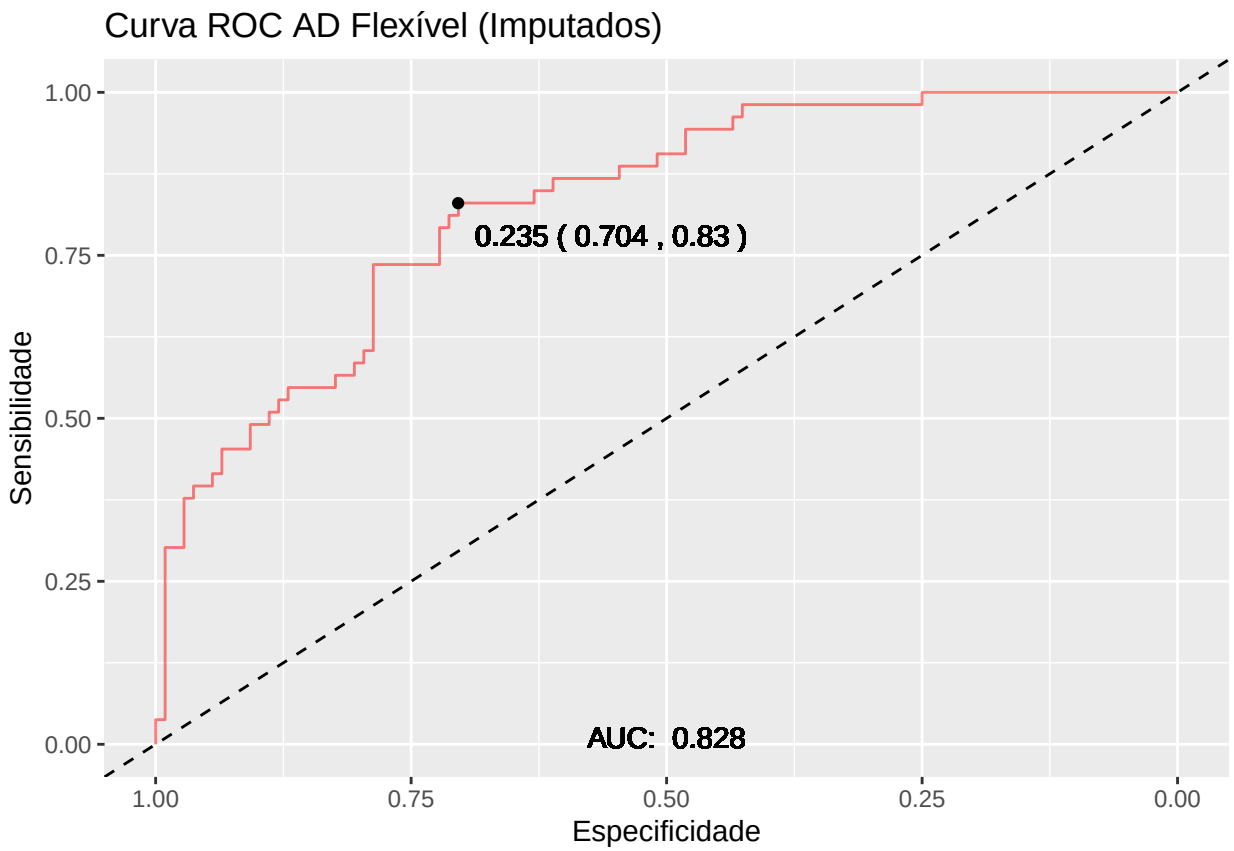
```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction No Yes
##        No  64  12
##        Yes  4   9
##
##                Accuracy : 0.8202
##                  95% CI : (0.7245, 0.8936)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.12910
##
##                   Kappa : 0.4258
##
##  Mcnemar's Test P-Value : 0.08012
##
##             Sensitivity : 0.9412
##             Specificity : 0.4286
##          Pos Pred Value : 0.8421
##          Neg Pred Value : 0.6923
##              Prevalence : 0.7640
##          Detection Rate : 0.7191
##    Detection Prevalence : 0.8539
##       Balanced Accuracy : 0.6849
##
##        'Positive' Class : No
##
```

```
## ADL Flexível com dados imputados
rocAD3 <- roc(response = test$diabetes, predictor = predict(modeloAD3, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD3, titulo = "Curva ROC AD Flexível (Imputados)")
```




```
predictAD3 <- predict(modeloAD3, newdata = test)
confusionMatrix(predictAD3, test$diabetes)
```

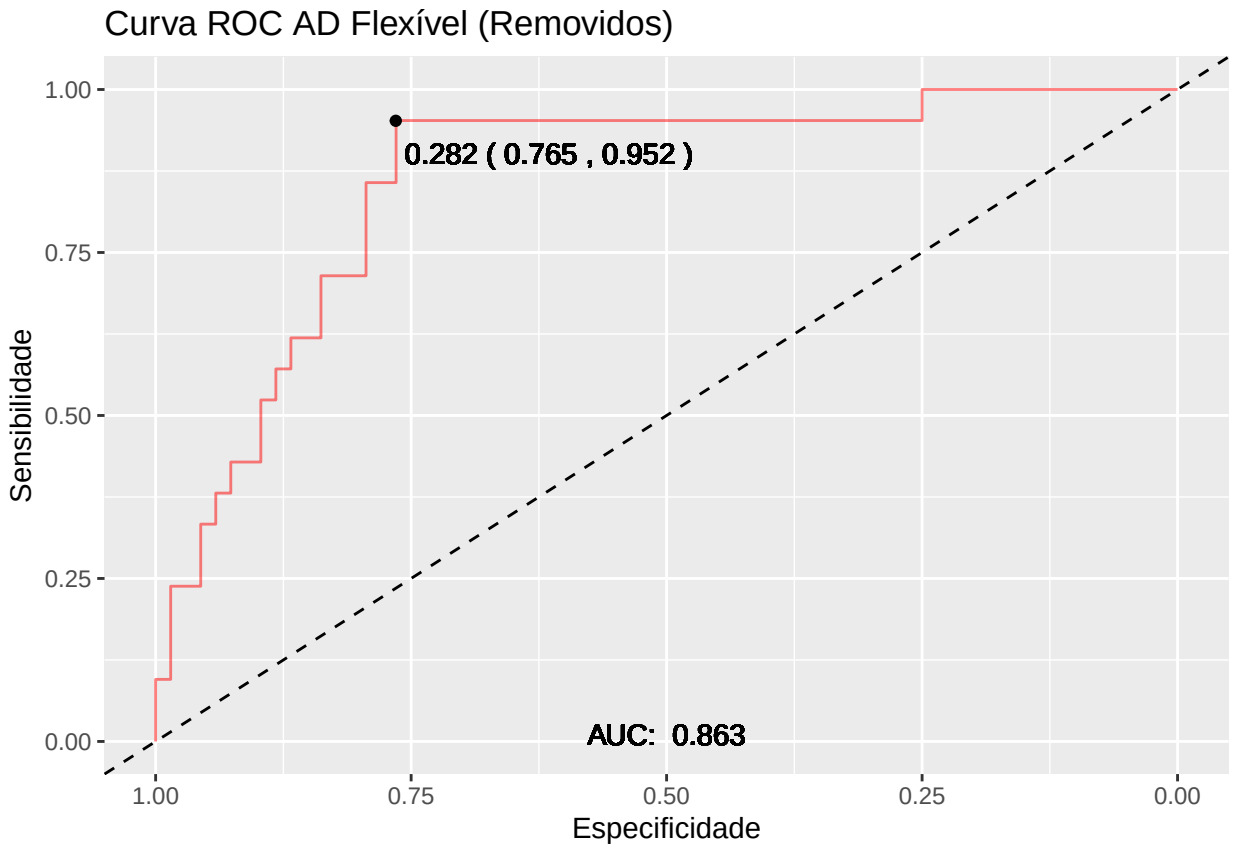
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  95  26
##        Yes 13  27
##
##                Accuracy : 0.7578
##                  95% CI : (0.6841, 0.8217)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.01039
##
##                   Kappa : 0.415
##
##  Mcnemar's Test P-Value : 0.05466
##
##             Sensitivity : 0.8796
##             Specificity : 0.5094
##          Pos Pred Value : 0.7851
##          Neg Pred Value : 0.6750
##              Prevalence : 0.6708
##          Detection Rate : 0.5901
```

```
##    Detection Prevalence : 0.7516
##       Balanced Accuracy : 0.6945
##
##        'Positive' Class : No
##
```

```
## ADL Flexível sem missing
rocAD4 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloAD4, test_without_NAs, ty
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD4, titulo = "Curva ROC AD Flexível (Removidos)")
```




```
predictAD4 <- predict(modeloAD4, newdata = test_without_NAs)
confusionMatrix(predictAD4, test_without_NAs$diabetes)
```

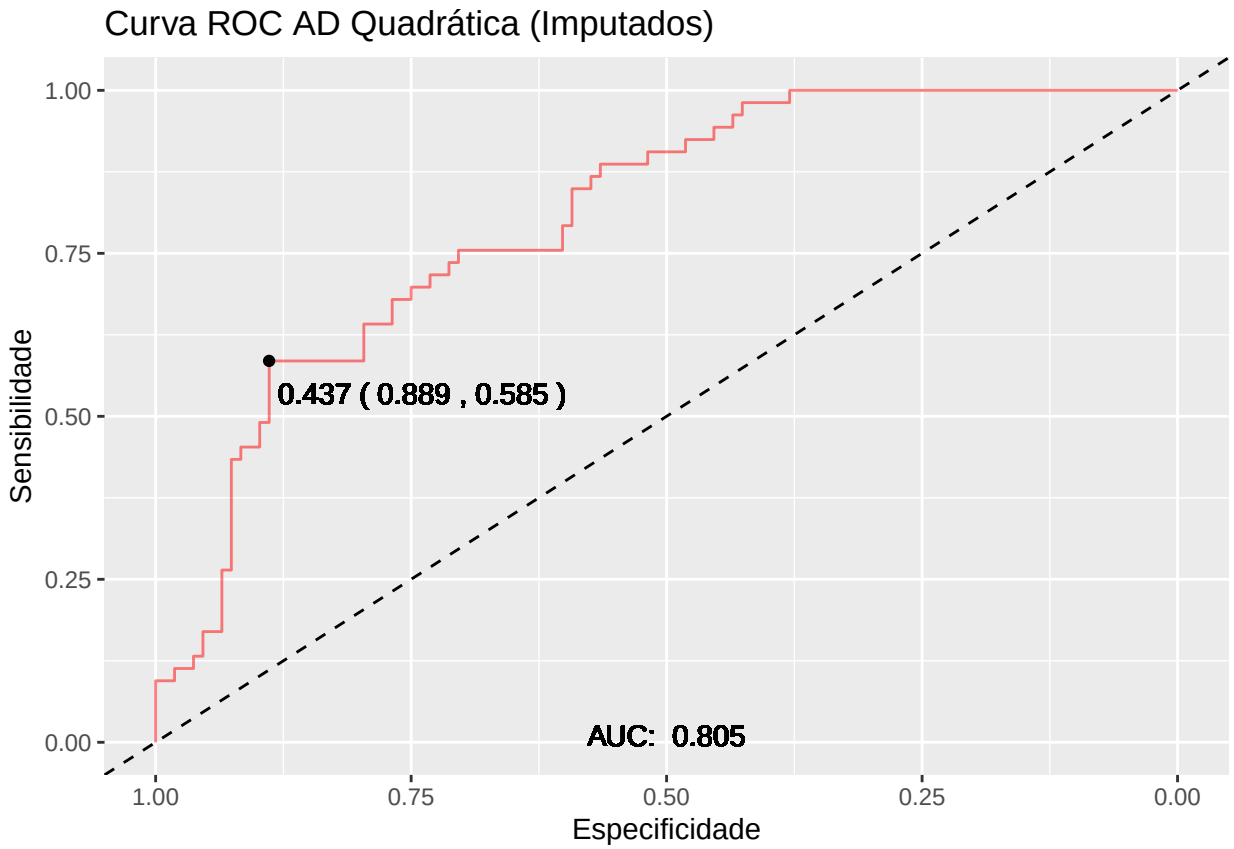
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  61  11
##        Yes  7  10
```

```
##
##                Accuracy : 0.7978
##                  95% CI : (0.6993, 0.8755)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.2709
##
##                   Kappa : 0.3996
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.8971
##             Specificity : 0.4762
##          Pos Pred Value : 0.8472
##          Neg Pred Value : 0.5882
##              Prevalence : 0.7640
##          Detection Rate : 0.6854
##    Detection Prevalence : 0.8090
##       Balanced Accuracy : 0.6866
##
##        'Positive' Class : No
##
```

```
## AD Quadrática com dados imputados
rocAD5 <- roc(response = test$diabetes, predictor = predict(modeloAD5, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD5, titulo = "Curva ROC AD Quadrática (Imputados)")
```




```
predictAD5 <- predict(modeloAD5, newdata = test)
confusionMatrix(predictAD5, test$diabetes)
```

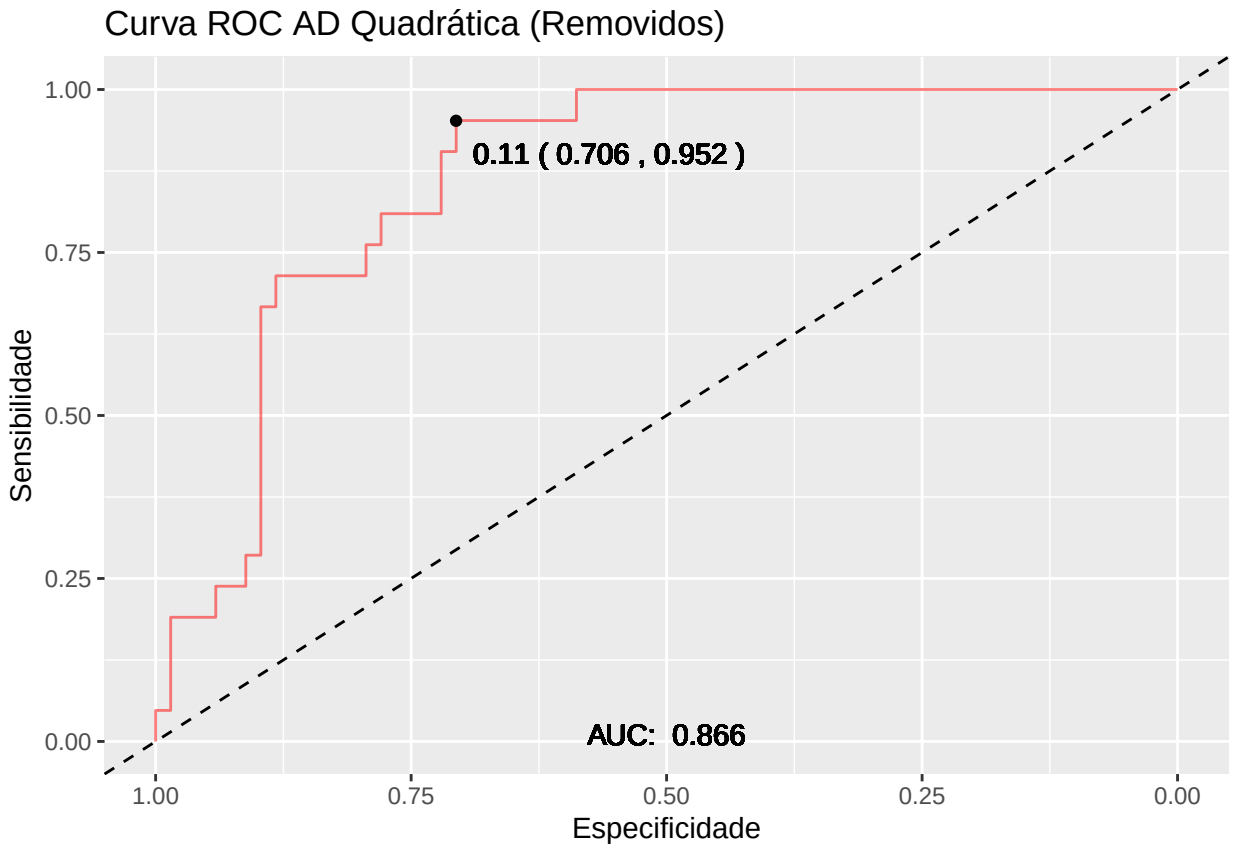
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  96  25
##        Yes 12  28
##
##                Accuracy : 0.7702
##                  95% CI : (0.6974, 0.8327)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.003815
##
##                   Kappa : 0.445
##
##  Mcnemar's Test P-Value : 0.048520
##
##             Sensitivity : 0.8889
##             Specificity : 0.5283
##          Pos Pred Value : 0.7934
##          Neg Pred Value : 0.7000
##              Prevalence : 0.6708
##          Detection Rate : 0.5963
```

```
##    Detection Prevalence : 0.7516
##       Balanced Accuracy : 0.7086
##
##        'Positive' Class : No
##
```

```
## AD Quadrática sem missing
rocAD6 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloAD6, test_without_NAs, ty
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocAD6, titulo = "Curva ROC AD Quadrática (Removidos)")
```




```
predictAD6 <- predict(modeloAD6, newdata = test_without_NAs)
confusionMatrix(predictAD6, test_without_NAs$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  60   6
##        Yes  8  15
```

