Integrador1

##bibliotecas

```
library(tidyverse)
library(rsample)
library(skimr)
library(rpart)
library(rpart.plot)
library(modeldata)
library(glmnet)
library(plotmo)
library(glmnet)
library(partykit)
library(GGally)
library(ranger)
library(pROC)
library(janitor)
library(tidymodels)
library(vip)
library(gbm)
library(xgboost)
library(randomForest)
library(mlbench)
library(caret)
library(pdp)
library(gridExtra)
```

Base

```
dados <- read.csv('resultados.csv', encoding = "UTF-8", row.names = "comp_id") %>% clean_names()
```

#tratando a base

```
dados[,c('x','balsheet_flag', 'balsheet_length', 'balsheet_notfullyear','nace_main','ind','founded_date')] <- l
ist(NULL)

names <- c('gender', 'origin', 'ind2', 'urban_m', 'region_m', 'default')
dados[,names] <- lapply(dados[,names] , factor)

dados$ind2 <- dados$ind2 %>% fct_lump(prop = .05)

dados[dados==""]<-NA

dados <- dados %>%
  mutate_if(is.numeric, function(x) ifelse(is.na(x), median(x, na.rm = T), x))

skim(dados)
```

Data summary

Name	dados
Number of rows	21717
Number of columns	31
_______________________	
Column type frequency:	
factor	6
numeric	25

Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	2007	0.91	FALSE	3	mal: 13409, fem: 3960, mix: 2341, emp: 0
origin	2007	0.91	FALSE	3	Dom: 17529, For: 1580, mix: 601, emp: 0
ind2	5	1.00	FALSE	5	56: 12709, Oth: 2843, 55: 2264, 28: 1952
urban_m	0	1.00	FALSE	3	3: 8712, 1: 6955, 2: 6050
region_m	59	1.00	FALSE	3	Cen: 12933, Eas: 5458, Wes: 3267, emp: 0
default	0	1.00	FALSE	2	0: 17251, 1: 4466

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
amort_log	0	1	2.60	1.45	0.00	2.06	2.93	3.59	6.55	
curr_assets_log	0	1	4.05	0.92	0.00	3.51	4.05	4.59	7.32	
curr_liab_log	0	1	4.05	1.02	0.00	3.55	4.15	4.67	7.62	
extra_exp_log	0	1	0.19	0.74	0.00	0.00	0.00	0.00	7.23	
extra_inc_log	0	1	0.26	0.94	0.00	0.00	0.00	0.00	7.23	
extra_profit_loss_log	0	1	0.23	0.89	0.00	0.00	0.00	0.00	6.51	
fixed_assets_log	0	1	3.10	1.98	0.00	1.85	3.63	4.57	8.02	
inc_bef_tax_log	0	1	3.53	0.66	0.00	3.43	3.51	3.59	6.64	
intang_assets_log	0	1	0.45	1.16	0.00	0.00	0.00	0.00	6.75	
inventories_log	0	1	2.33	1.88	0.00	0.00	3.04	3.84	7.24	
liq_assets_log	0	1	3.25	1.06	0.00	2.63	3.29	3.95	6.89	
material_exp_log	0	1	4.47	0.83	0.00	3.99	4.49	4.97	7.15	
personnel_exp_log	0	1	3.76	1.34	0.00	3.60	4.04	4.47	6.90	
profit_loss_year_log	0	1	3.26	0.86	0.00	3.24	3.32	3.40	6.59	
sales_log	0	1	4.65	0.74	3.00	4.16	4.62	5.11	7.00	
share_eq_log	0	1	4.27	0.71	0.00	4.06	4.24	4.46	7.50	
subscribed_cap_log	0	1	3.53	0.83	0.00	3.27	3.27	4.05	7.30	
tang_assets_log	0	1	3.05	1.98	0.00	0.00	3.59	4.54	8.02	
ceo_count	0	1	1.26	0.52	1.00	1.00	1.00	1.00	15.00	
foreign	0	1	0.09	0.27	0.00	0.00	0.00	0.00	1.00	
female	0	1	0.24	0.39	0.00	0.00	0.00	0.50	1.00	
birth_year	0	1	1965.87	10.01	1920.00	1960.00	1967.00	1972.00	2015.00	
inoffice_days	0	1	2983.55	1648.37	10.00	1926.00	2598.75	3496.00	10041.00	
labor_avg	0	1	0.56	1.47	0.08	0.12	0.23	0.43	42.12	
company_age	0	1	8.62	6.62	0.00	3.00	7.00	14.00	34.00	

#retirando os NAs dos fatores

```
replace_factor_na <- function(x){
  x <- as.character(x)
  x <- if_else(is.na(x), "None", x)
  x <- as.factor(x)
}

dados <- dados %>%
  mutate_if(is.factor, replace_factor_na)

skim(dados)
```

Data summary

Name	dados
Number of rows	21717
Number of columns	31
_______________________	
Column type frequency:	
factor	6
numeric	25
________________________	
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	4	mal: 13409, fem: 3960, mix: 2341, Non: 2007
origin	0	1	FALSE	4	Dom: 17529, Non: 2007, For: 1580, mix: 601
ind2	0	1	FALSE	6	56: 12709, Oth: 2843, 55: 2264, 28: 1952
urban_m	0	1	FALSE	3	3: 8712, 1: 6955, 2: 6050
region_m	0	1	FALSE	4	Cen: 12933, Eas: 5458, Wes: 3267, Non: 59
default	0	1	FALSE	2	0: 17251, 1: 4466

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
amort_log	0	1	2.60	1.45	0.00	2.06	2.93	3.59	6.55	
curr_assets_log	0	1	4.05	0.92	0.00	3.51	4.05	4.59	7.32	
curr_liab_log	0	1	4.05	1.02	0.00	3.55	4.15	4.67	7.62	
extra_exp_log	0	1	0.19	0.74	0.00	0.00	0.00	0.00	7.23	
extra_inc_log	0	1	0.26	0.94	0.00	0.00	0.00	0.00	7.23	
extra_profit_loss_log	0	1	0.23	0.89	0.00	0.00	0.00	0.00	6.51	
fixed_assets_log	0	1	3.10	1.98	0.00	1.85	3.63	4.57	8.02	
inc_bef_tax_log	0	1	3.53	0.66	0.00	3.43	3.51	3.59	6.64	
intang_assets_log	0	1	0.45	1.16	0.00	0.00	0.00	0.00	6.75	
inventories_log	0	1	2.33	1.88	0.00	0.00	3.04	3.84	7.24	
liq_assets_log	0	1	3.25	1.06	0.00	2.63	3.29	3.95	6.89	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
material_exp_log	0	1	4.47	0.83	0.00	3.99	4.49	4.97	7.15	
personnel_exp_log	0	1	3.76	1.34	0.00	3.60	4.04	4.47	6.90	
profit_loss_year_log	0	1	3.26	0.86	0.00	3.24	3.32	3.40	6.59	
sales_log	0	1	4.65	0.74	3.00	4.16	4.62	5.11	7.00	
share_eq_log	0	1	4.27	0.71	0.00	4.06	4.24	4.46	7.50	
subscribed_cap_log	0	1	3.53	0.83	0.00	3.27	3.27	4.05	7.30	
tang_assets_log	0	1	3.05	1.98	0.00	0.00	3.59	4.54	8.02	
ceo_count	0	1	1.26	0.52	1.00	1.00	1.00	1.00	15.00	
foreign	0	1	0.09	0.27	0.00	0.00	0.00	0.00	1.00	
female	0	1	0.24	0.39	0.00	0.00	0.00	0.50	1.00	
birth_year	0	1	1965.87	10.01	1920.00	1960.00	1967.00	1972.00	2015.00	
inoffice_days	0	1	2983.55	1648.37	10.00	1926.00	2598.75	3496.00	10041.00	
labor_avg	0	1	0.56	1.47	0.08	0.12	0.23	0.43	42.12	
company_age	0	1	8.62	6.62	0.00	3.00	7.00	14.00	34.00	