```
##
##                Accuracy : 0.8427
##                  95% CI : (0.7502, 0.9112)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.04778
##
##                   Kappa : 0.5776
##
##  Mcnemar's Test P-Value : 0.78927
##
##             Sensitivity : 0.8824
##             Specificity : 0.7143
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.6522
##              Prevalence : 0.7640
##          Detection Rate : 0.6742
##    Detection Prevalence : 0.7416
##       Balanced Accuracy : 0.7983
##
##        'Positive' Class : No
##
```

Regressão Logística

Modelagem

```
library(caret)

train.control <- caret::trainControl(method = "cv", number = 10) # Cross-validation com k=10

## Logística simples com ligação logit / dados imputados
set.seed(23)
modeloRL1 <- caret::train(diabetes ~ ., data = train, trControl = train.control, method = "glm", family=
print(modeloRL1)
```

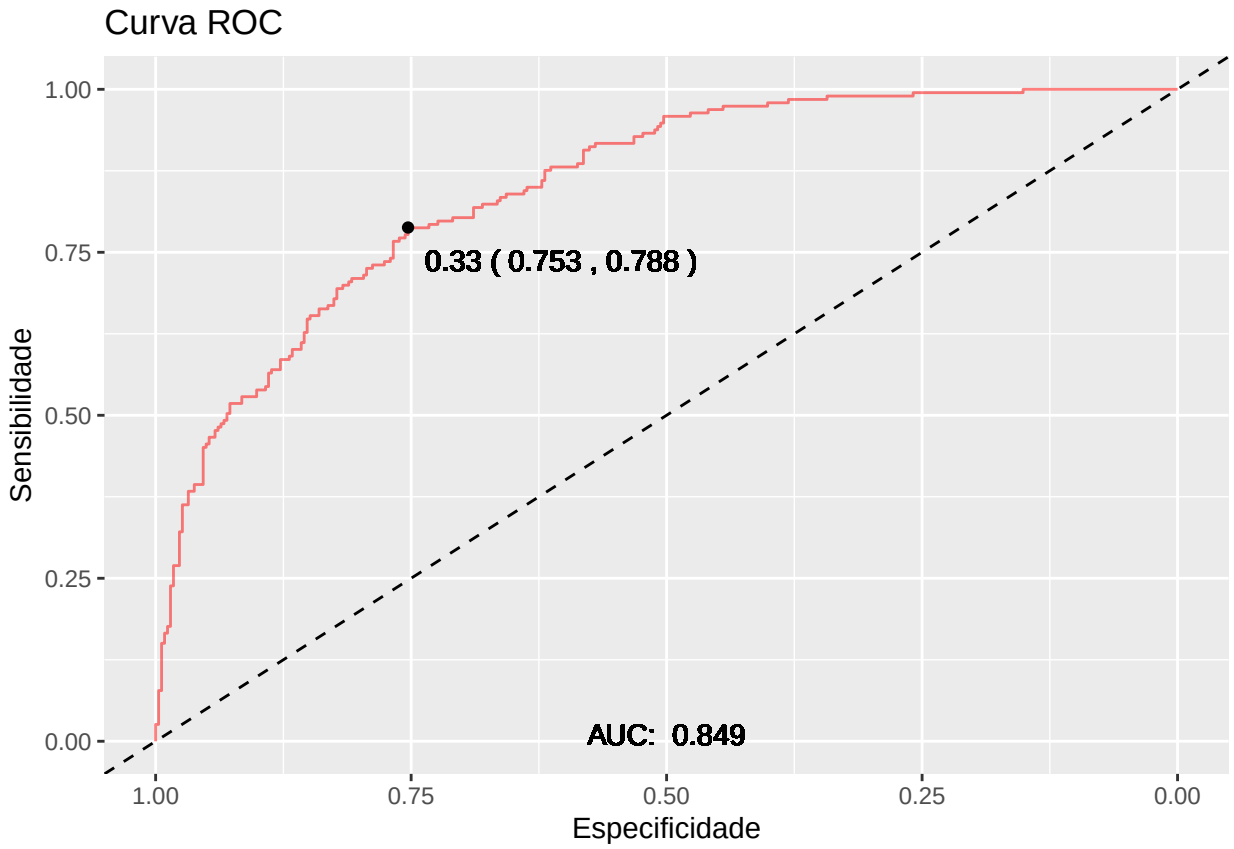
```
## Generalized Linear Model
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 483, 483, 484, 483, 482, 484, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7595623  0.4586088
```

```
rocRL1 <- roc(response = train$diabetes, predictor = predict(modeloRL1, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocRL1)
```




```
## Logística simples com ligação logit / dados sem missing
set.seed(23)
modeloRL2 <- caret::train(diabetes ~ ., data = train_without_NAs, trControl = train.control, method = "
print(modeloRL2)
```

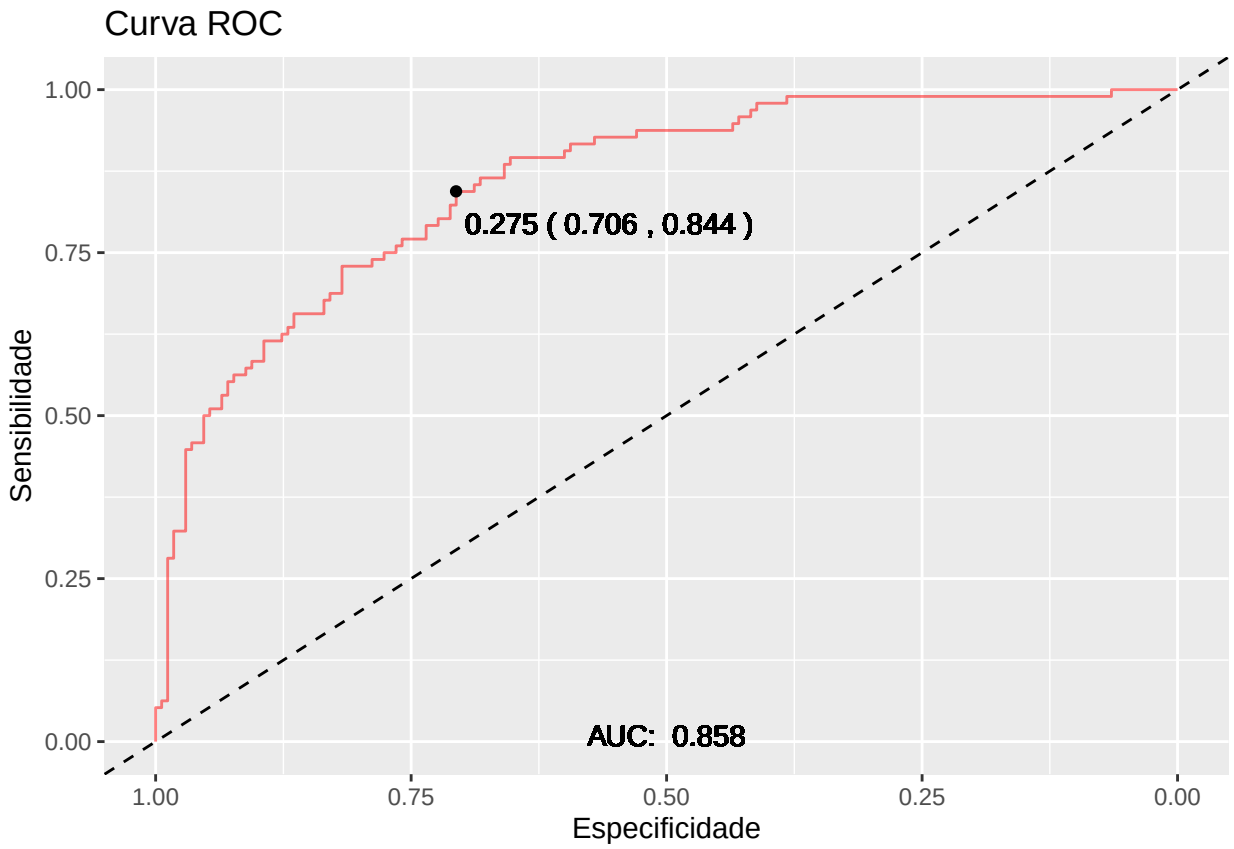
```
## Generalized Linear Model
##
## 266 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 240, 240, 239, 239, 239, 239, ...
```

```
## Resampling results:
##
##   Accuracy   Kappa
##   0.7673789  0.4828079
```

```
rocRL2 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloRL2, train_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRL2)
```




```
## Logística regularizada / dados imputados
set.seed(23)
modeloRL3 <- caret::train(diabetes ~ ., data = train, trControl = train.control, method = "regLogistic"
print(modeloRL3)
```

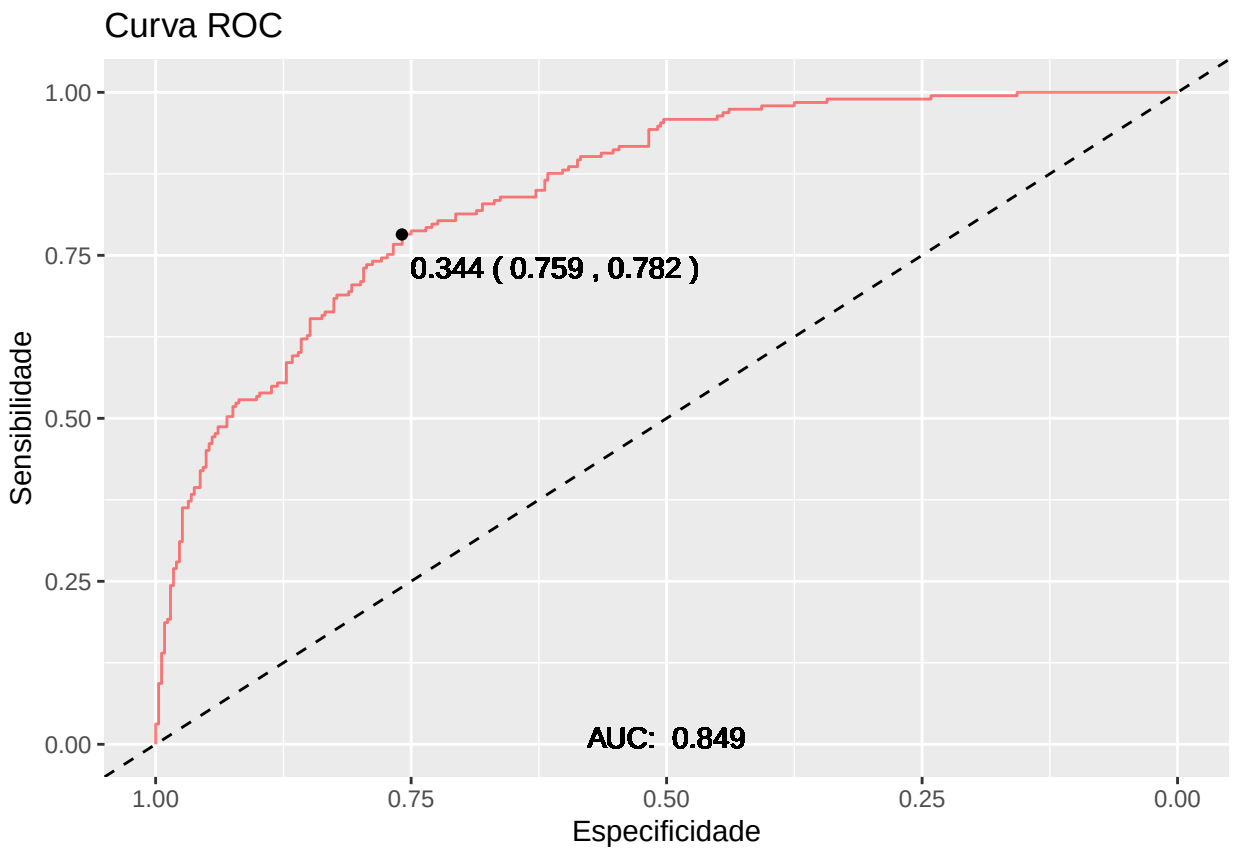
```
## Regularized Logistic Regression
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 483, 483, 484, 483, 482, 484, ...
## Resampling results across tuning parameters:
##
##   cost  loss       epsilon  Accuracy   Kappa
##   0.5   L1         0.001    0.7482765  0.4294592
##   0.5   L1         0.010    0.7426847  0.4156713
##   0.5   L1         0.100    0.7482053  0.4254680
##   0.5   L2_dual    0.001    0.6961737  0.2507155
##   0.5   L2_dual    0.010    0.6813925  0.2298081
##   0.5   L2_dual    0.100    0.6217813  0.1749702
##   0.5   L2_primal  0.001    0.7445715  0.4210244
##   0.5   L2_primal  0.010    0.6998126  0.3193385
##   0.5   L2_primal  0.100    0.6964195  0.3018360
##   1.0   L1         0.001    0.7558236  0.4469887
##   1.0   L1         0.010    0.7539369  0.4431172
##   1.0   L1         0.100    0.7426161  0.4055951
##   1.0   L2_dual    0.001    0.6039839  0.1296527
##   1.0   L2_dual    0.010    0.6144038  0.2065002
##   1.0   L2_dual    0.100    0.6311416  0.1155712
##   1.0   L2_primal  0.001    0.7464583  0.4229977
##   1.0   L2_primal  0.010    0.6998126  0.3193385
##   1.0   L2_primal  0.100    0.6964195  0.3018360
##   2.0   L1         0.001    0.7595623  0.4554919
##   2.0   L1         0.010    0.7558236  0.4468964
##   2.0   L1         0.100    0.7538670  0.4388071
##   2.0   L2_dual    0.001    0.6149863  0.1897020
##   2.0   L2_dual    0.010    0.5702999  0.1900379
##   2.0   L2_dual    0.100    0.5902649  0.1230814
##   2.0   L2_primal  0.001    0.7500934  0.4318992
##   2.0   L2_primal  0.010    0.7070802  0.3426133
##   2.0   L2_primal  0.100    0.6964195  0.3018360
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were cost = 2, loss = L1 and epsilon
##  = 0.001.
```

```
rocRL3 <- roc(response = train$diabetes, predictor = predict(modeloRL3, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRL3)
```




```
## Logística regularizada / dados sem missing
set.seed(23)
modeloRL4 <- caret::train(diabetes ~ ., data = train_without_NAs, trControl = train.control, method = "
print(modeloRL4)
```

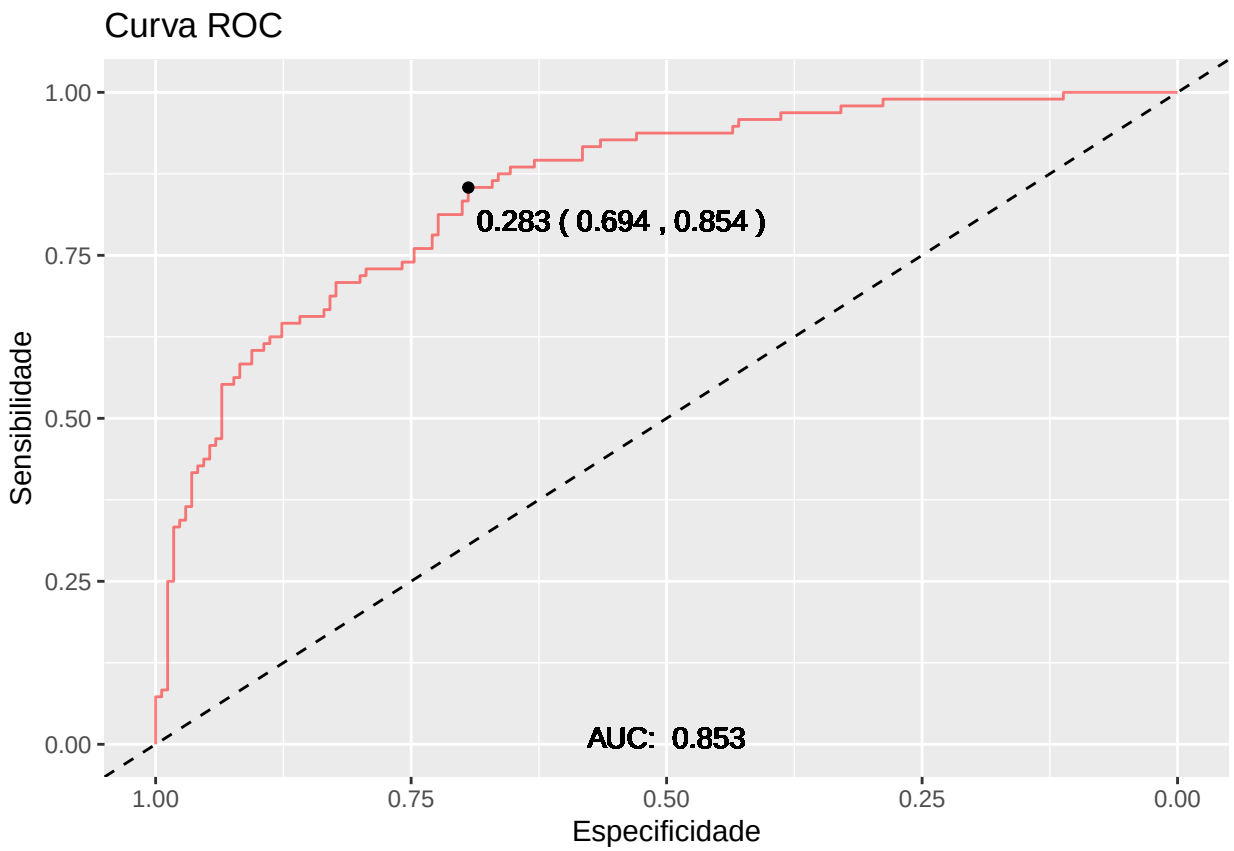
```
## Regularized Logistic Regression
##
## 266 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 240, 240, 239, 239, 239, 239, ...
## Resampling results across tuning parameters:
##
##   cost  loss       epsilon  Accuracy   Kappa
##   0.5   L1         0.001    0.7598291  0.4596795
##   0.5   L1         0.010    0.7636752  0.4669602
##   0.5   L1         0.100    0.7488604  0.4299137
##   0.5   L2_dual    0.001    0.6394587  0.1554546
##   0.5   L2_dual    0.010    0.6065527  0.1369186
##   0.5   L2_dual    0.100    0.6501425  0.1759711
##   0.5   L2_primal  0.001    0.7263533  0.3848223
##   0.5   L2_primal  0.010    0.7149573  0.3616567
```