#conjuntos treinamento e teste

```
set.seed(1234)

splits <- initial_split(dados, prop = .8, strata = "default")

tr <- training(splits)
teste <- testing(splits)
```

#Colocando no formato matriz para o GLMNET

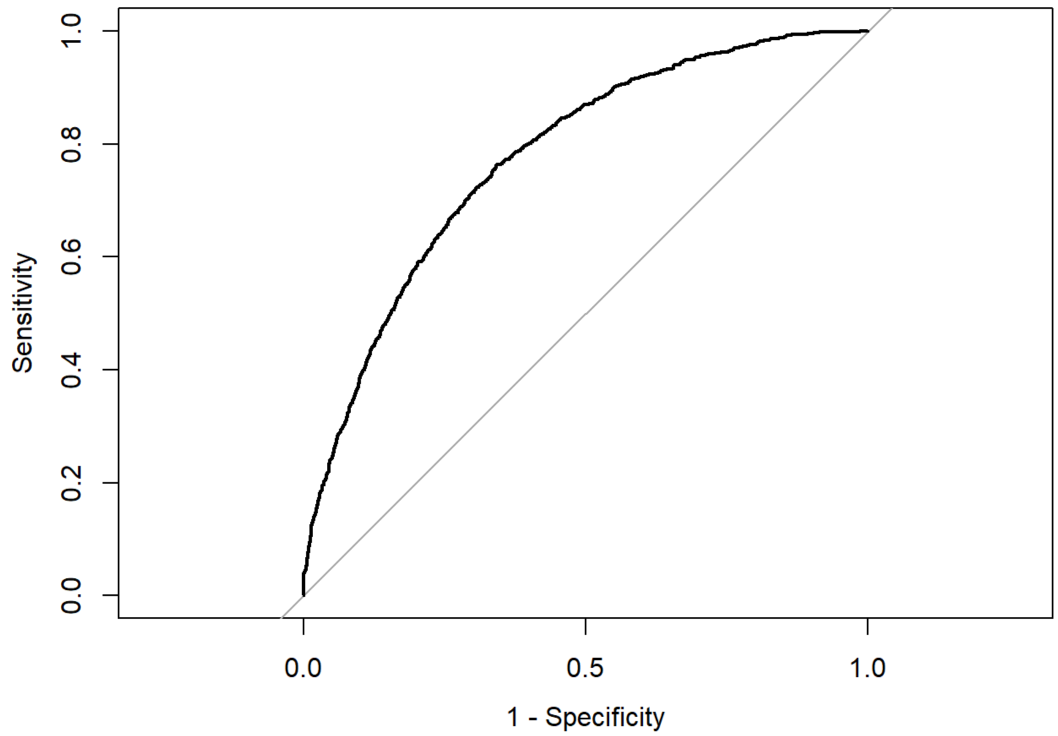
```
X <- model.matrix(default~., data = dados)[,-1]
Y <- dados$default

X_tr <- model.matrix(default~., data = tr)[,-1]
X_teste <- model.matrix(default~., data = teste)[,-1]

y_tr <- tr$default
Y_teste <- teste$default
```

#tabela de resultados

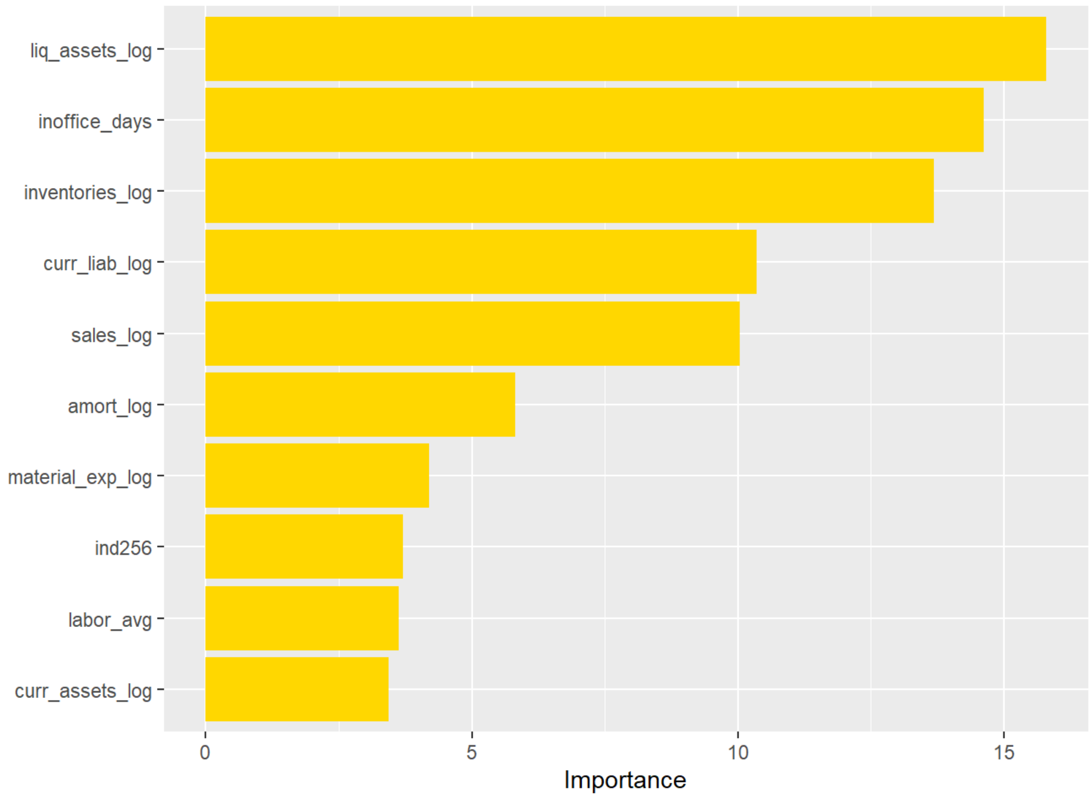
```
tab_AUC <- tibble(metodo = c("log", "ridge", "lasso", "árvore", "forest"),
              AUC = NA)
```

#regressão logística

```
fit_log <- glm(default ~., family = "binomial", data = tr)

prob_log <- predict(fit_log, teste, type = "response")

roc_log <- roc(teste$default, prob_log)

plot(roc_log, legacy.axes = TRUE)
```




```
vip(fit_log, aesthetics = list(fill = "gold"))
```