```
##   0.5   L2_primal  0.100    0.6807692  0.2740010
##   1.0   L1         0.001    0.7709402  0.4834174
##   1.0   L1         0.010    0.7747863  0.4903988
##   1.0   L1         0.100    0.7522792  0.4352616
##   1.0   L2_dual    0.001    0.6249288  0.1842344
##   1.0   L2_dual    0.010    0.5915954  0.1786622
##   1.0   L2_dual    0.100    0.6173789  0.1770171
##   1.0   L2_primal  0.001    0.7565527  0.4527102
##   1.0   L2_primal  0.010    0.7339031  0.4037206
##   1.0   L2_primal  0.100    0.6807692  0.2740010
##   2.0   L1         0.001    0.7710826  0.4902196
##   2.0   L1         0.010    0.7747863  0.4970394
##   2.0   L1         0.100    0.7712251  0.4816945
##   2.0   L2_dual    0.001    0.5992877  0.1660801
##   2.0   L2_dual    0.010    0.6019943  0.2116899
##   2.0   L2_dual    0.100    0.6427350  0.1862283
##   2.0   L2_primal  0.001    0.7601140  0.4623563
##   2.0   L2_primal  0.010    0.7487179  0.4343284
##   2.0   L2_primal  0.100    0.6807692  0.2740010
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were cost = 1, loss = L1 and epsilon = 0.01.
```

```
rocRL4 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloRL4, train_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRL4)
```



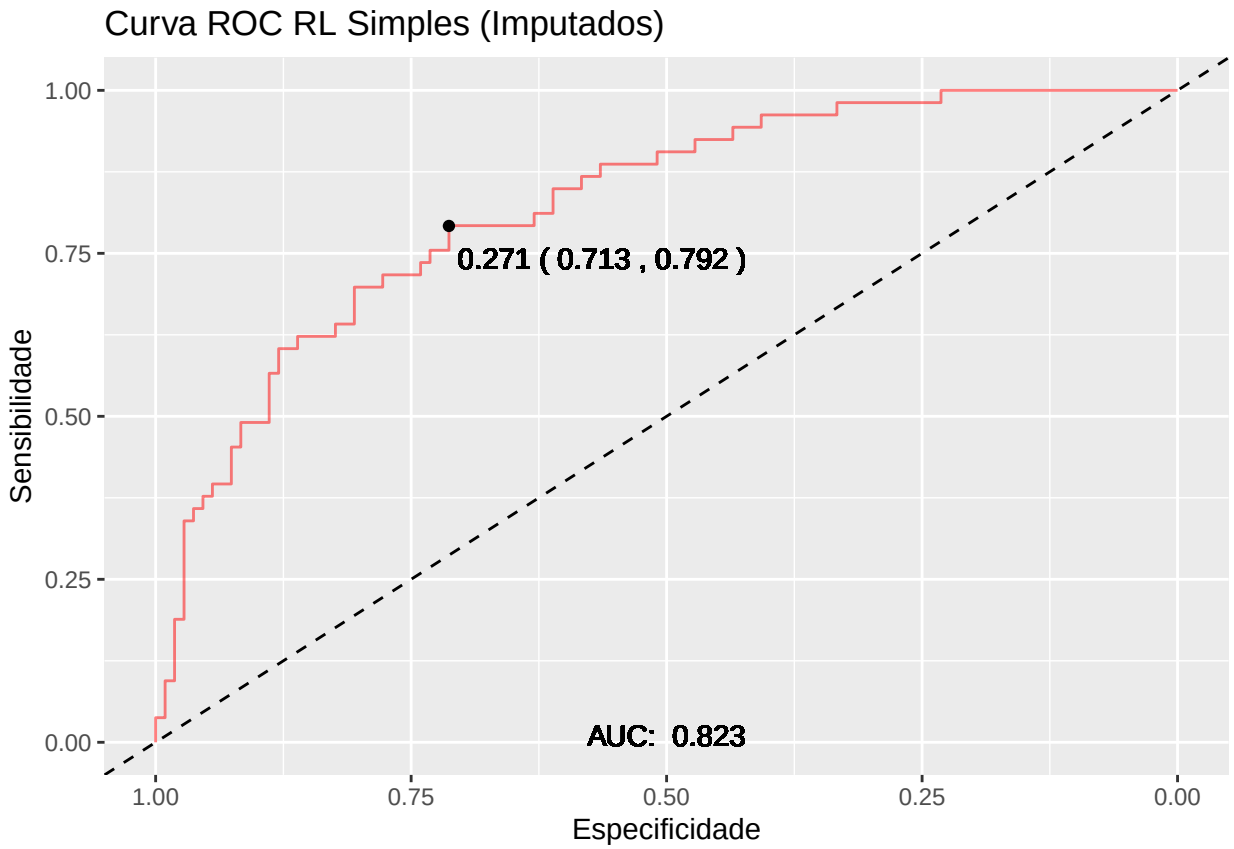
Teste

```
## Regressão Logística Simples com dados imputados
rocRL1 <- roc(response = test$diabetes, predictor = predict(modeloRL1, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocRL1, titulo = "Curva ROC RL Simples (Imputados)")
```




```
predictRL1 <- predict(modeloRL1, newdata = test)
confusionMatrix(predictRL1, test$diabetes)
```

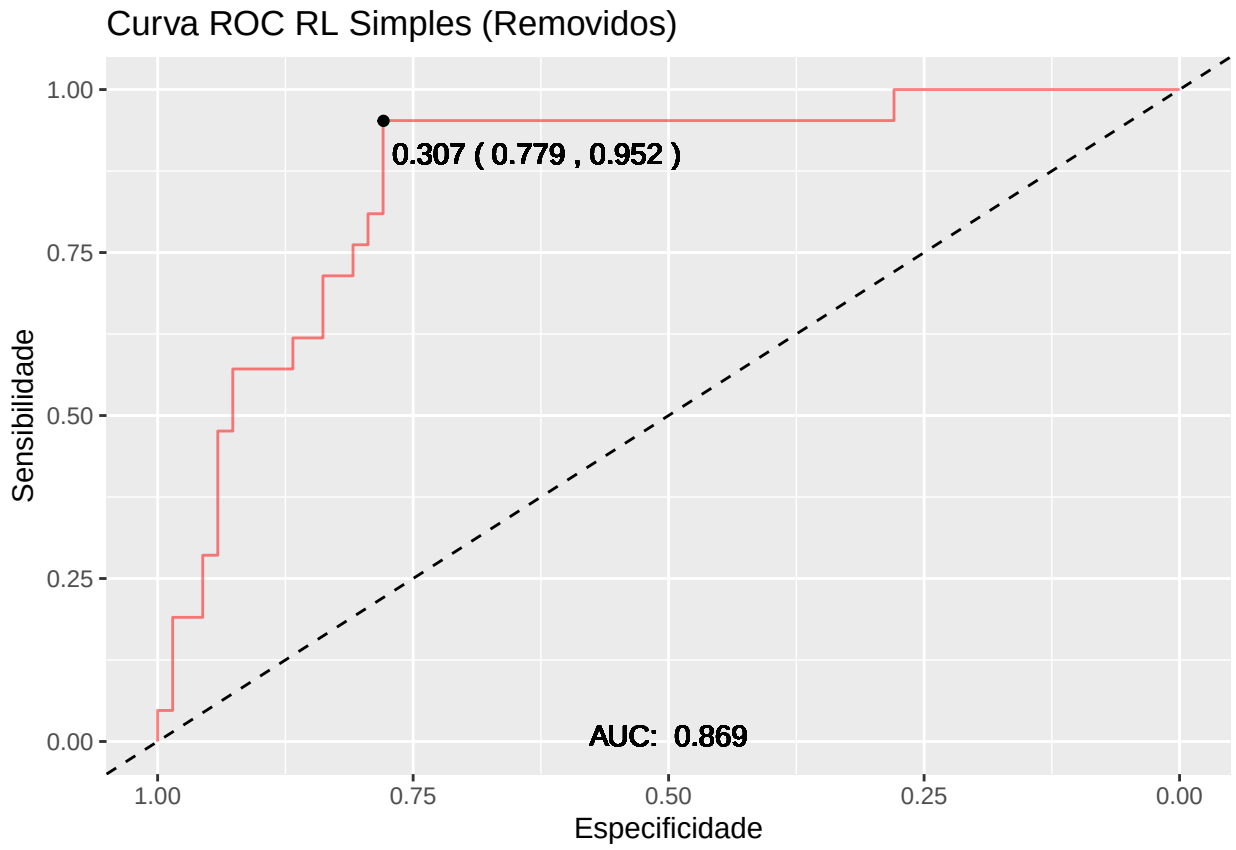
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  99  27
##        Yes  9  26
##
##                Accuracy : 0.7764
##                  95% CI : (0.7041, 0.8382)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.002203
##
##                   Kappa : 0.4458
##
##  Mcnemar's Test P-Value : 0.004607
##
##             Sensitivity : 0.9167
##             Specificity : 0.4906
##          Pos Pred Value : 0.7857
##          Neg Pred Value : 0.7429
##              Prevalence : 0.6708
##          Detection Rate : 0.6149
```

```
##    Detection Prevalence : 0.7826
##       Balanced Accuracy : 0.7036
##
##        'Positive' Class : No
##
```

```
## Regressão Logística Simples sem missing
rocRL2 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloRL2, test_without_NAs, ty
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRL2, titulo = "Curva ROC RL Simples (Removidos)")
```




```
predictRL2 <- predict(modeloRL2, newdata = test_without_NAs)
confusionMatrix(predictRL2, test_without_NAs$diabetes)
```

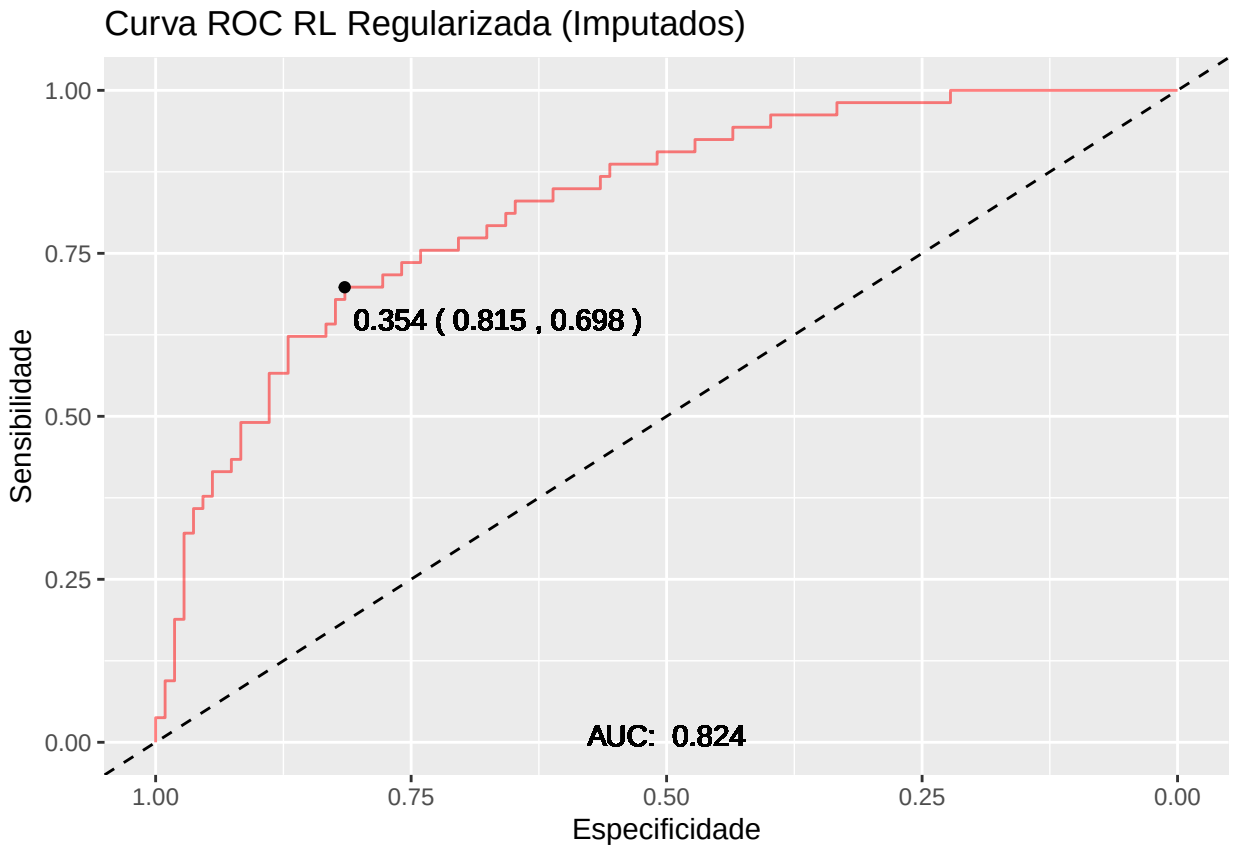
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  64  11
##        Yes  4  10
```

```
##
##                Accuracy : 0.8315
##                  95% CI : (0.7373, 0.9025)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.08127
##
##                   Kappa : 0.4717
##
##  Mcnemar's Test P-Value : 0.12134
##
##             Sensitivity : 0.9412
##             Specificity : 0.4762
##          Pos Pred Value : 0.8533
##          Neg Pred Value : 0.7143
##              Prevalence : 0.7640
##          Detection Rate : 0.7191
##    Detection Prevalence : 0.8427
##       Balanced Accuracy : 0.7087
##
##        'Positive' Class : No
##
```

```
## Regressão Logística Regularizada com dados imputados
rocRL3 <- roc(response = test$diabetes, predictor = predict(modeloRL3, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRL3, titulo = "Curva ROC RL Regularizada (Imputados)")
```




```
predictRL3 <- predict(modeloRL3, newdata = test)
confusionMatrix(predictRL3, test$diabetes)
```

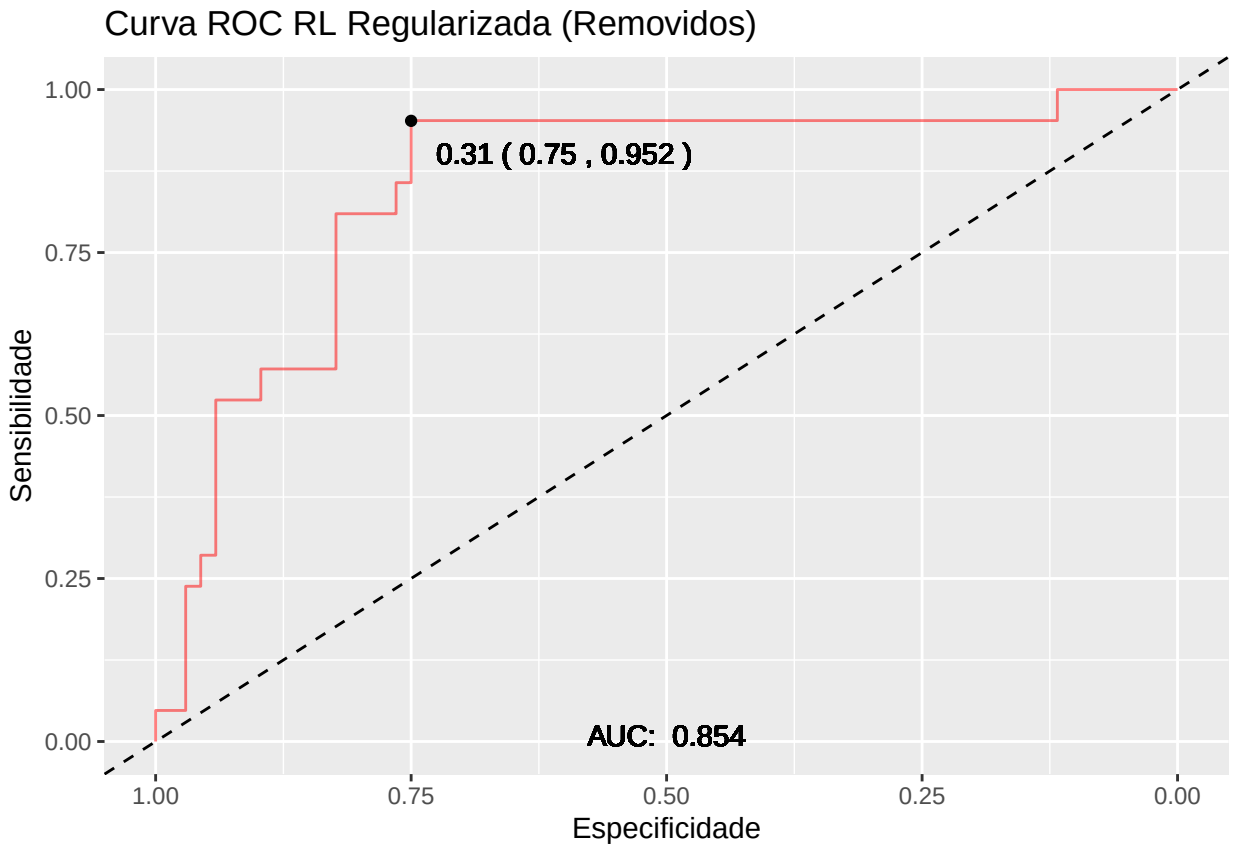
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  99  28
##        Yes  9  25
##
##                Accuracy : 0.7702
##                  95% CI : (0.6974, 0.8327)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.003815
##
##                   Kappa : 0.4274
##
##  Mcnemar's Test P-Value : 0.003085
##
##             Sensitivity : 0.9167
##             Specificity : 0.4717
##          Pos Pred Value : 0.7795
##          Neg Pred Value : 0.7353
##              Prevalence : 0.6708
##          Detection Rate : 0.6149
```

```
##    Detection Prevalence : 0.7888
##       Balanced Accuracy : 0.6942
##
##        'Positive' Class : No
##
```

```
## Regressão Logística Regularizada sem missing
rocRL4 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloRL4, test_without_NAs, ty
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRL4, titulo = "Curva ROC RL Regularizada (Removidos)")
```




```
predictRL4 <- predict(modeloRL4, newdata = test_without_NAs)
confusionMatrix(predictRL4, test_without_NAs$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  64  11
##        Yes  4  10
```

```
##
##                Accuracy : 0.8315
##                  95% CI : (0.7373, 0.9025)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.08127
##
##                   Kappa : 0.4717
##
##  Mcnemar's Test P-Value : 0.12134
##
##             Sensitivity : 0.9412
##             Specificity : 0.4762
##          Pos Pred Value : 0.8533
##          Neg Pred Value : 0.7143
##              Prevalence : 0.7640
##          Detection Rate : 0.7191
##    Detection Prevalence : 0.8427
##       Balanced Accuracy : 0.7087
##
##        'Positive' Class : No
##
```