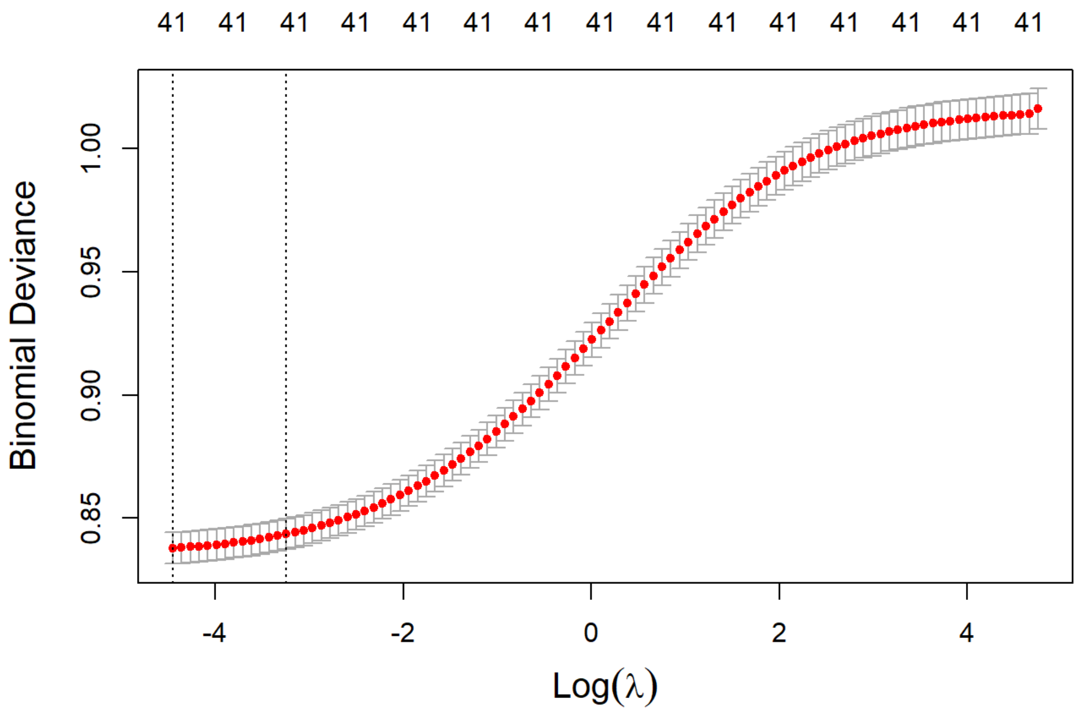



```
tab_AUC$AUC[tab_AUC$metodo == "log"] <- roc_log$auc
tab_AUC
```

```
## # A tibble: 5 × 2
##   metodo    AUC
##   <chr>   <dbl>
## 1 log     0.776
## 2 ridge  NA
## 3 lasso  NA
## 4 árvore NA
## 5 forest NA
```

#ridge

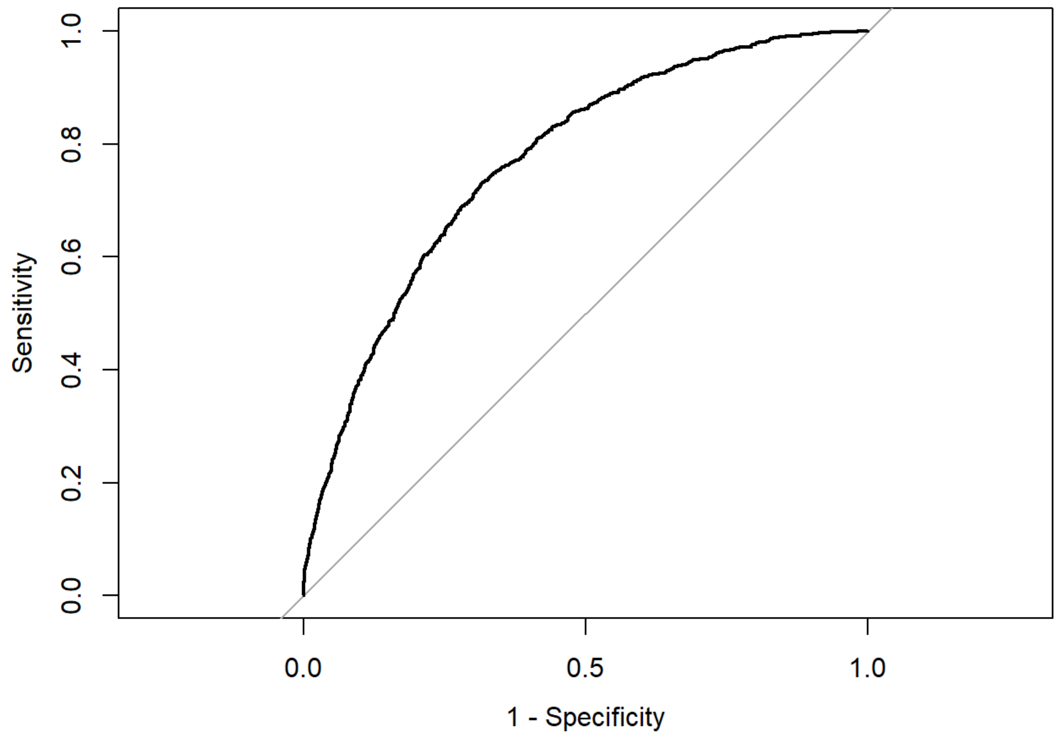
```
fit_ridge <- glmnet(X_tr, y_tr, family = "binomial", alpha = 0, lambda = 500)

cv_ridge <- cv.glmnet(X_tr, y_tr, family = "binomial", alpha = 0)

plot(cv_ridge, cex.lab = 1.3)
```



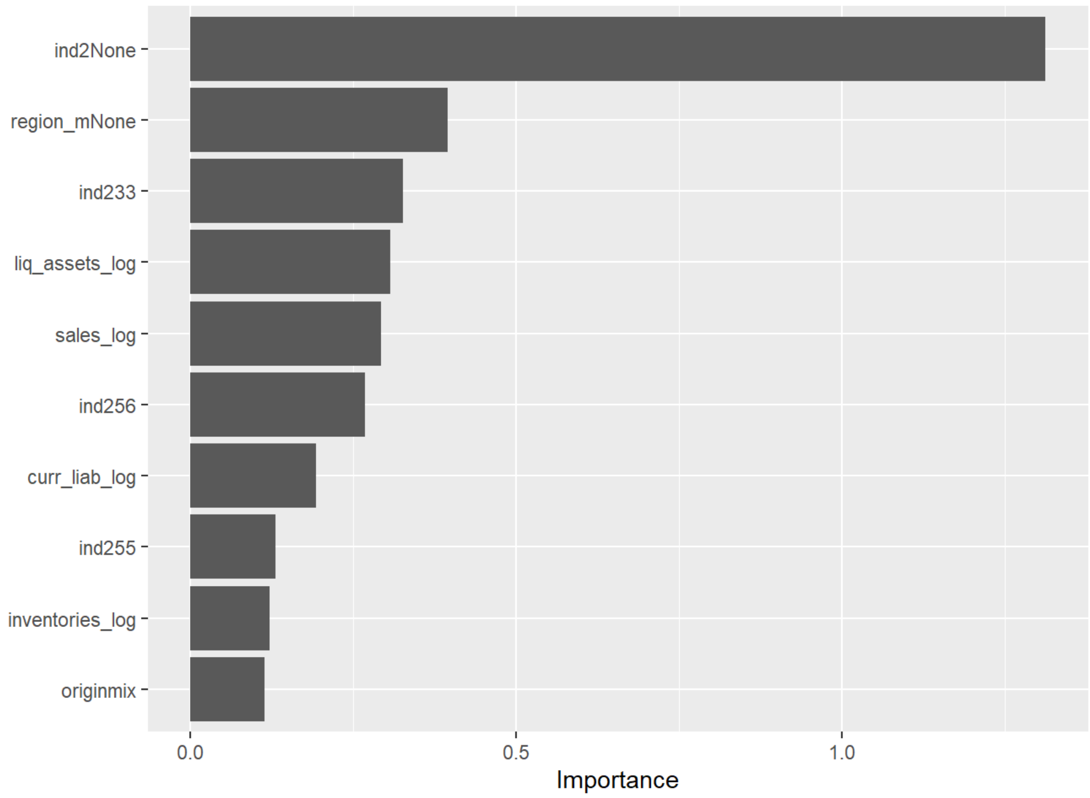

```
cv_ridge$lambda.1se
```

```
## [1] 0.0389424
```

```
fit_ridge <- glmnet(X_tr, y_tr, family = "binomial", alpha = 0, lambda = cv_ridge$lambda.1se)

prob_ridge <- predict(fit_ridge, newx = X_teste, s = cv_ridge$lambda.1se, type = "response" )

roc_ridge <- roc(Y_teste, prob_ridge)

plot(roc_ridge, legacy.axes = TRUE)
```




```
vip(fit_ridge)
```