Random Forest

Modelagem

```
library(caret)

train.control <- caret::trainControl(method = "cv", number = 10) # Cross-validation com k=10

## Random Forest / dados imputados
set.seed(23)
tuneGrid <- expand.grid(.mtry = c(1: 10))
modeloRF1 <- train(diabetes~.,
    train,
    method = "rf",
    metric = "Accuracy",
    tuneGrid = tuneGrid,
    trControl = train.control,
    importance = TRUE,
    ntree = 1000)
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range
```

```r
print(modeloRF1)
```

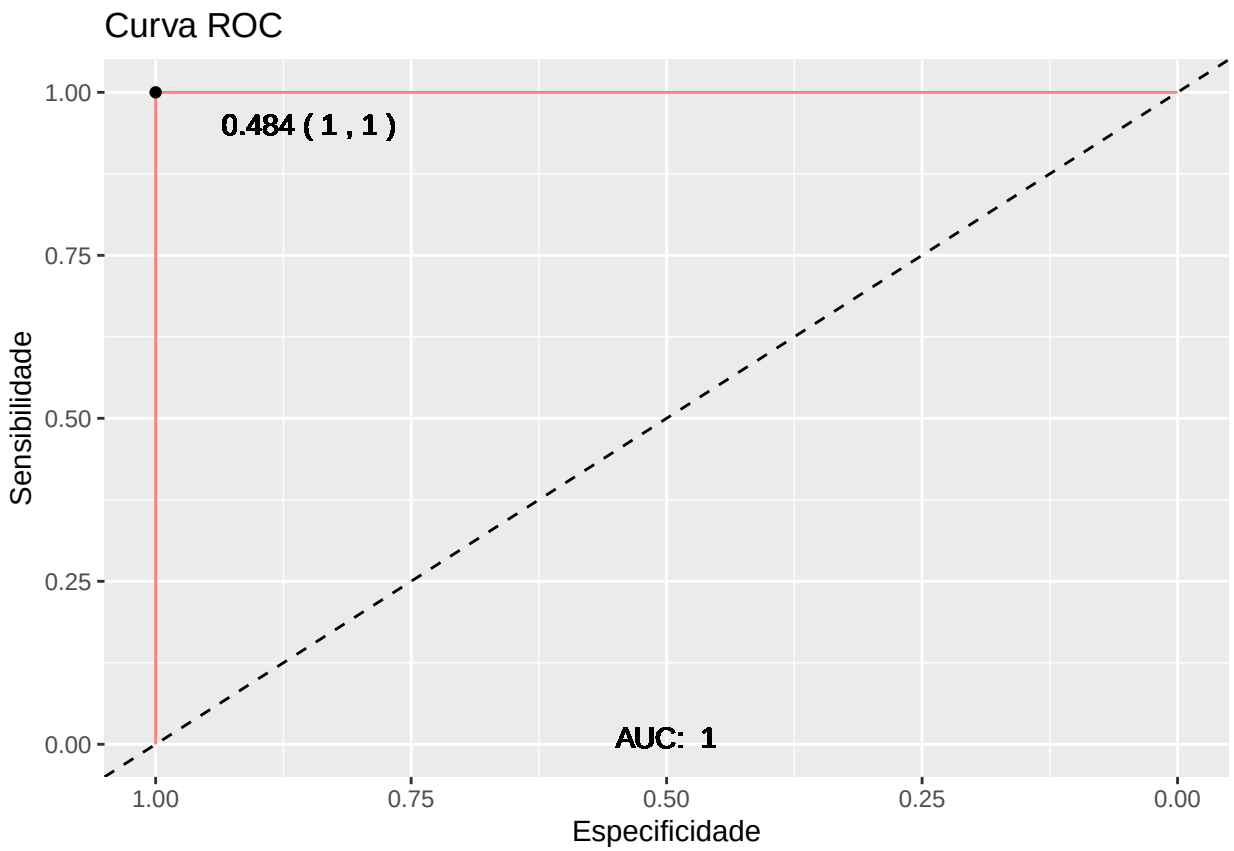
```
## Random Forest
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 483, 483, 484, 483, 482, 484, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.7506099  0.4361204
##    2    0.7562004  0.4559871
##    3    0.7468001  0.4344252
##    4    0.7486869  0.4382093
##    5    0.7487218  0.4409065
##    6    0.7505387  0.4457852
##    7    0.7523569  0.4506149
##    8    0.7468001  0.4377380
##    9    0.7431300  0.4295050
##   10    0.7487567  0.4392488
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
rocRF1 <- roc(response = train$diabetes, predictor = predict(modeloRF1, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocRF1)
```




```
## Random Forest / sem missing
set.seed(23)
tuneGrid <- expand.grid(.mtry = c(1: 10))
modeloRF2 <- train(diabetes~.,
    train_without_NAs,
    method = "rf",
    metric = "Accuracy",
    tuneGrid = tuneGrid,
    trControl = train.control,
    importance = TRUE,
    ntree = 1000)
```

```
## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
```

```
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range

## Warning in randomForest.default(x, y, mtry = param$mtry, ...): invalid mtry:
## reset to within valid range
```

```r
print(modeloRF2)
```

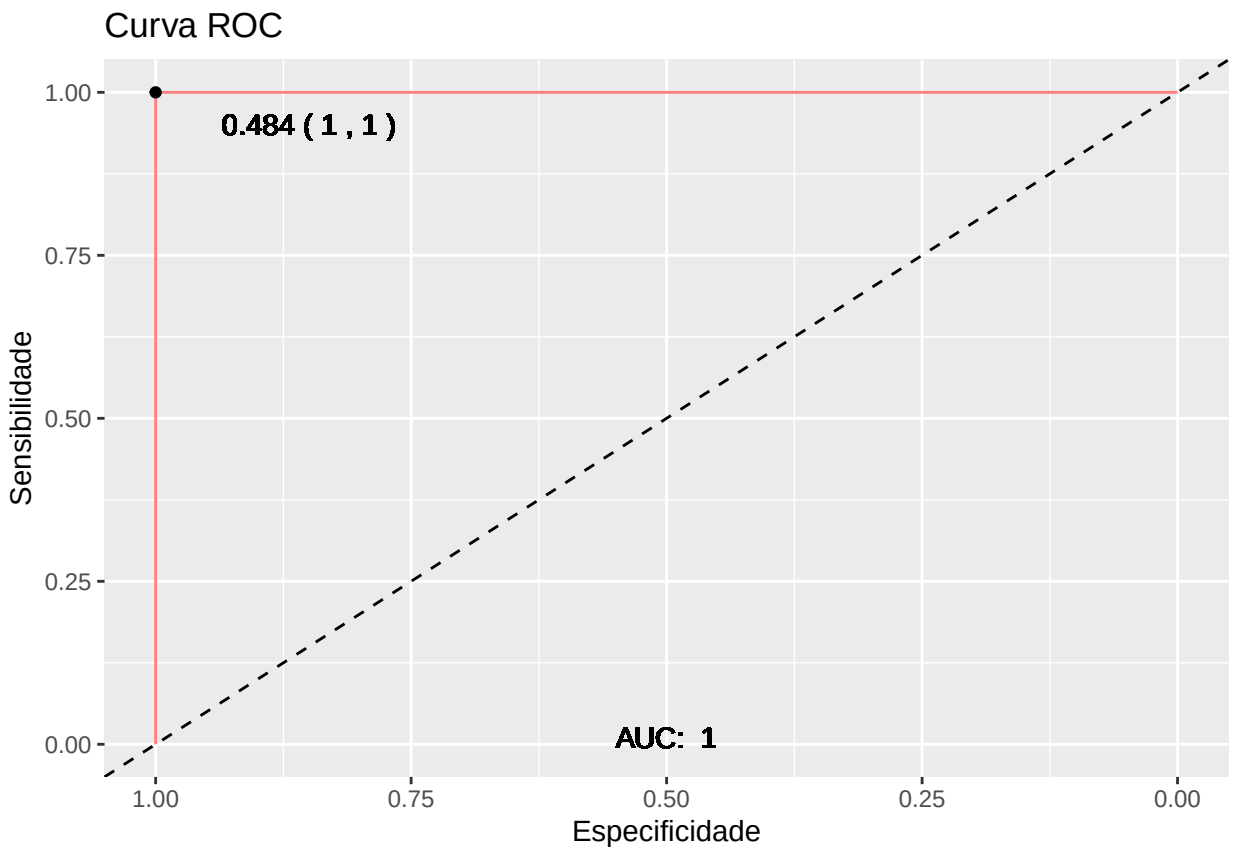
```
## Random Forest
##
## 266 samples
##   8 predictor
```

```
##   2 classes: 'No', 'Yes'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 240, 240, 239, 239, 239, 239, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.7750712  0.4927747
##    2    0.7787749  0.5063822
##    3    0.7903134  0.5363471
##    4    0.7938746  0.5468630
##    5    0.7826211  0.5182975
##    6    0.7827635  0.5166230
##    7    0.7864672  0.5255357
##    8    0.7827635  0.5201542
##    9    0.7904558  0.5372170
##   10    0.7903134  0.5347168
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```
rocRF2 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloRF1, train_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRF2)
```



Árvores de decisão geradas

```
tree_num <- which(modeloRF1$finalModel$forest$ndbigtree == min(modeloRF1$finalModel$forest$ndbigtree))
print(paste("Menor árvore com imputação",tree_num))
```

```
## [1] "Menor árvore com imputação 876"
```

```
tree_func(final_model = modeloRF1$finalModel, tree_num[1])
```

```
## Warning: Duplicated aesthetics after name standardisation: na.rm
```

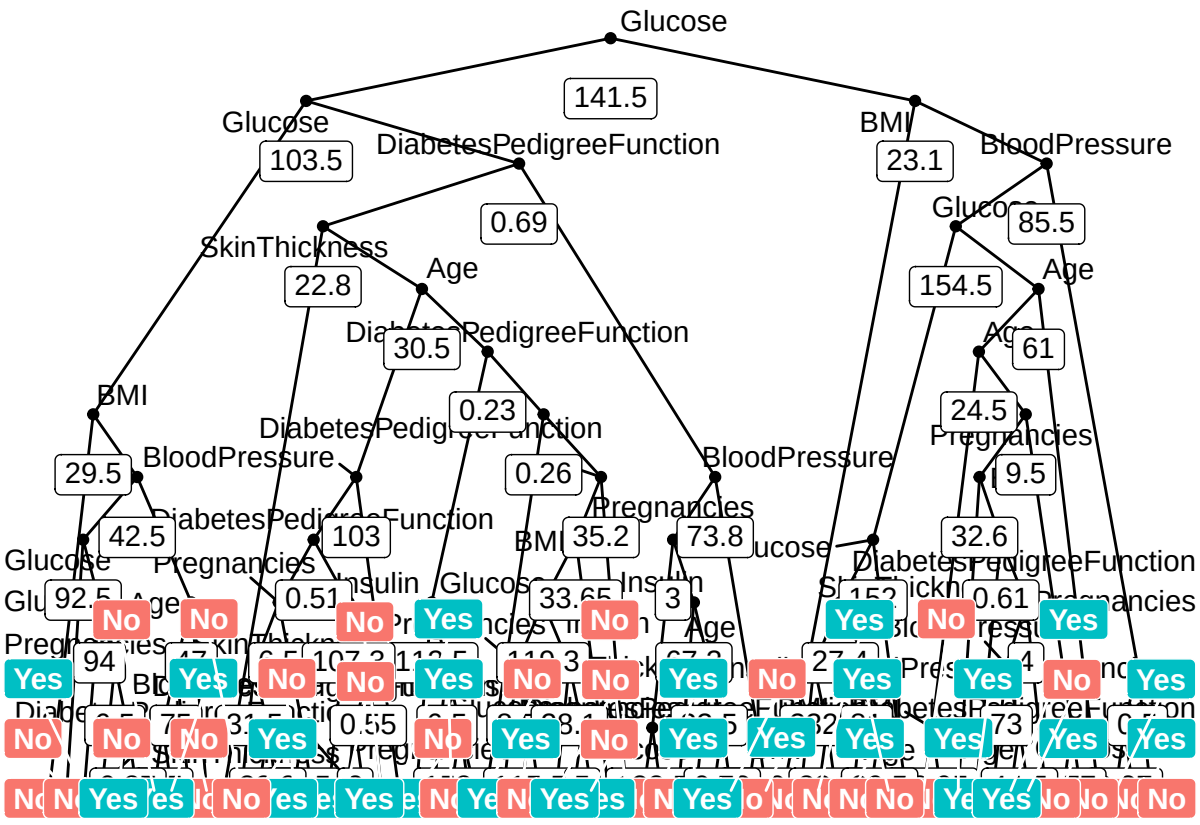
```
## Warning: Duplicated aesthetics after name standardisation: na.rm
```

```
## Warning: Duplicated aesthetics after name standardisation: na.rm
```

```
## Warning: Removed 64 rows containing missing values (geom_text_repel).
```

```
## Warning: Removed 64 rows containing missing values (geom_label).
```

```
## Warning: Removed 63 rows containing missing values (geom_label_repel).
```




```
tree_num <- which(modeloRF2$finalModel$forest$ndbigtree == min(modeloRF2$finalModel$forest$ndbigtree))
print(paste("Menor árvore sem imputação:",tree_num))
```

```
## [1] "Menor árvore sem imputação: 250"
```

```
tree_func(final_model = modeloRF2$finalModel, tree_num[1])
```

```
## Warning: Duplicated aesthetics after name standardisation: na.rm
```