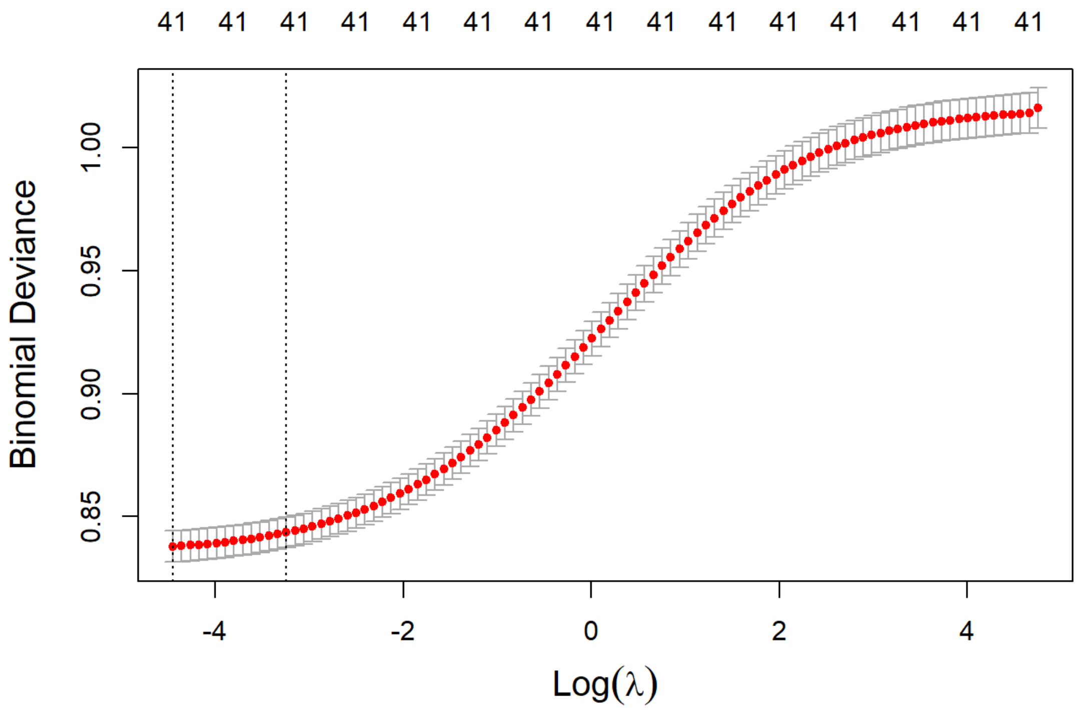



```
tab_AUC$AUC[tab_AUC$metodo == "ridge"] <- roc_ridge$auc
tab_AUC
```

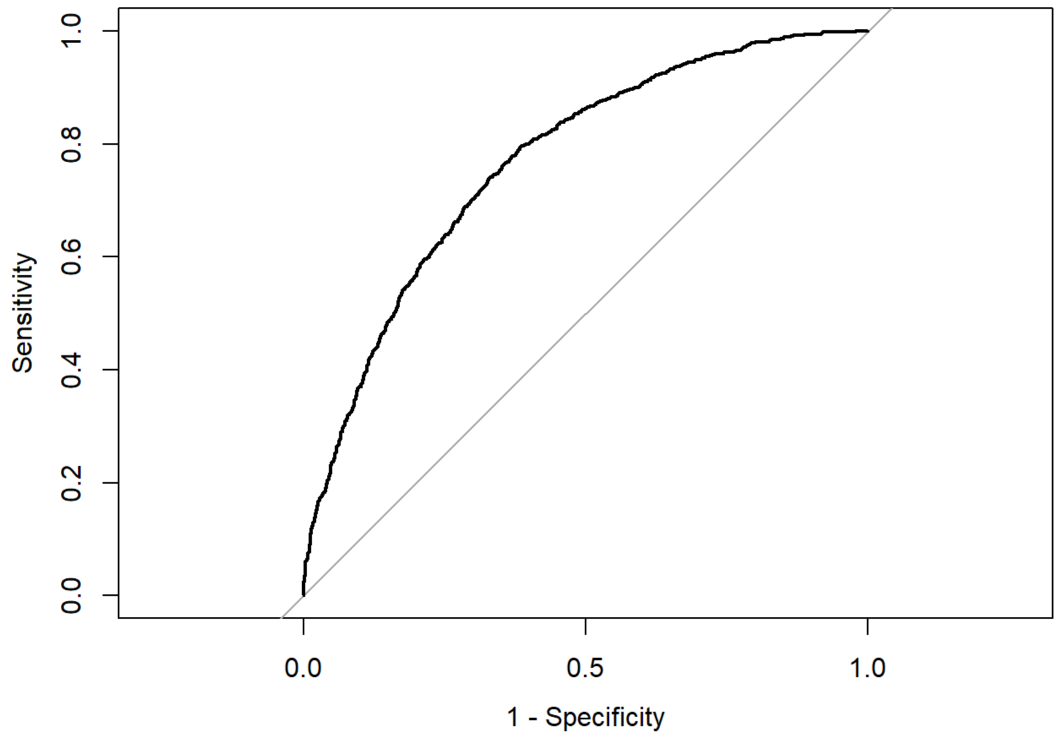
```
## # A tibble: 5 × 2
##   metodo    AUC
##   <chr>   <dbl>
## 1 log     0.776
## 2 ridge   0.771
## 3 lasso  NA
## 4 árvore NA
## 5 forest NA
```