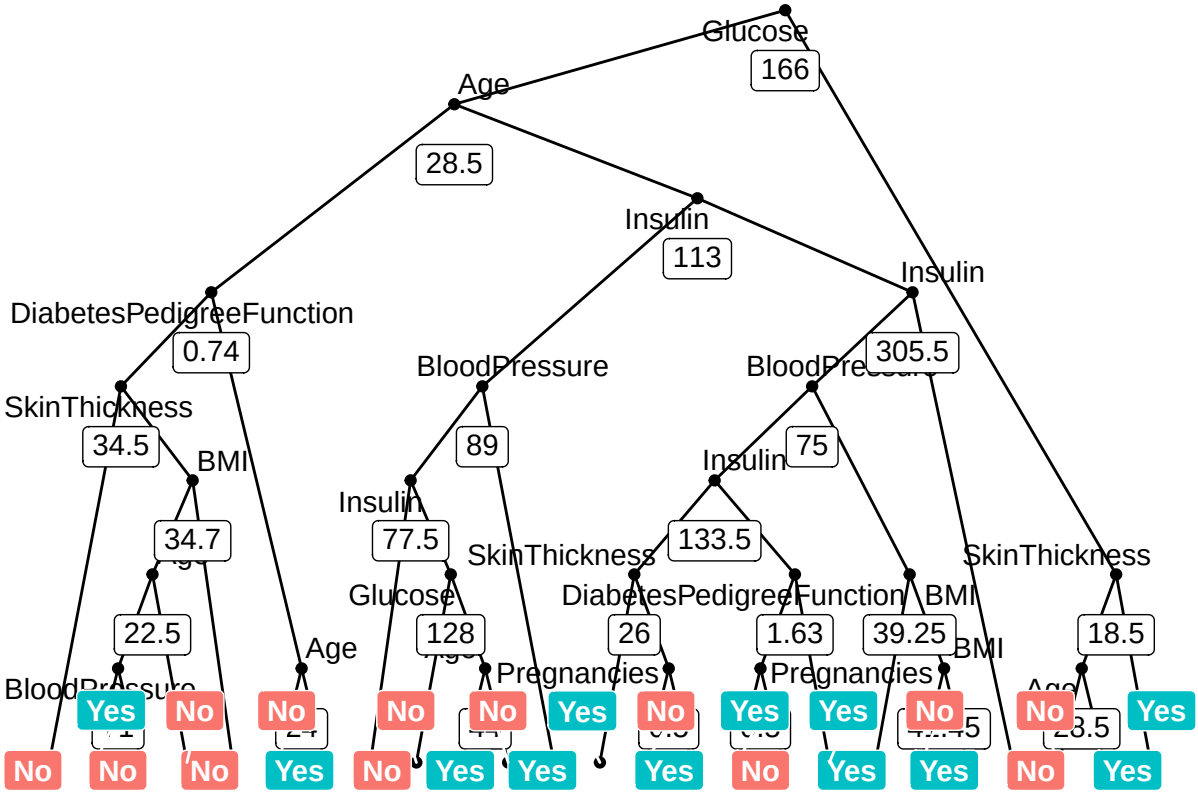
```
## Warning: Duplicated aesthetics after name standardisation: na.rm
```

```
## Warning: Duplicated aesthetics after name standardisation: na.rm
```

```
## Warning: Removed 25 rows containing missing values (geom_text_repel).
```

```
## Warning: Removed 25 rows containing missing values (geom_label).
```

```
## Warning: Removed 24 rows containing missing values (geom_label_repel).
```



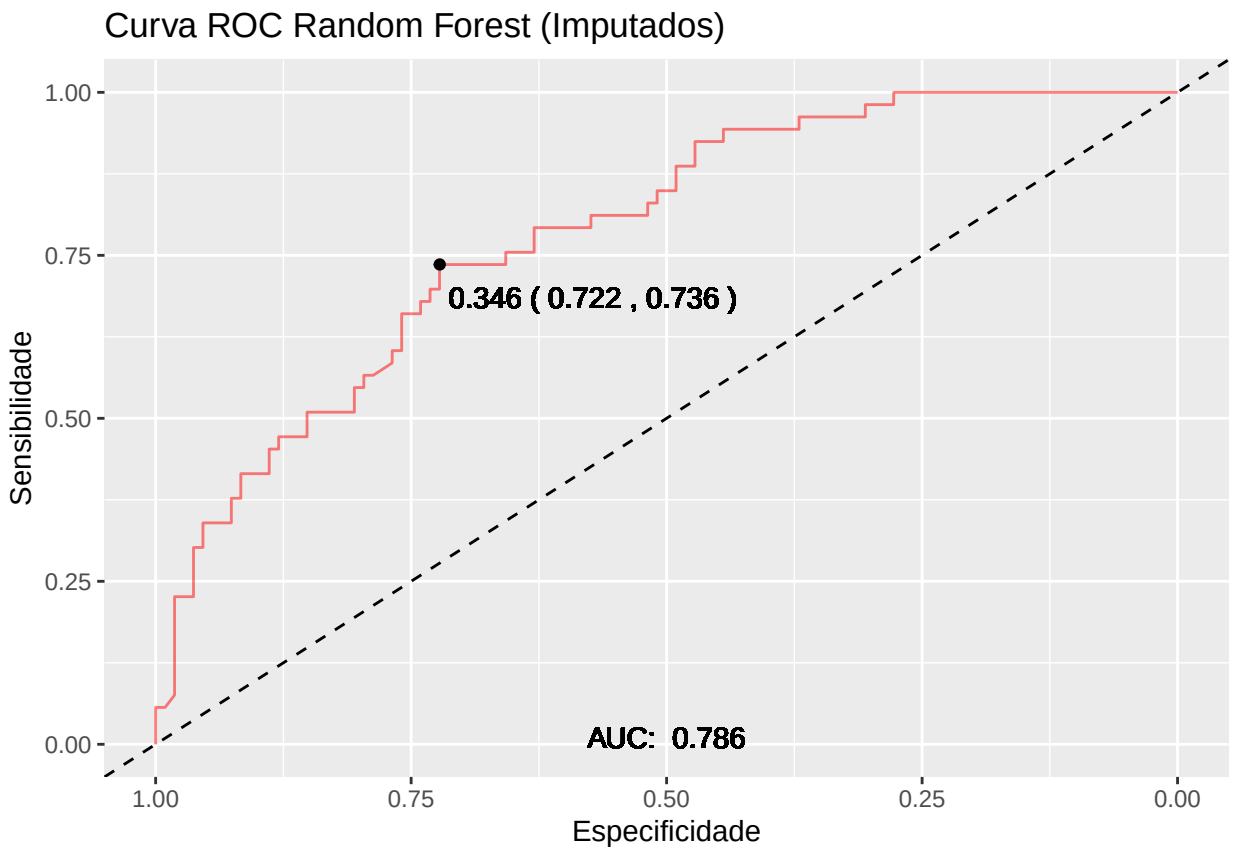
Teste

```
## Random Forest com dados imputados
rocRF1 <- roc(response = test$diabetes, predictor = predict(modeloRF1, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocRF1, titulo = "Curva ROC Random Forest (Imputados)")
```




```
predictRF1 <- predict(modeloRF1, newdata = test)
print("Dados imputados:")
```

```
## [1] "Dados imputados:"
```

```
confusionMatrix(predictRF1, test$diabetes)
```

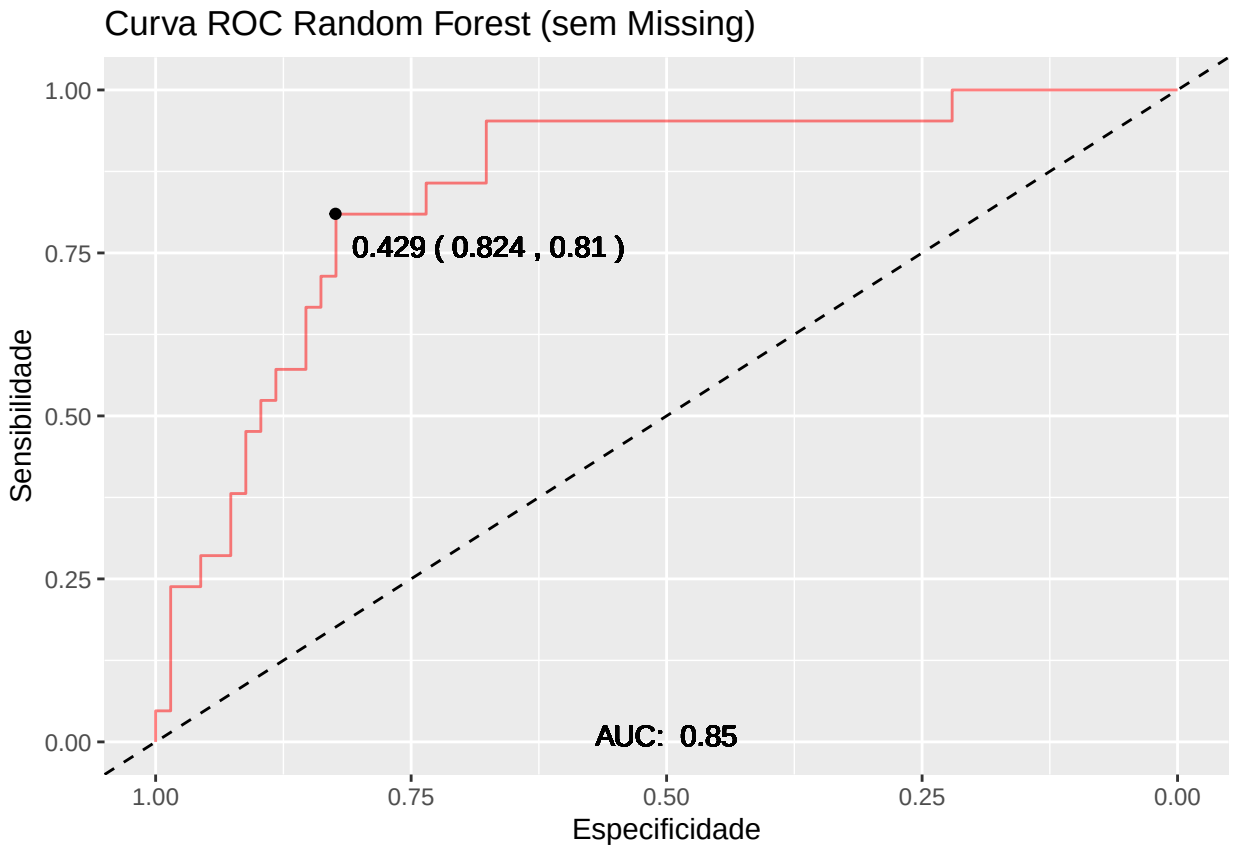
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  93  28
##        Yes 15  25
##
##                Accuracy : 0.7329
##                  95% CI : (0.6576, 0.7995)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.05368
##
##                   Kappa : 0.355
##
##  Mcnemar's Test P-Value : 0.06725
##
##             Sensitivity : 0.8611
```

```
##             Specificity : 0.4717
##          Pos Pred Value : 0.7686
##          Neg Pred Value : 0.6250
##              Prevalence : 0.6708
##          Detection Rate : 0.5776
##    Detection Prevalence : 0.7516
##       Balanced Accuracy : 0.6664
##
##        'Positive' Class : No
##
```

```
## Random Forest com dados imputados
rocRF2 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloRF2, test_without_NAs, ty
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocRF2, titulo = "Curva ROC Random Forest (sem Missing)")
```




```
predictRF2 <- predict(modeloRF2, newdata = test_without_NAs)
print("Sem Missing:")
```

```
## [1] "Sem Missing:"
```

```r
confusionMatrix(predictRF2, test_without_NAs$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  59   9
##        Yes  9  12
##
##                Accuracy : 0.7978
##                  95% CI : (0.6993, 0.8755)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.2709
##
##                   Kappa : 0.4391
##
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.8676
##             Specificity : 0.5714
##          Pos Pred Value : 0.8676
##          Neg Pred Value : 0.5714
##              Prevalence : 0.7640
##          Detection Rate : 0.6629
##    Detection Prevalence : 0.7640
##       Balanced Accuracy : 0.7195
##
##        'Positive' Class : No
##
```

Suporte Vector Machine

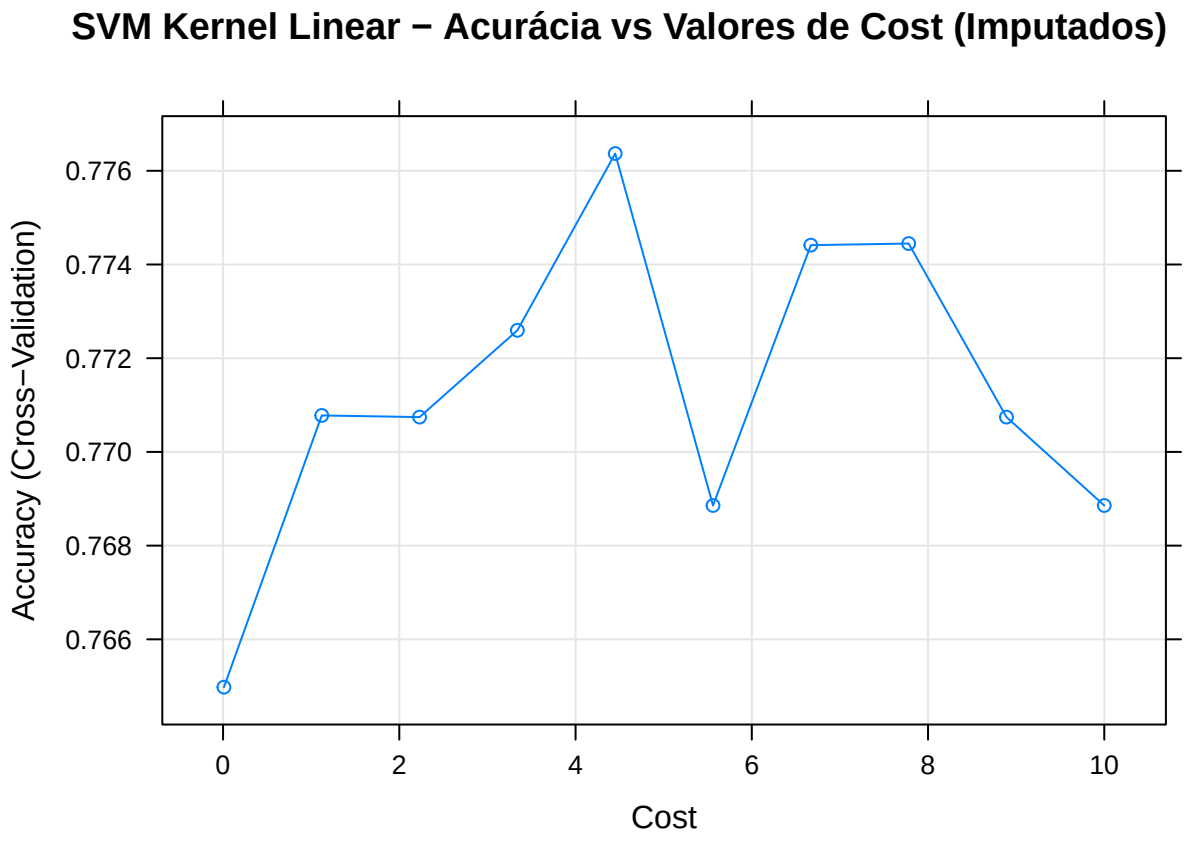
Modelagem

SVM Kernel Linear com dados imputados

```r
train.control <- caret::trainControl(method = "cv", number = 10, savePred=T, classProb=T) # Cross-valid
#library(e1071) #SVM
## SVM Kernel Linear com dados imputados
set.seed(23)
modeloSVM1 <- caret::train(
  diabetes ~., data = train, method = "svmLinear",
  probability = T,
  trControl = train.control,
  preProcess = c("center","scale"),
  tuneGrid = expand.grid(C = seq(0.01, 10, length = 10))
)
print(modeloSVM1)
```

```
## Support Vector Machines with Linear Kernel
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 483, 483, 484, 483, 482, 484, ...
## Resampling results across tuning parameters:
##
##   C      Accuracy   Kappa
##    0.01  0.7649794  0.4734524
##    1.12  0.7707782  0.4824355
##    2.23  0.7707433  0.4808278
##    3.34  0.7725951  0.4843473
##    4.45  0.7763687  0.4917503
##    5.56  0.7688565  0.4769505
##    6.67  0.7744133  0.4896157
##    7.78  0.7744470  0.4862920
##    8.89  0.7707433  0.4778763
##   10.00  0.7688565  0.4754528
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 4.45.
```

```
plot(modeloSVM1, main="SVM Kernel Linear - Acurácia vs Valores de Cost (Imputados)")
```

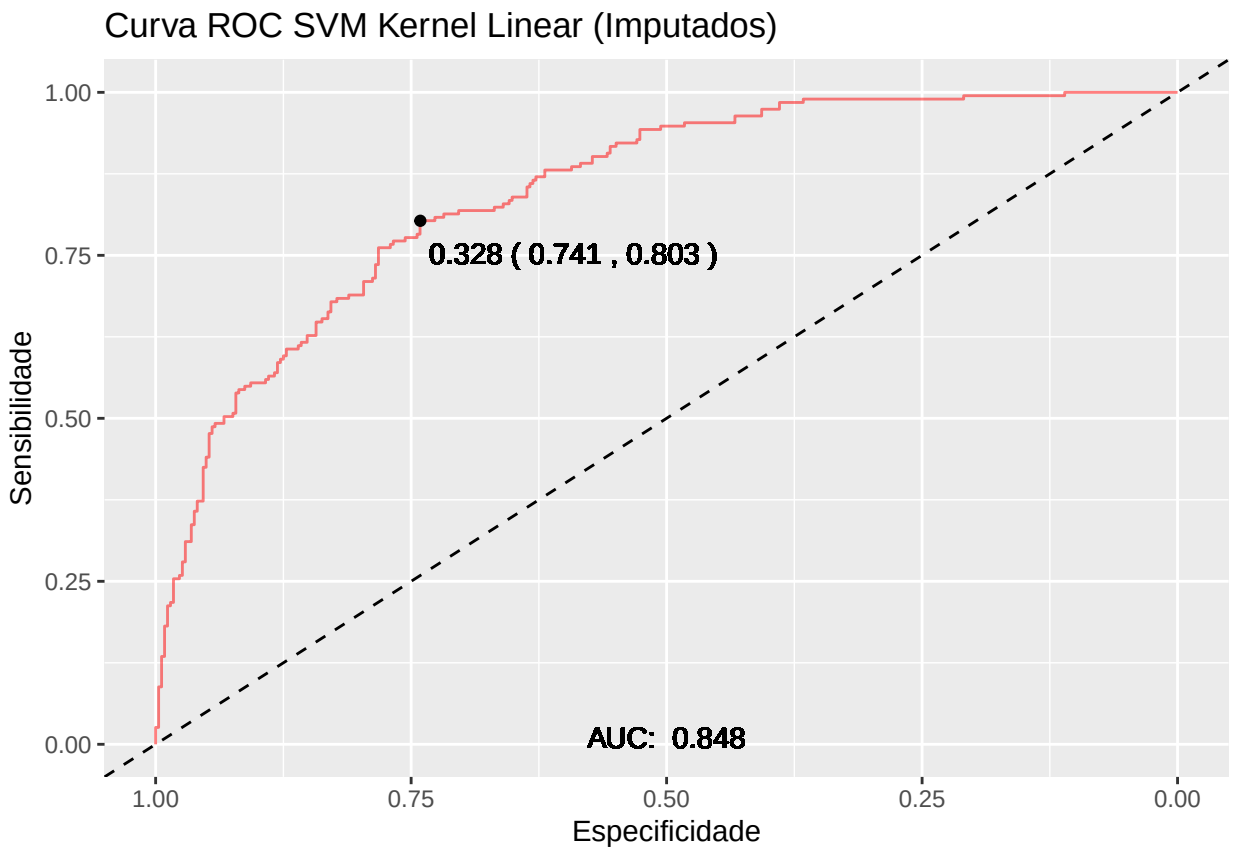