#lasso

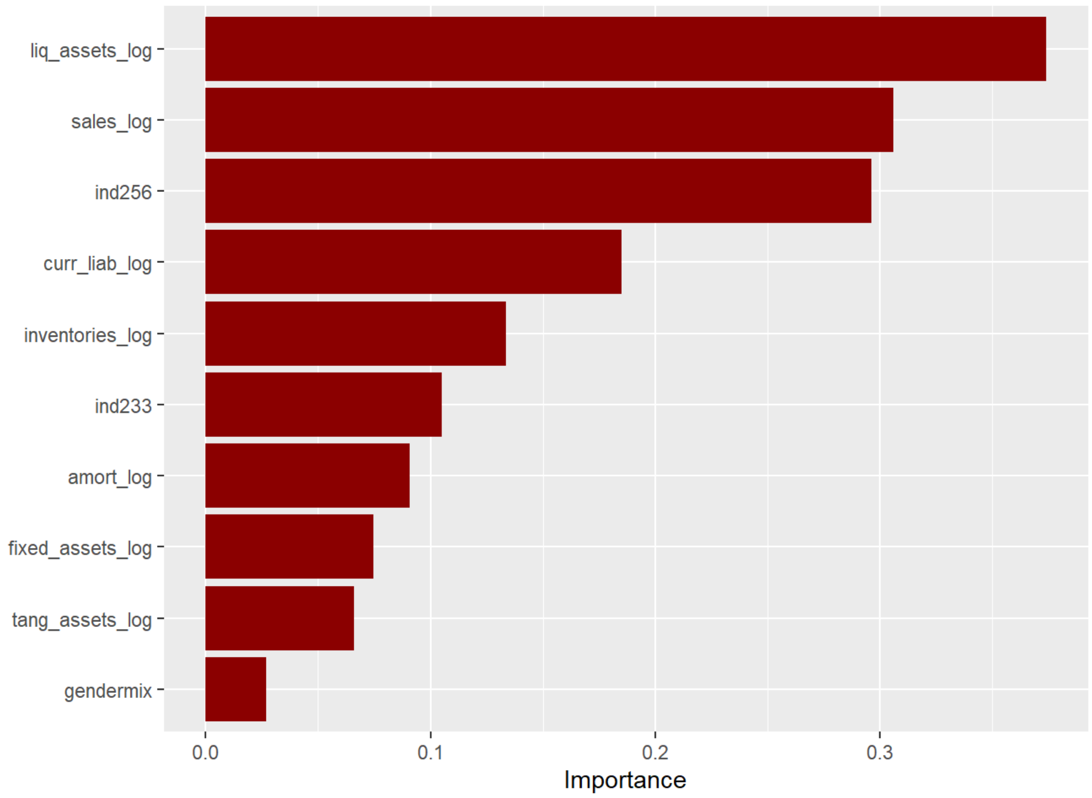
```
fit_lasso <- glmnet(X_tr, y_tr, family = "binomial", alpha = 1, lambda = 500)

cv_lasso <- cv.glmnet(X_tr, y_tr, family = "binomial", alpha = 1)
plot(cv_ridge, cex.lab = 1.3)
```




```
fit_lasso <- glmnet(X_tr, y_tr, family = "binomial", alpha = 1, lambda = cv_lasso$lambda.1se)

prob_lasso <- predict(fit_lasso, newx = X_teste, s = cv_lasso$lambda.1se, type = "response" )

roc_lasso <- roc(Y_teste, prob_lasso)

plot(roc_lasso, legacy.axes = TRUE)
```




```
vip(fit_lasso, aesthetics = list(fill = "dark red"))
```




```
tab_AUC$AUC[tab_AUC$metodo == "lasso"] <- roc_lasso$auc
tab_AUC
```

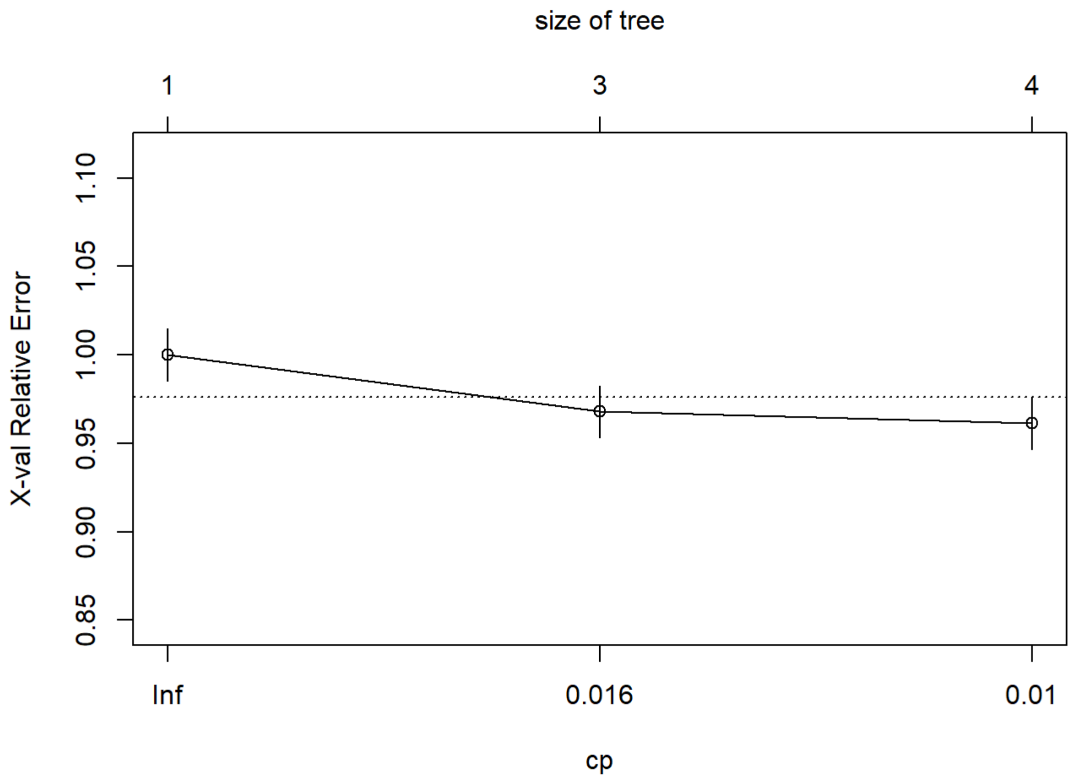
```
## # A tibble: 5 × 2
##   metodo    AUC
##   <chr>   <dbl>
## 1 log     0.776
## 2 ridge   0.771
## 3 lasso   0.769
## 4 árvore NA
## 5 forest NA
```

#arvore

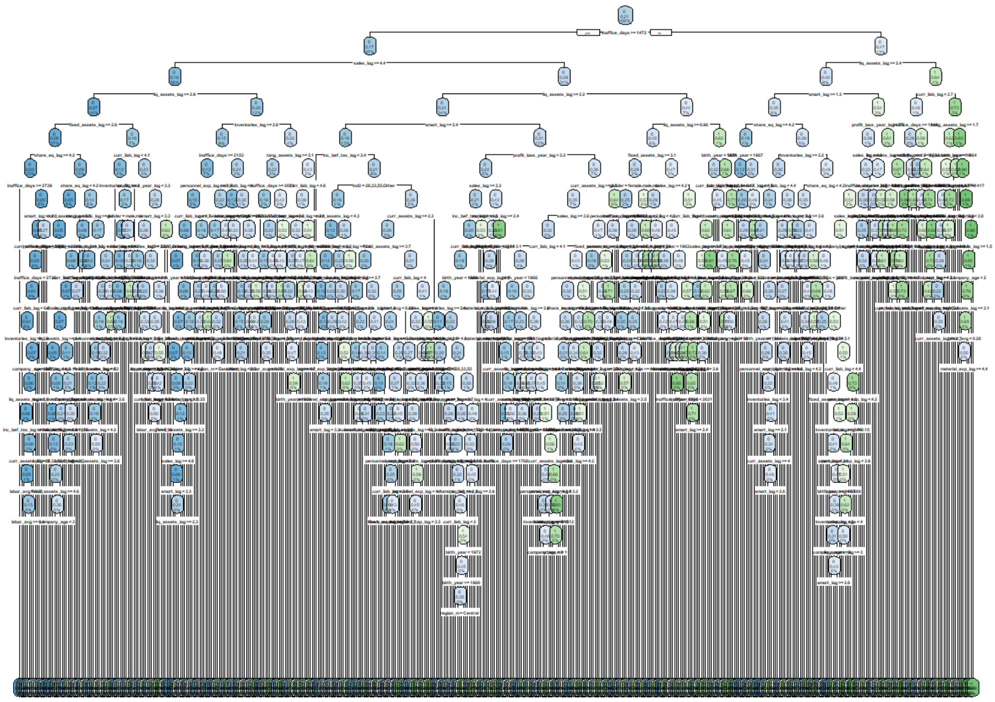
```
arvore <- rpart(default ~., tr)

arvore$cptable
```

```
##           CP nsplit rel error    xerror       xstd
## 1 0.02547592      0 1.0000000 1.0000000 0.01491279
## 2 0.01007839      2 0.9490482 0.9678052 0.01473177
## 3 0.01000000      3 0.9389698 0.9613662 0.01469481
```