```
rocSVM1 <- roc(response = train$diabetes, predictor = predict(modeloSVM1, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocSVM1, titulo = "Curva ROC SVM Kernel Linear (Imputados)")
```



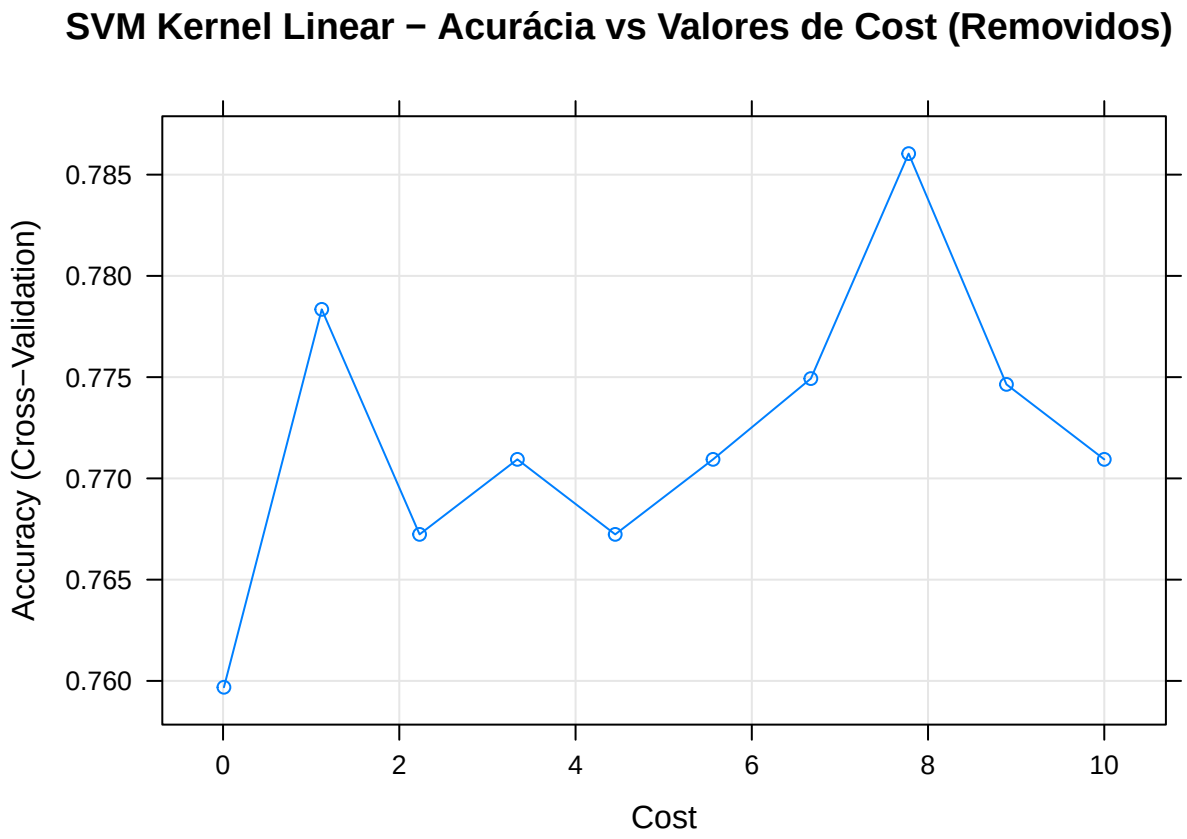
SVM Kernel Linear sem missing

```
## SVM Kernel Linear sem missing
set.seed(23)
modeloSVM2 <- caret::train(
  diabetes ~., data = train_without_NAs, method = "svmLinear",
  trControl = train.control,
  preProcess = c("center","scale"),
  tuneGrid = expand.grid(C = seq(0.01, 10, length = 10))
)
print(modeloSVM2)
```

```
## Support Vector Machines with Linear Kernel
##
## 266 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 240, 240, 239, 239, 239, 239, ...
## Resampling results across tuning parameters:
##
```

```
##    C      Accuracy   Kappa
##     0.01  0.7596866  0.4629441
##     1.12  0.7783476  0.4980691
##     2.23  0.7672365  0.4744505
##     3.34  0.7709402  0.4839230
##     4.45  0.7672365  0.4774236
##     5.56  0.7709402  0.4781789
##     6.67  0.7749288  0.4893745
##     7.78  0.7860399  0.5103509
##     8.89  0.7746439  0.4906956
##    10.00  0.7709402  0.4754996
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 7.78.
```

```
plot(modeloSVM2, main="SVM Kernel Linear - Acurácia vs Valores de Cost (Removidos)")
```

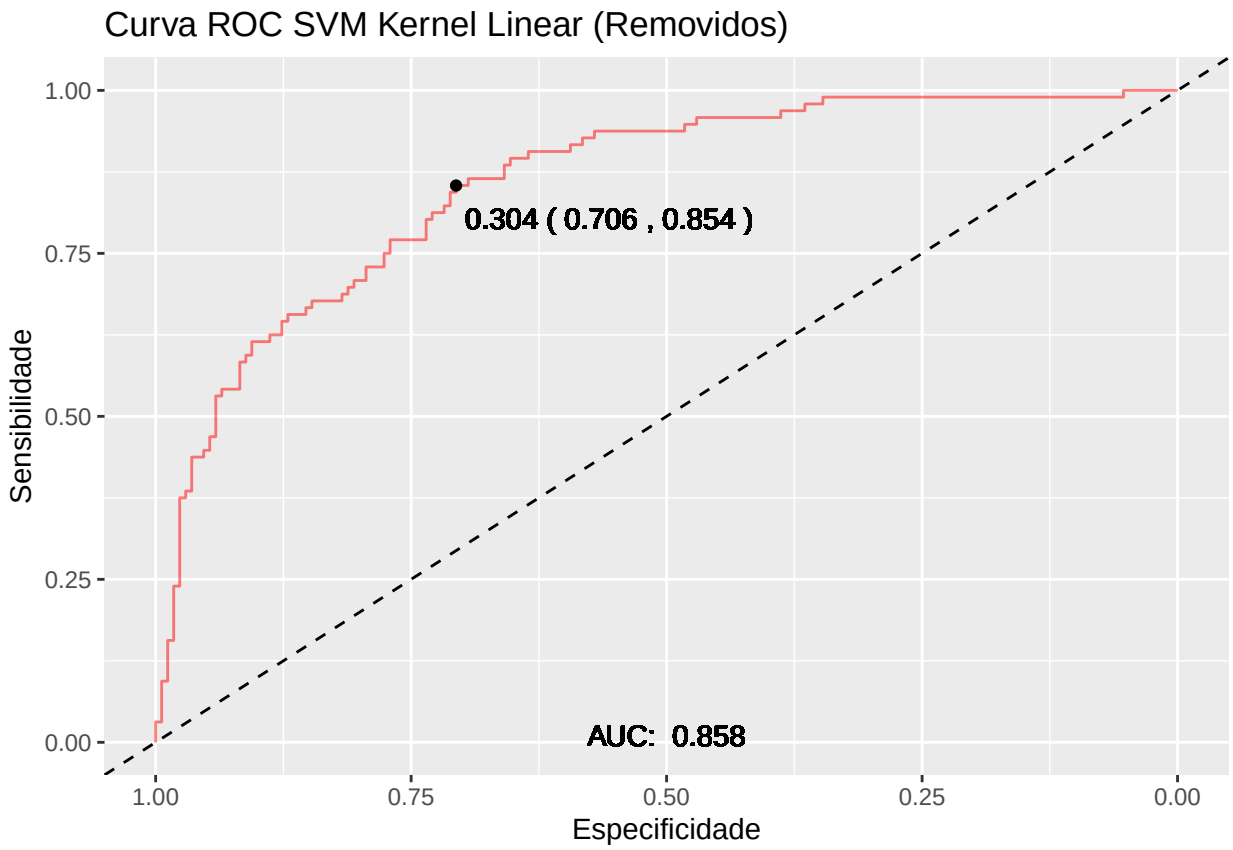



```
rocSVM2 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloSVM2, train_without_NAs
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocSVM2, titulo ="Curva ROC SVM Kernel Linear (Removidos)")
```



SVM Kernel Não Linear com dados imputados

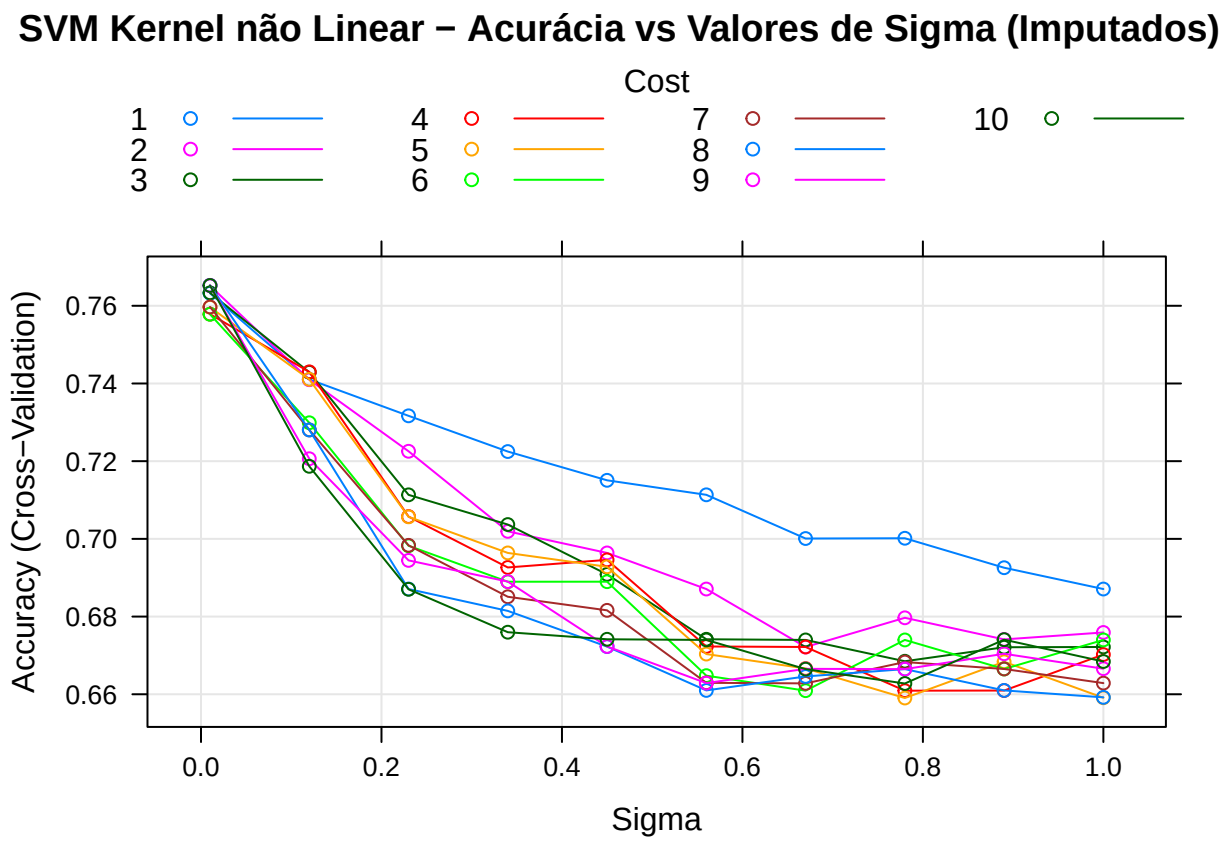
```
## SVM Kernel Não Linear com dados imputados
set.seed(23)
modeloSVM3 <- caret::train(
  diabetes ~., data = train, method = "svmRadial",
  trControl = train.control,
  preProcess = c("center","scale"),
  tuneGrid = expand.grid(sigma = seq(0.01, 1, length = 10) , C = seq(1, 10, length = 10))
)
print(modeloSVM3)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 537 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
```

```
## Summary of sample sizes: 483, 483, 484, 483, 482, 484, ...
## Resampling results across tuning parameters:
##
##   sigma  C   Accuracy   Kappa
##   0.01    1  0.7632323  0.4654255
##   0.01    2  0.7651541  0.4686906
##   0.01    3  0.7633371  0.4652478
##   0.01    4  0.7578502  0.4499811
##   0.01    5  0.7596334  0.4565153
##   0.01    6  0.7578153  0.4521902
##   0.01    7  0.7597021  0.4544051
##   0.01    8  0.7652576  0.4649773
##   0.01    9  0.7652576  0.4667128
##   0.01   10  0.7652227  0.4665617
##   0.12    1  0.7410412  0.4172729
##   0.12    2  0.7409688  0.4130742
##   0.12    3  0.7428905  0.4189473
##   0.12    4  0.7429928  0.4173560
##   0.12    5  0.7410724  0.4173790
##   0.12    6  0.7298901  0.3891193
##   0.12    7  0.7280382  0.3836402
##   0.12    8  0.7280382  0.3865933
##   0.12    9  0.7206296  0.3666429
##   0.12   10  0.7186754  0.3627544
##   0.23    1  0.7316746  0.3933253
##   0.23    2  0.7225513  0.3738773
##   0.23    3  0.7113354  0.3457537
##   0.23    4  0.7057449  0.3323967
##   0.23    5  0.7056750  0.3313558
##   0.23    6  0.6982314  0.3163946
##   0.23    7  0.6983362  0.3096625
##   0.23    8  0.6870853  0.2825074
##   0.23    9  0.6944578  0.3009454
##   0.23   10  0.6870167  0.2782018
##   0.34    1  0.7224827  0.3714102
##   0.34    2  0.7019376  0.3273452
##   0.34    3  0.7036872  0.3239199
##   0.34    4  0.6926421  0.2970354
##   0.34    5  0.6963808  0.3111219
##   0.34    6  0.6889721  0.2906708
##   0.34    7  0.6850975  0.2810966
##   0.34    8  0.6814624  0.2675126
##   0.34    9  0.6890071  0.2828918
##   0.34   10  0.6759742  0.2551517
##   0.45    1  0.7150753  0.3509448
##   0.45    2  0.6963833  0.3062685
##   0.45    3  0.6908589  0.2926231
##   0.45    4  0.6945988  0.3106269
##   0.45    5  0.6928156  0.3041198
##   0.45    6  0.6889734  0.2984869
##   0.45    7  0.6816009  0.2875827
##   0.45    8  0.6722718  0.2590777
##   0.45    9  0.6723067  0.2552091
##   0.45   10  0.6741586  0.2616750
```

```
##   0.56    1  0.7113379  0.3428026
##   0.56    2  0.6870866  0.2825257
##   0.56    3  0.6741573  0.2575748
##   0.56    4  0.6723054  0.2584329
##   0.56    5  0.6703500  0.2495136
##   0.56    6  0.6647958  0.2365721
##   0.56    7  0.6629439  0.2385834
##   0.56    8  0.6609872  0.2203032
##   0.56    9  0.6628740  0.2280941
##   0.56   10  0.6739864  0.2501550
##   0.67    1  0.7000870  0.3084599
##   0.67    2  0.6721670  0.2416235
##   0.67    3  0.6739864  0.2500257
##   0.67    4  0.6721695  0.2490706
##   0.67    5  0.6666463  0.2338014
##   0.67    6  0.6609186  0.2177422
##   0.67    7  0.6628054  0.2275974
##   0.67    8  0.6645887  0.2323034
##   0.67    9  0.6665428  0.2321039
##   0.67   10  0.6664754  0.2375644
##   0.78    1  0.7001569  0.3088335
##   0.78    2  0.6796805  0.2585207
##   0.78    3  0.6684982  0.2323193
##   0.78    4  0.6609186  0.2142967
##   0.78    5  0.6590318  0.2111359
##   0.78    6  0.6739877  0.2470423
##   0.78    7  0.6682924  0.2377308
##   0.78    8  0.6664754  0.2266465
##   0.78    9  0.6664754  0.2258914
##   0.78   10  0.6627705  0.2125283
##   0.89    1  0.6925761  0.2908961
##   0.89    2  0.6741236  0.2431205
##   0.89    3  0.6721009  0.2334575
##   0.89    4  0.6609536  0.2093185
##   0.89    5  0.6683610  0.2275698
##   0.89    6  0.6665091  0.2235560
##   0.89    7  0.6665091  0.2188595
##   0.89    8  0.6609872  0.2072523
##   0.89    9  0.6704212  0.2236802
##   0.89   10  0.6740550  0.2476682
##   1.00    1  0.6870891  0.2753881
##   1.00    2  0.6759081  0.2379519
##   1.00    3  0.6721695  0.2306436
##   1.00    4  0.6702128  0.2300421
##   1.00    5  0.6591017  0.2023119
##   1.00    6  0.6739864  0.2383797
##   1.00    7  0.6628728  0.2097380
##   1.00    8  0.6592040  0.2043506
##   1.00    9  0.6665790  0.2159346
##   1.00   10  0.6684308  0.2258624
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.01 and C = 9.
```

```
plot(modeloSVM3, main="SVM Kernel não Linear - Acurácia vs Valores de Sigma (Imputados)")
```

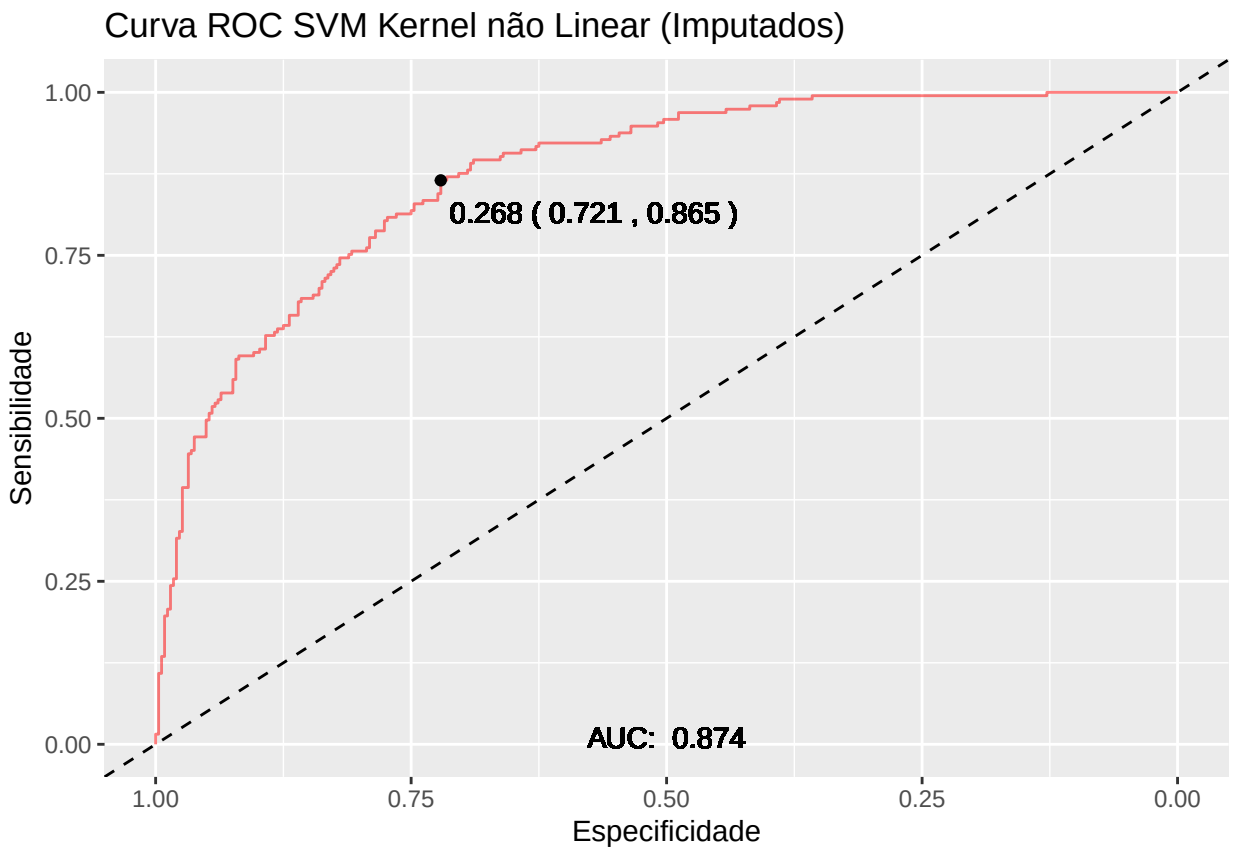



```
rocSVM3 <- roc(response = train$diabetes, predictor = predict(modeloSVM3, train, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocSVM3, titulo ="Curva ROC SVM Kernel não Linear (Imputados)")
```



SVM Kernel Não Linear sem missing

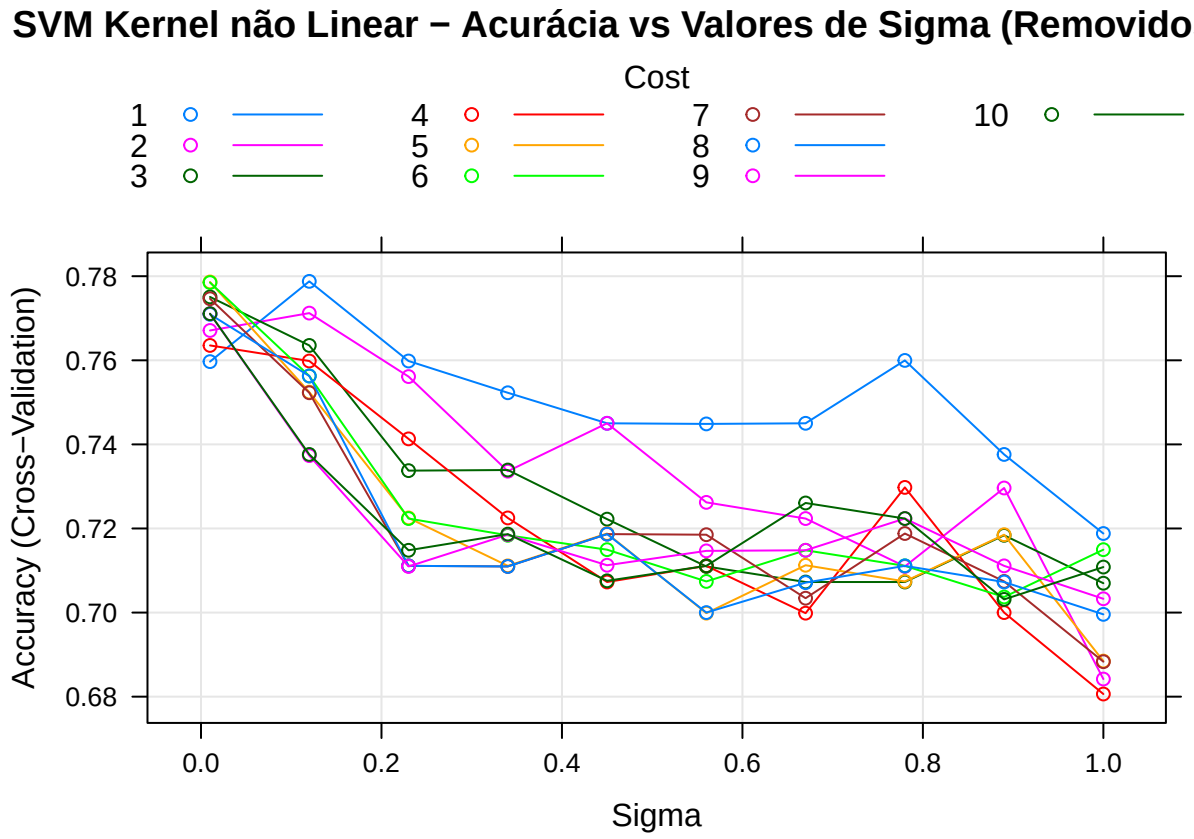
```
## SVM Kernel Não Linear sem missing
set.seed(23)
modeloSVM4 <- caret::train(
  diabetes ~., data = train_without_NAs, method = "svmRadial",
  trControl = train.control,
  preProcess = c("center","scale"),
  tuneGrid = expand.grid(sigma = seq(0.01, 1, length = 10) , C = seq(1, 10, length = 10))
)
print(modeloSVM4)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 266 samples
##   8 predictor
##   2 classes: 'No', 'Yes'
##
## Pre-processing: centered (8), scaled (8)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 240, 240, 239, 239, 239, 239, ...
## Resampling results across tuning parameters:
##
```

```
##   sigma  C   Accuracy   Kappa
##   0.01    1  0.7596866  0.4650398
##   0.01    2  0.7670940  0.4812458
##   0.01    3  0.7750712  0.4961683
##   0.01    4  0.7635328  0.4698621
##   0.01    5  0.7786325  0.4972411
##   0.01    6  0.7784900  0.4946139
##   0.01    7  0.7746439  0.4843767
##   0.01    8  0.7709402  0.4757028
##   0.01    9  0.7710826  0.4782618
##   0.01   10  0.7710826  0.4782618
##   0.12    1  0.7787749  0.4951781
##   0.12    2  0.7712251  0.4802989
##   0.12    3  0.7635328  0.4566007
##   0.12    4  0.7598291  0.4445875
##   0.12    5  0.7524217  0.4302310
##   0.12    6  0.7562678  0.4371691
##   0.12    7  0.7522792  0.4200541
##   0.12    8  0.7562678  0.4370246
##   0.12    9  0.7373219  0.3911532
##   0.12   10  0.7376068  0.3892707
##   0.23    1  0.7598291  0.4523565
##   0.23    2  0.7561254  0.4451085
##   0.23    3  0.7337607  0.4021808
##   0.23    4  0.7413105  0.4199586
##   0.23    5  0.7225071  0.3736789
##   0.23    6  0.7223647  0.3670588
##   0.23    7  0.7111111  0.3417905
##   0.23    8  0.7111111  0.3408347
##   0.23    9  0.7109687  0.3405406
##   0.23   10  0.7148148  0.3367158
##   0.34    1  0.7522792  0.4378279
##   0.34    2  0.7336182  0.4009842
##   0.34    3  0.7339031  0.3950800
##   0.34    4  0.7225071  0.3770402
##   0.34    5  0.7111111  0.3487114
##   0.34    6  0.7183761  0.3638857
##   0.34    7  0.7109687  0.3366912
##   0.34    8  0.7109687  0.3441970
##   0.34    9  0.7185185  0.3508626
##   0.34   10  0.7186610  0.3590489
##   0.45    1  0.7450142  0.4217840
##   0.45    2  0.7450142  0.4322630
##   0.45    3  0.7222222  0.3712480
##   0.45    4  0.7072650  0.3469918
##   0.45    5  0.7188034  0.3656221
##   0.45    6  0.7149573  0.3580174
##   0.45    7  0.7186610  0.3583701
##   0.45    8  0.7186610  0.3619806
##   0.45    9  0.7112536  0.3476623
##   0.45   10  0.7075499  0.3454457
##   0.56    1  0.7448718  0.4243368
##   0.56    2  0.7262108  0.3809057
##   0.56    3  0.7109687  0.3386173
```

```
##   0.56    4  0.7111111  0.3454171
##   0.56    5  0.6998575  0.3195248
##   0.56    6  0.7074074  0.3369069
##   0.56    7  0.7185185  0.3779832
##   0.56    8  0.7000000  0.3153407
##   0.56    9  0.7146724  0.3515126
##   0.56   10  0.7111111  0.3485484
##   0.67    1  0.7450142  0.4361749
##   0.67    2  0.7223647  0.3844141
##   0.67    3  0.7072650  0.3418878
##   0.67    4  0.6998575  0.3169579
##   0.67    5  0.7112536  0.3447735
##   0.67    6  0.7148148  0.3577632
##   0.67    7  0.7034188  0.3278498
##   0.67    8  0.7071225  0.3375983
##   0.67    9  0.7148148  0.3570661
##   0.67   10  0.7260684  0.3767459
##   0.78    1  0.7599715  0.4719043
##   0.78    2  0.7109687  0.3479731
##   0.78    3  0.7072650  0.3401029
##   0.78    4  0.7297721  0.3925998
##   0.78    5  0.7074074  0.3478802
##   0.78    6  0.7111111  0.3430724
##   0.78    7  0.7188034  0.3574965
##   0.78    8  0.7111111  0.3475439
##   0.78    9  0.7223647  0.3713868
##   0.78   10  0.7223647  0.3729232
##   0.89    1  0.7376068  0.4239749
##   0.89    2  0.7296296  0.3890878
##   0.89    3  0.7183761  0.3677189
##   0.89    4  0.7000000  0.3174928
##   0.89    5  0.7185185  0.3622729
##   0.89    6  0.7037037  0.3369286
##   0.89    7  0.7074074  0.3383008
##   0.89    8  0.7072650  0.3383578
##   0.89    9  0.7111111  0.3472359
##   0.89   10  0.7031339  0.3065973
##   1.00    1  0.7188034  0.3915682
##   1.00    2  0.6841880  0.2662962
##   1.00    3  0.7069801  0.3218391
##   1.00    4  0.6806268  0.2520008
##   1.00    5  0.6884615  0.2735769
##   1.00    6  0.7149573  0.3506383
##   1.00    7  0.6883191  0.2773509
##   1.00    8  0.6995726  0.2933212
##   1.00    9  0.7032764  0.3099552
##   1.00   10  0.7108262  0.3449889
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.12 and C = 1.
```

```
plot(modeloSVM4, main="SVM Kernel não Linear - Acurácia vs Valores de Sigma (Removidos)")
```

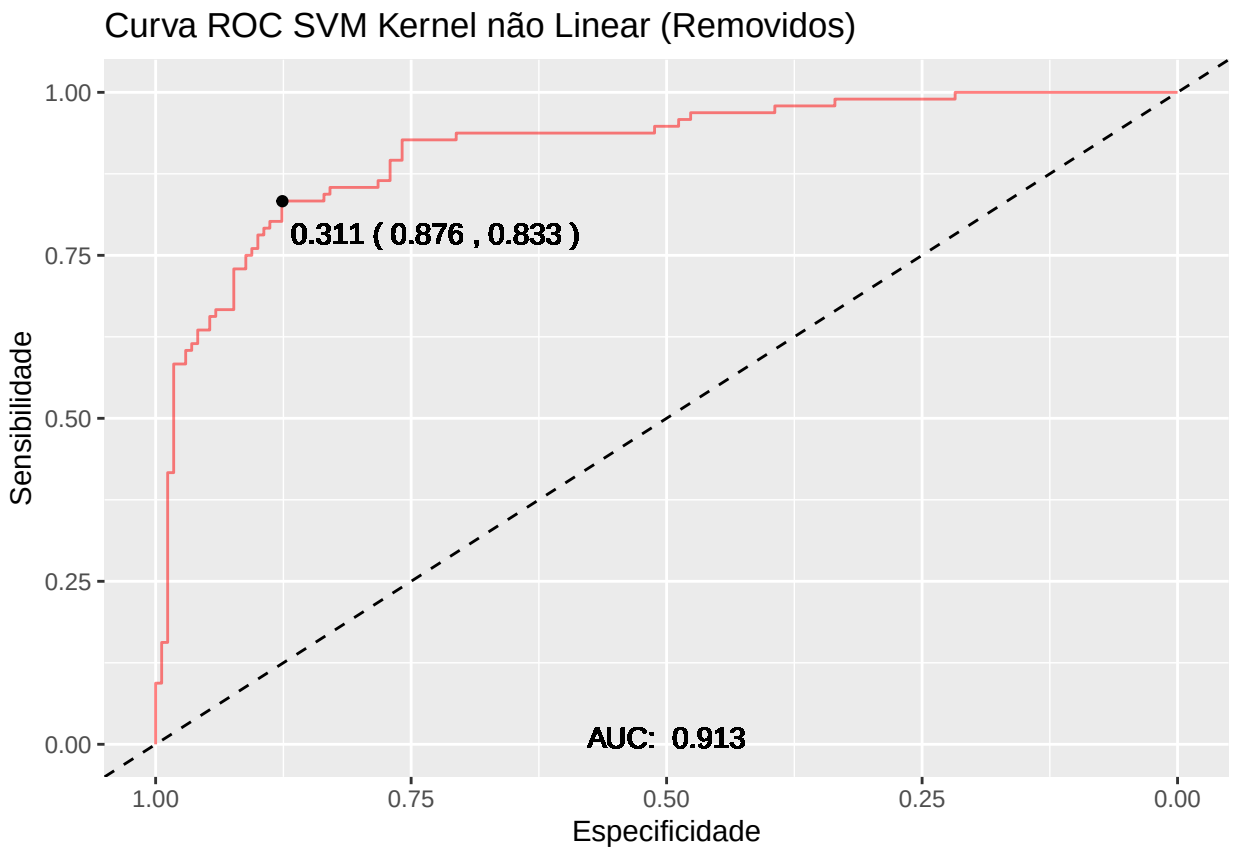