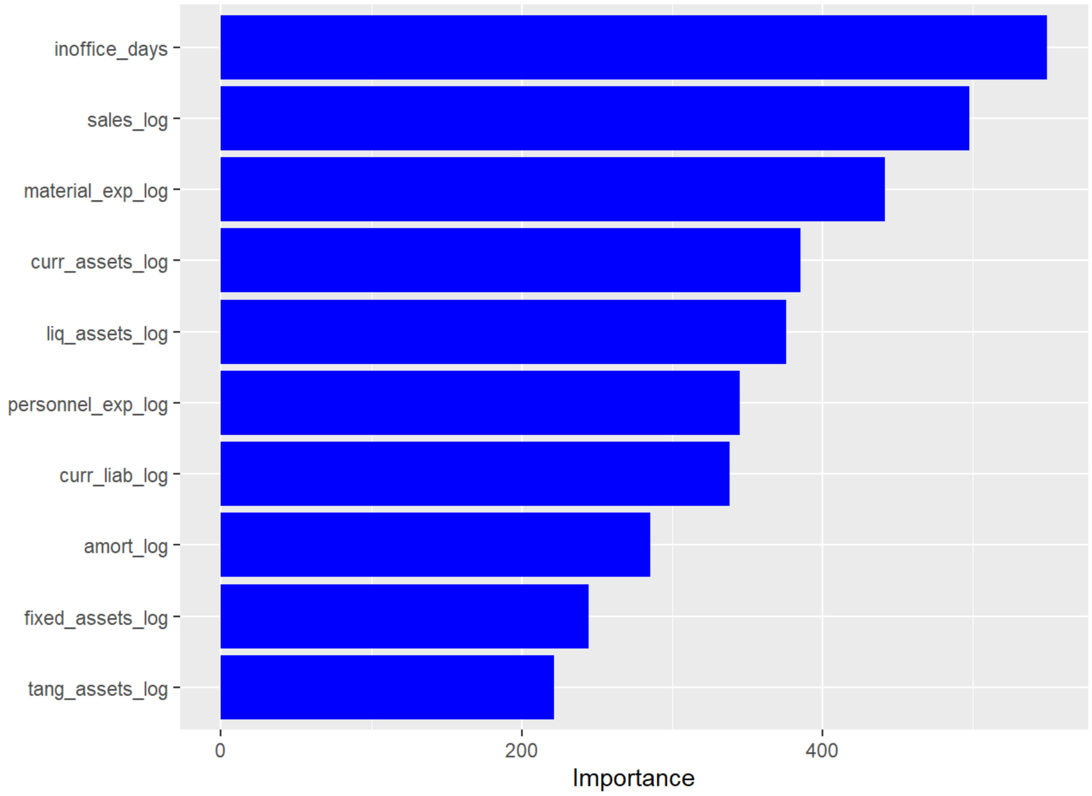
```
plotcp(arvore)
```




```
arvore <- rpart(default ~., tr, control = rpart.control(cp = 0))

rpart.plot(arvore)
```



```
prob_arvore <- predict(arvore, teste, type = "prob")[,1]

roc_arvore <- roc(Y_teste, prob_arvore)

vip::vip(arvore, aesthetics = list(fill = "blue"))
```




```
tab_AUC$AUC[tab_AUC$metodo == "árvore"] <- roc_arvore$auc
tab_AUC
```

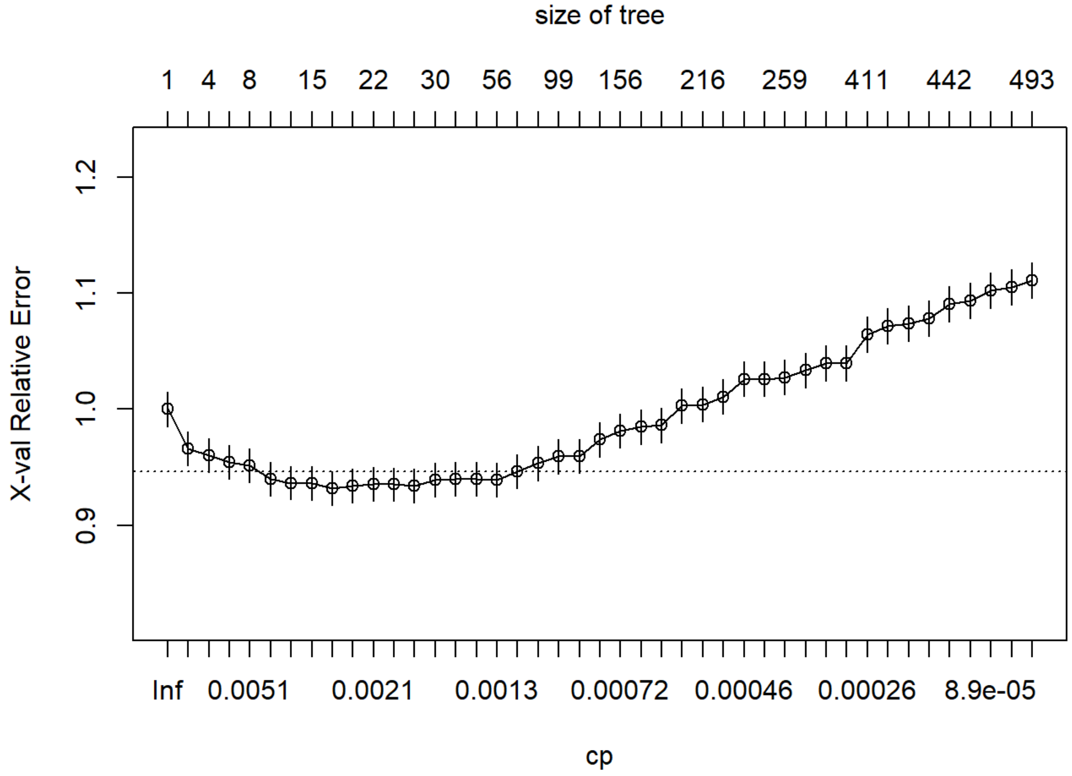
```
## # A tibble: 5 × 2
##   metodo    AUC
##   <chr>   <dbl>
## 1 log     0.776
## 2 ridge   0.771
## 3 lasso   0.769
## 4 árvore  0.720
## 5 forest NA
```