```
rocSVM4 <- roc(response = train_without_NAs$diabetes, predictor = predict(modeloSVM4, train_without_NAs
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocSVM4, titulo ="Curva ROC SVM Kernel não Linear (Removidos)")
```



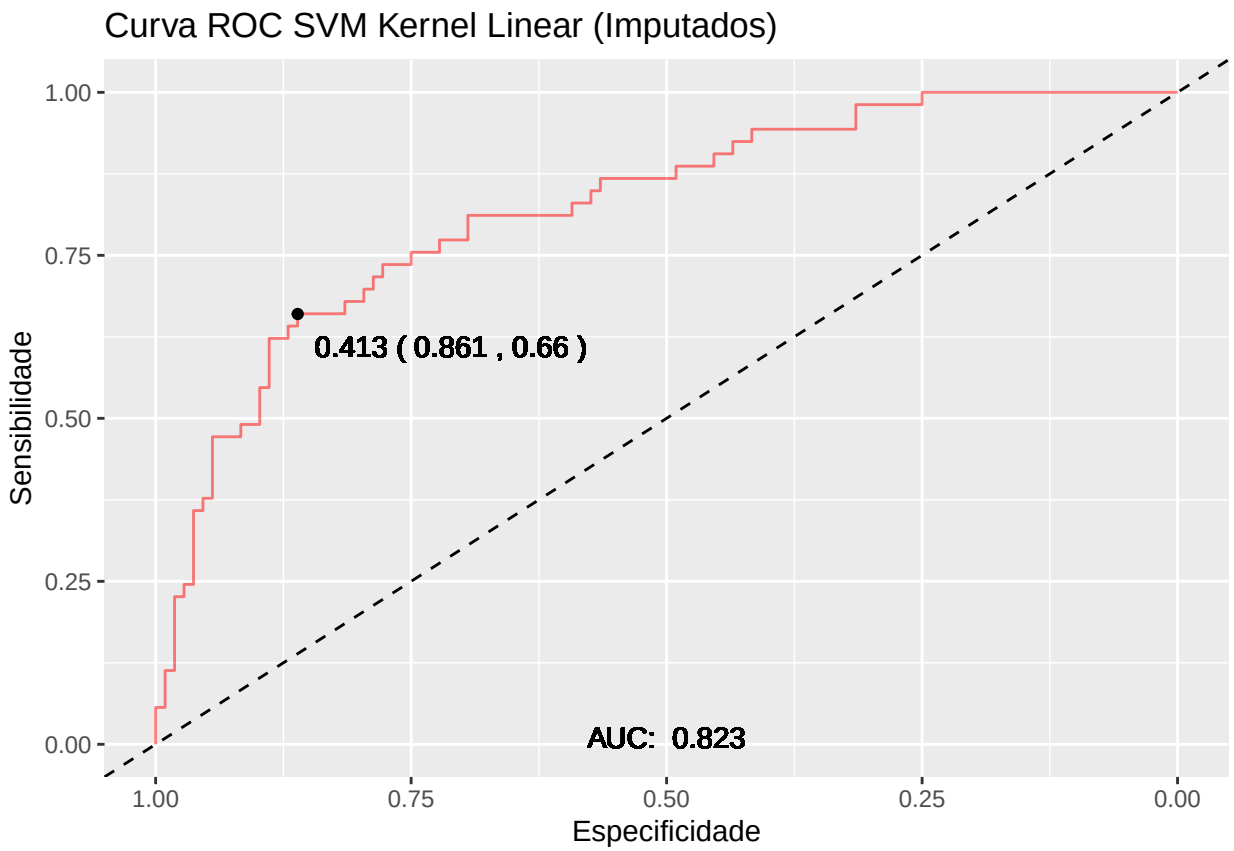
Testes

```
## SVM Kernel Linear com dados imputados
rocSVM1 <- roc(response = test$diabetes, predictor = predict(modeloSVM1, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
```

```
## Setting direction: controls < cases
```

```
plotaroc(rocSVM1, titulo = "Curva ROC SVM Kernel Linear (Imputados)")
```




```
predictSVM1 <- predict(modeloSVM1, newdata = test)
confusionMatrix(predictSVM1, test$diabetes)
```

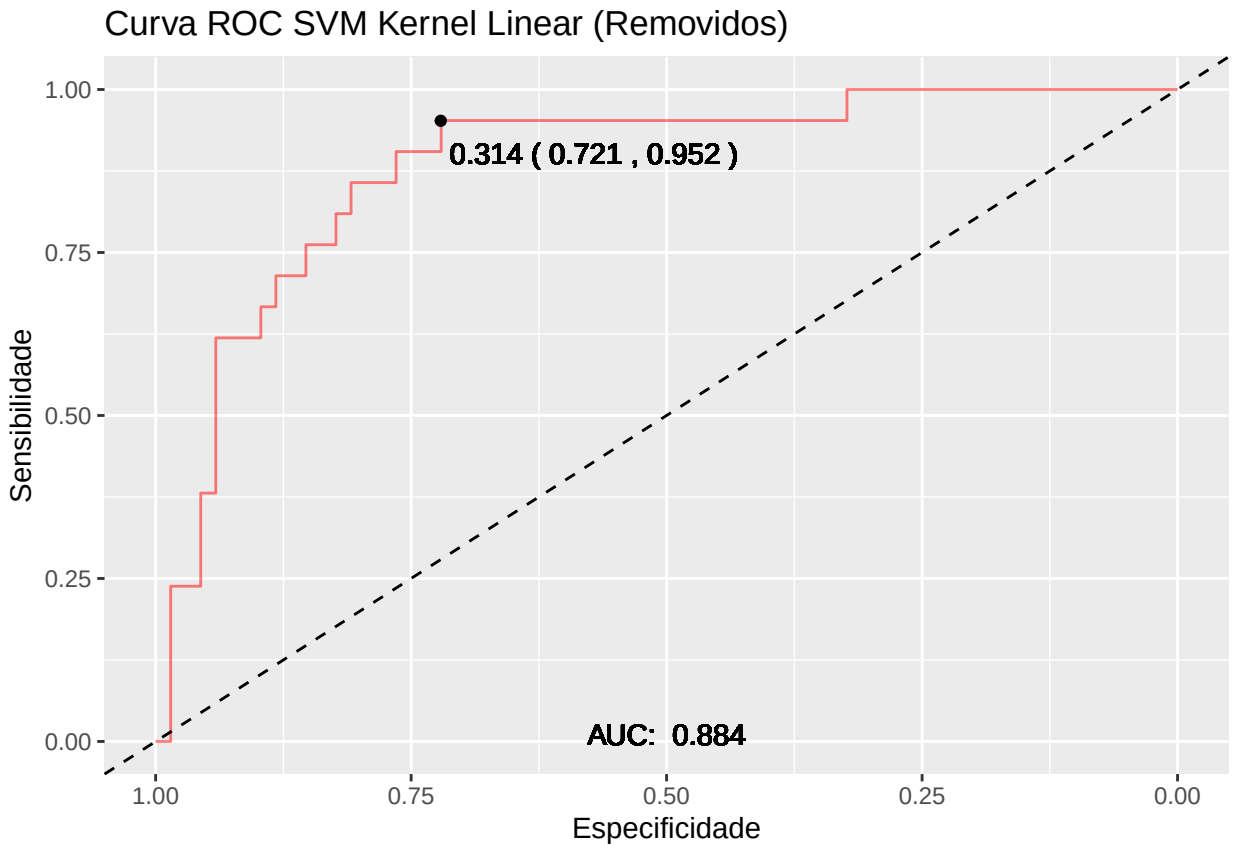
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  98  27
##        Yes 10  26
##
##                Accuracy : 0.7702
##                  95% CI : (0.6974, 0.8327)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.003815
##
##                   Kappa : 0.4334
##
##  Mcnemar's Test P-Value : 0.008529
##
##             Sensitivity : 0.9074
##             Specificity : 0.4906
##          Pos Pred Value : 0.7840
##          Neg Pred Value : 0.7222
##              Prevalence : 0.6708
##          Detection Rate : 0.6087
```

```
##    Detection Prevalence : 0.7764
##       Balanced Accuracy : 0.6990
##
##        'Positive' Class : No
##
```

```
## SVM Kernel Linear sem missing
rocSVM2 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloSVM2, test_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocSVM2, titulo = "Curva ROC SVM Kernel Linear (Removidos)")
```




```
predictSVM2 <- predict(modeloSVM2, newdata = test_without_NAs)
confusionMatrix(predictSVM2, test_without_NAs$diabetes)
```

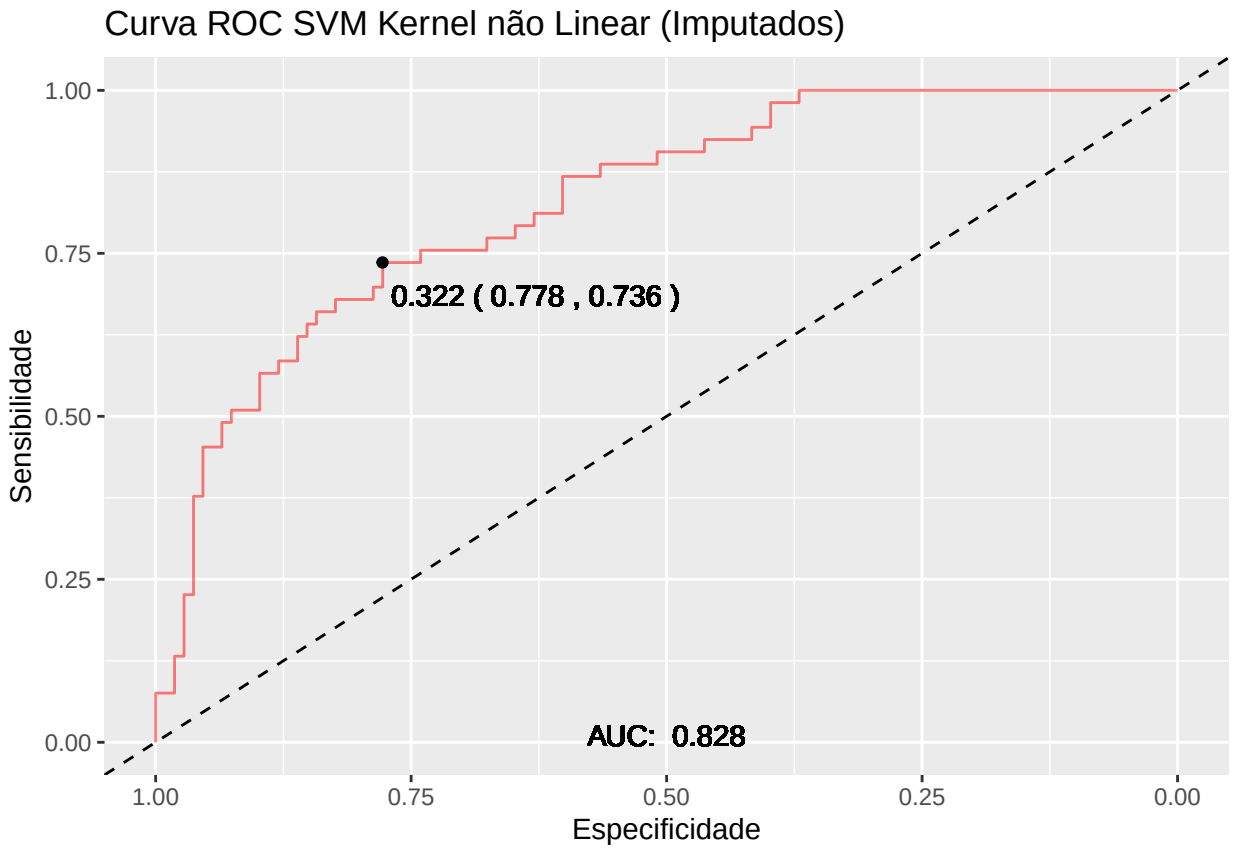
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  64  12
##        Yes  4   9
```

```
##
##                Accuracy : 0.8202
##                  95% CI : (0.7245, 0.8936)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.12910
##
##                   Kappa : 0.4258
##
##  Mcnemar's Test P-Value : 0.08012
##
##             Sensitivity : 0.9412
##             Specificity : 0.4286
##          Pos Pred Value : 0.8421
##          Neg Pred Value : 0.6923
##              Prevalence : 0.7640
##          Detection Rate : 0.7191
##    Detection Prevalence : 0.8539
##       Balanced Accuracy : 0.6849
##
##        'Positive' Class : No
##
```

```
## SVM Kernel não Linear com dados imputados
rocSVM3 <- roc(response = test$diabetes, predictor = predict(modeloSVM3, test, type = "prob")[,2])
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocSVM3, titulo = "Curva ROC SVM Kernel não Linear (Imputados)")
```




```
predictSVM3 <- predict(modeloSVM3, newdata = test)
confusionMatrix(predictSVM3, test$diabetes)
```

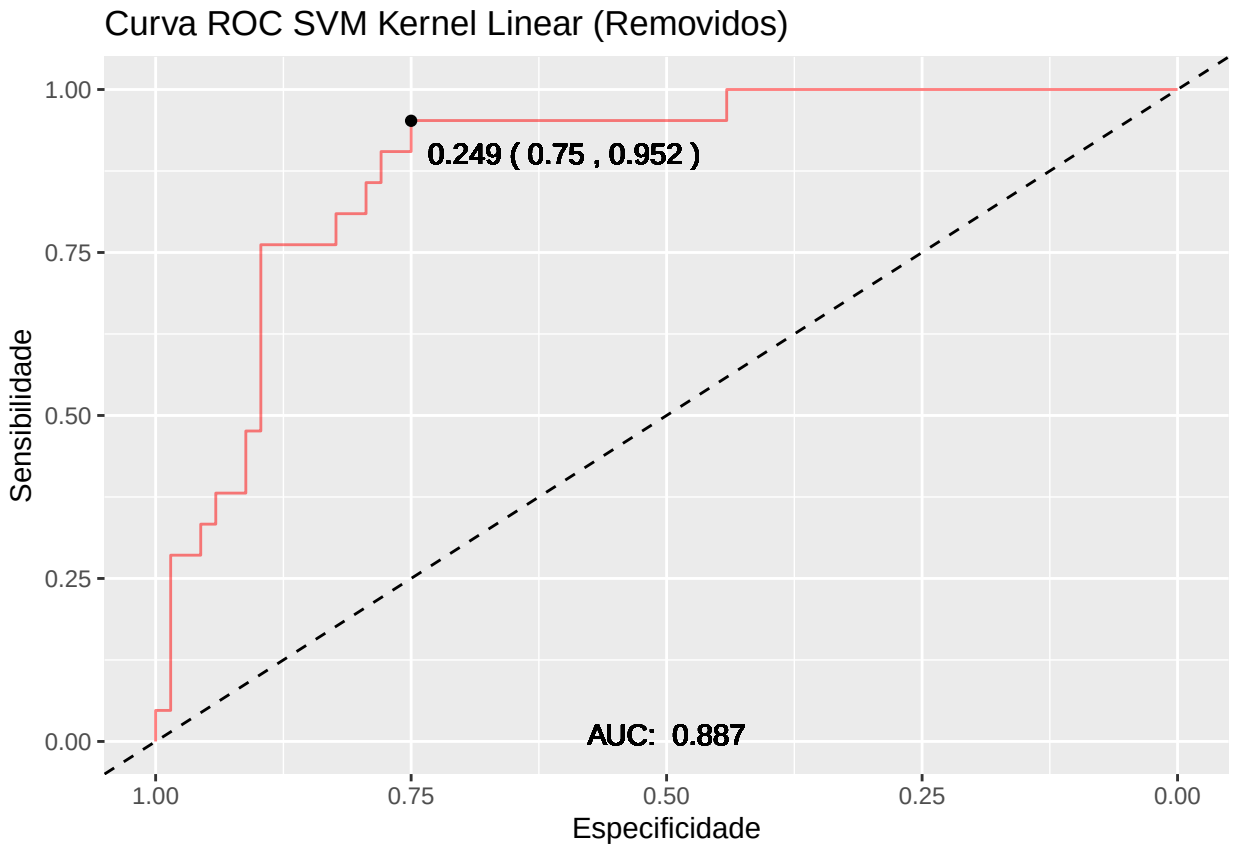
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  99  26
##        Yes  9  27
##
##                Accuracy : 0.7826
##                  95% CI : (0.7109, 0.8437)
##     No Information Rate : 0.6708
##     P-Value [Acc > NIR] : 0.001230
##
##                   Kappa : 0.464
##
##  Mcnemar's Test P-Value : 0.006841
##
##             Sensitivity : 0.9167
##             Specificity : 0.5094
##          Pos Pred Value : 0.7920
##          Neg Pred Value : 0.7500
##              Prevalence : 0.6708
##          Detection Rate : 0.6149
```

```
##    Detection Prevalence : 0.7764
##       Balanced Accuracy : 0.7131
##
##        'Positive' Class : No
##
```

```
## SVM Kernel Linear sem missing
rocSVM4 <- roc(response = test_without_NAs$diabetes, predictor = predict(modeloSVM4, test_without_NAs,
```

```
## Setting levels: control = No, case = Yes
## Setting direction: controls < cases
```

```
plotaroc(rocSVM4, titulo = "Curva ROC SVM Kernel Linear (Removidos)")
```




```
predictSVM4 <- predict(modeloSVM4, newdata = test_without_NAs)
confusionMatrix(predictSVM4, test_without_NAs$diabetes)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
##        No  63  13
##        Yes  5   8
```

```
##
##                Accuracy : 0.7978
##                  95% CI : (0.6993, 0.8755)
##     No Information Rate : 0.764
##     P-Value [Acc > NIR] : 0.27088
##
##                   Kappa : 0.354
##
##  Mcnemar's Test P-Value : 0.09896
##
##             Sensitivity : 0.9265
##             Specificity : 0.3810
##          Pos Pred Value : 0.8289
##          Neg Pred Value : 0.6154
##              Prevalence : 0.7640
##          Detection Rate : 0.7079
##    Detection Prevalence : 0.8539
##       Balanced Accuracy : 0.6537
##
##        'Positive' Class : No
##
```