#Árvore de classificação Otimizada

```
#função de teste dos hiperparâmetros
arvore$cptable
```

```
##              CP nsplit rel error    xerror       xstd
## 1  2.547592e-02      0 1.0000000 1.0000000 0.01491279
## 2  1.007839e-02      2 0.9490482 0.9661254 0.01472215
## 3  8.398656e-03      3 0.9389698 0.9599664 0.01468674
## 4  5.412467e-03      4 0.9305711 0.9540873 0.01465272
## 5  4.899216e-03      7 0.9143337 0.9512878 0.01463645
## 6  3.639418e-03      9 0.9045353 0.9395297 0.01456755
## 7  2.939530e-03     12 0.8936170 0.9364502 0.01454937
## 8  2.799552e-03     14 0.8877380 0.9361702 0.01454771
## 9  2.519597e-03     15 0.8849384 0.9316909 0.01452114
## 10 2.099664e-03     17 0.8798992 0.9339306 0.01453444
## 11 2.053005e-03     21 0.8715006 0.9353303 0.01454274
## 12 1.959686e-03     26 0.8605823 0.9350504 0.01454108
## 13 1.819709e-03     27 0.8586226 0.9339306 0.01453444
## 14 1.539754e-03     29 0.8549832 0.9389698 0.01456425
## 15 1.455767e-03     39 0.8379059 0.9398096 0.01456921
## 16 1.399776e-03     52 0.8124300 0.9400896 0.01457086
## 17 1.259798e-03     55 0.8082307 0.9392497 0.01456590
## 18 1.119821e-03     69 0.7875140 0.9462486 0.01460703
## 19 1.026502e-03     86 0.7665174 0.9532475 0.01464785
## 20 9.798432e-04     98 0.7513998 0.9591265 0.01468190
## 21 8.398656e-04    109 0.7404815 0.9596865 0.01468513
## 22 7.465472e-04    141 0.7127660 0.9736842 0.01476529
## 23 6.998880e-04    155 0.6982083 0.9812430 0.01480809
## 24 6.532288e-04    167 0.6898096 0.9846025 0.01482699
## 25 6.298992e-04    191 0.6716125 0.9862822 0.01483642
## 26 5.599104e-04    195 0.6690929 1.0030795 0.01492978
## 27 5.249160e-04    215 0.6578947 1.0041993 0.01493595
## 28 4.665920e-04    227 0.6494961 1.0106383 0.01497125
## 29 4.479283e-04    234 0.6461366 1.0260358 0.01505469
## 30 4.399296e-04    248 0.6385778 1.0260358 0.01505469
## 31 4.199328e-04    258 0.6340985 1.0274356 0.01506221
## 32 3.732736e-04    302 0.6139418 1.0335946 0.01509515
## 33 3.499440e-04    312 0.6100224 1.0394737 0.01512640
## 34 2.799552e-04    342 0.5923852 1.0394737 0.01512640
## 35 2.399616e-04    410 0.5708287 1.0646697 0.01525811
## 36 2.099664e-04    417 0.5691489 1.0713886 0.01529264
## 37 1.866368e-04    421 0.5683091 1.0741881 0.01530695
## 38 1.555307e-04    424 0.5677492 1.0783875 0.01532834
## 39 1.399776e-04    441 0.5646697 1.0907055 0.01539053
## 40 9.331840e-05    451 0.5632699 1.0937850 0.01540595
## 41 8.398656e-05    469 0.5615901 1.1024636 0.01544913
## 42 5.599104e-05    479 0.5607503 1.1052632 0.01546298
## 43 0.000000e+00    492 0.5599104 1.1111422 0.01549192
```

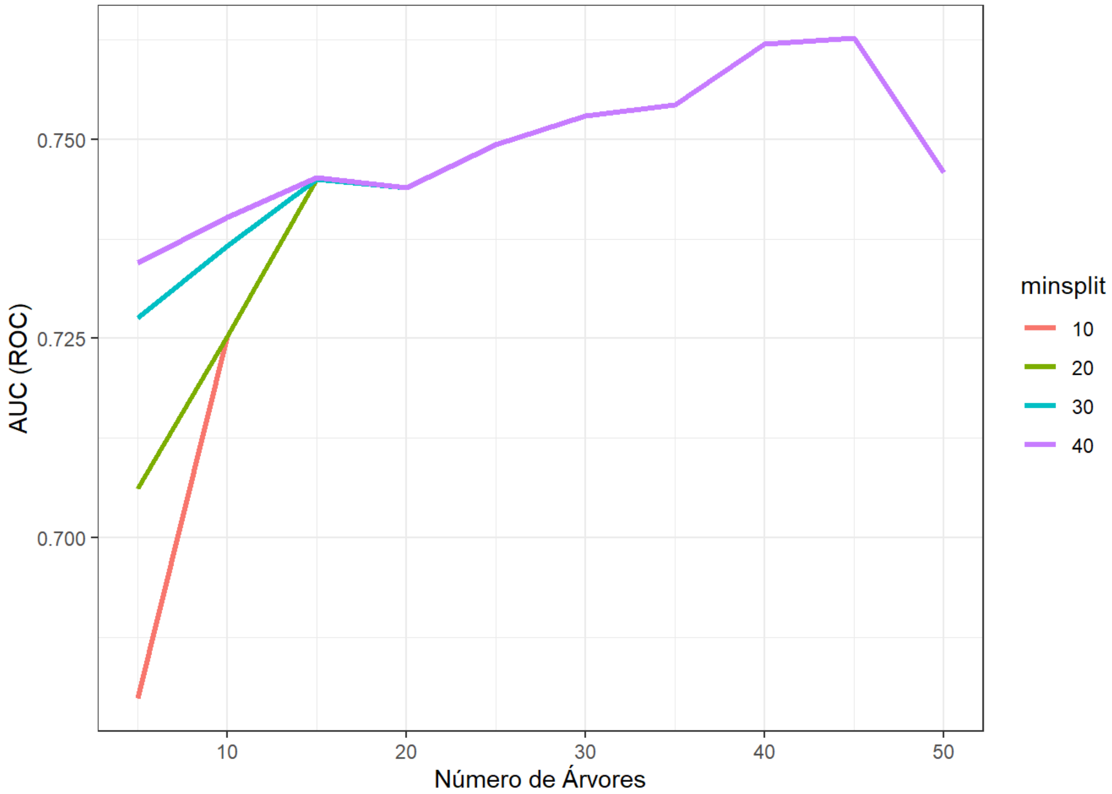
```
plotcp(arvore)
```




```
#Otimização do modelo

resultados <- crossing(minsplit = c(10, 20, 30, 40),
 min_buckets = c(seq(5, 50, 5)))
ajuste <- function(minsplit, min_buckets) {
 arvore <- rpart(default ~ ., data = tr,
               control = rpart.control(minsplit=minsplit,minbucket = min_buckets, cp = 0))
 prob_arvore <- predict(arvore, teste, type = "prob")[,1]

roc_arvore <- roc(Y_teste, prob_arvore)

 return(roc_arvore$auc)
}

resultados <- resultados %>%
  mutate(erro = map2_dbl(minsplit, min_buckets, ajuste))

head(resultados)
```

```
## # A tibble: 6 × 3
##   minsplit min_buckets  erro
##      <dbl>       <dbl> <dbl>
## 1       10           5 0.680
## 2       10          10 0.725
## 3       10          15 0.745
## 4       10          20 0.744
## 5       10          25 0.749
## 6       10          30 0.753
```

```
resultados %>%
 mutate(minsplit = factor(minsplit)) %>%
 ggplot(aes(min_buckets, erro, group = minsplit, color = minsplit)) +
 geom_line( size = 1.2) +
 labs(x = "Número de Árvores", y = "AUC (ROC)") +
 theme_bw()
```




```
#Resultados Finais

arvore <- rpart(default ~ ., data = tr,
                control = rpart.control(minsplit=20,minbucket = 40, cp = 0))

prob_arvore <- predict(arvore, teste, type = "prob")[,1]

roc_arvore <- roc(Y_teste, prob_arvore)
tab_AUC$AUC[tab_AUC$metodo == "árvore"] <- roc_arvore$auc
tab_AUC
```

```
## # A tibble: 5 × 2
##   metodo    AUC
##   <chr>   <dbl>
## 1 log     0.776
## 2 ridge   0.771
## 3 lasso   0.769
## 4 árvore  0.762
## 5 forest NA
```

#random forest

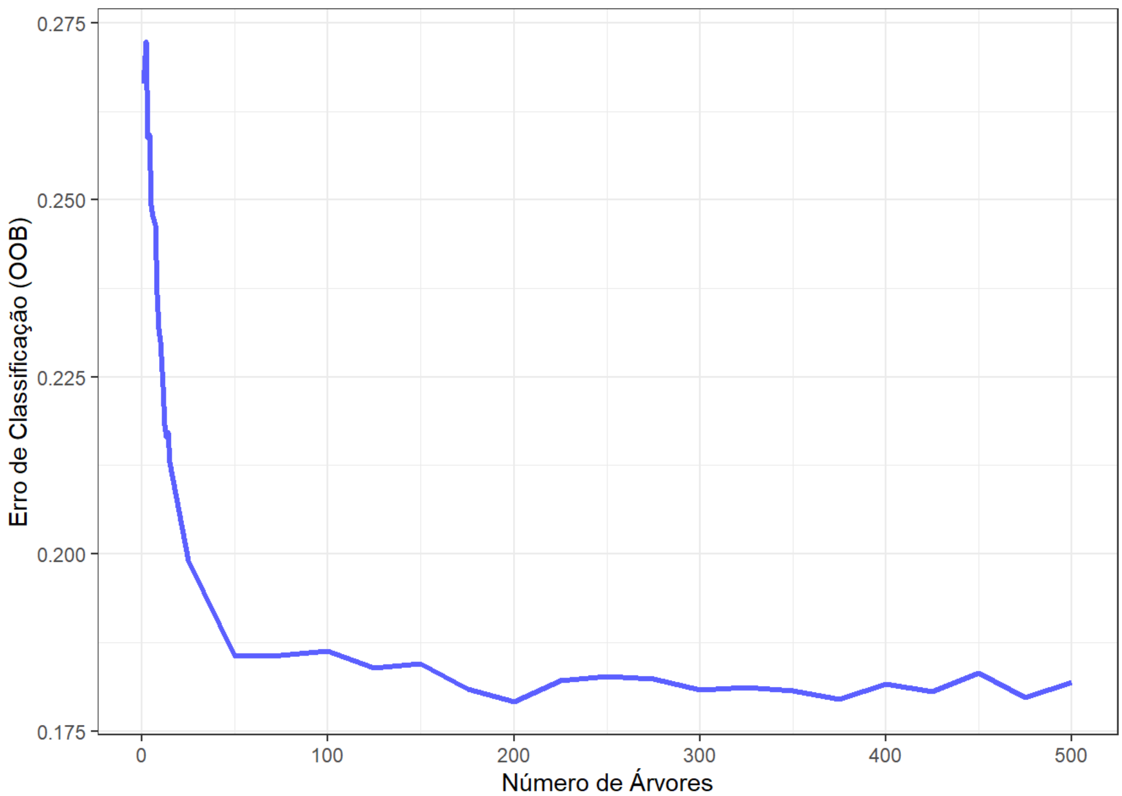
```
floresta <- ranger(default ~., data = tr)
floresta
```

```
## Ranger result
##
## Call:
##  ranger(default ~ ., data = tr)
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      17372
## Number of independent variables:  30
## Mtry:                             5
## Target node size:                 1
## Variable importance mode:         none
## Splitrule:                        gini
## OOB prediction error:             18.21 %
```

```
floresta$confusion.matrix
```

```
##      predicted
## true     0     1
##     0 13277   523
##     1  2641   931
```

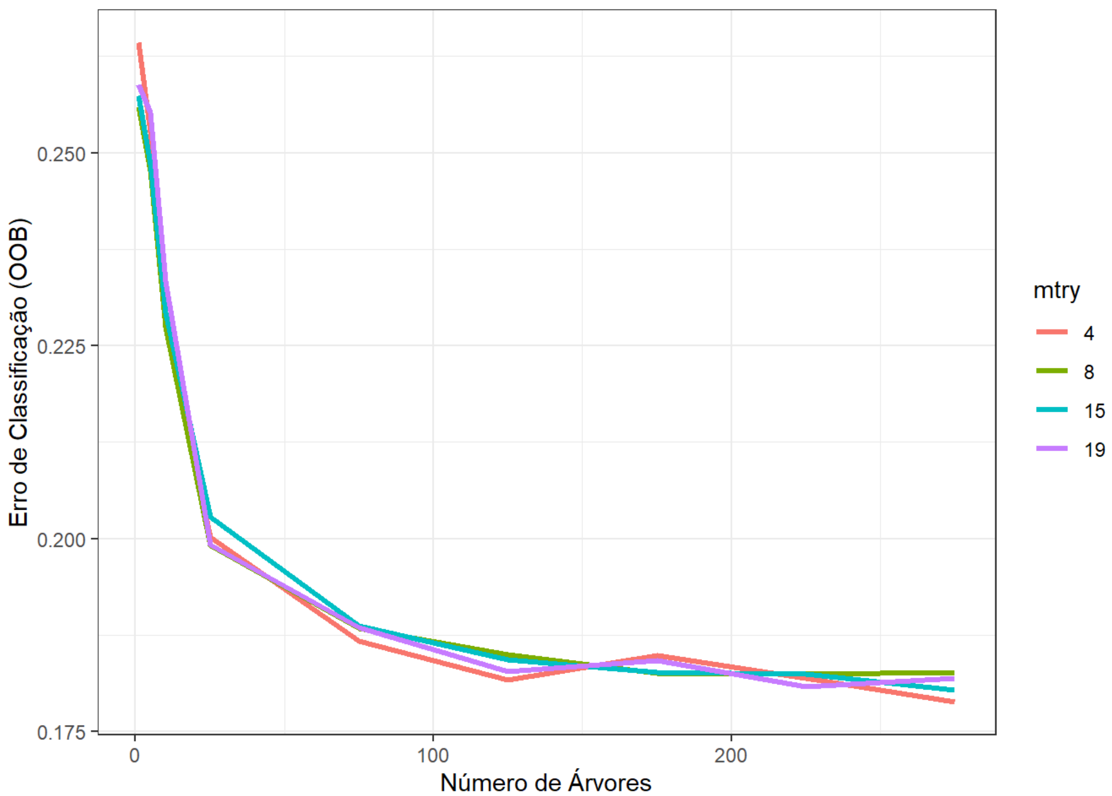
```
resultados <- tibble(n_arvores = c(1:15, seq(25, 500, 25)),
 erro = NA)

resultados <- resultados %>%
 mutate(erro = map_dbl(n_arvores, ~ranger(default ~ ., num.trees = .x,
 data = tr)$prediction.error))
resultados %>%
 ggplot(aes(n_arvores, erro)) +
 geom_line(color = "#5B5FFF", size = 1.2) +
 labs(x = "Número de Árvores", y = "Erro de Classificação (OOB)") +
 theme_bw()
```



```
resultados <- crossing(mtry = c(4, 8, 15, 19),
 n_arvores = c(1, 5, 10, seq(25, 300, 50)))
ajuste <- function(mtry, n_arvores) {
 floresta <- ranger(default ~ ., num.trees = n_arvores, mtry = mtry, data = tr)
 return(floresta$prediction.error)
}
resultados <- resultados %>%
 mutate(erro = map2_dbl(mtry, n_arvores, ajuste))
head(resultados)
```

```
## # A tibble: 6 × 3
##    mtry n_arvores  erro
##   <dbl>     <dbl> <dbl>
## 1     4         1 0.264
## 2     4         5 0.253
## 3     4        10 0.227
## 4     4        25 0.200
## 5     4        75 0.187
## 6     4       125 0.182
```

```
resultados %>%
 mutate(mtry = factor(mtry)) %>%
 ggplot(aes(n_arvores, erro, group = mtry, color = mtry)) +
 geom_line( size = 1.2) +
 labs(x = "Número de Árvores", y = "Erro de Classificação (OOB)") +
 theme_bw()
```




```
floresta <- ranger(default ~ ., num.trees = 250, mtry = 15, data = tr, probability = TRUE, importance = 'permut
ation')

prob_floresta <- predict(floresta, data = teste)

roc_floresta <- roc(teste$default, prob_floresta$predictions[,1])

class(teste$default)
```

```
## [1] "factor"
```

```
head(prob_floresta$predictions)
```

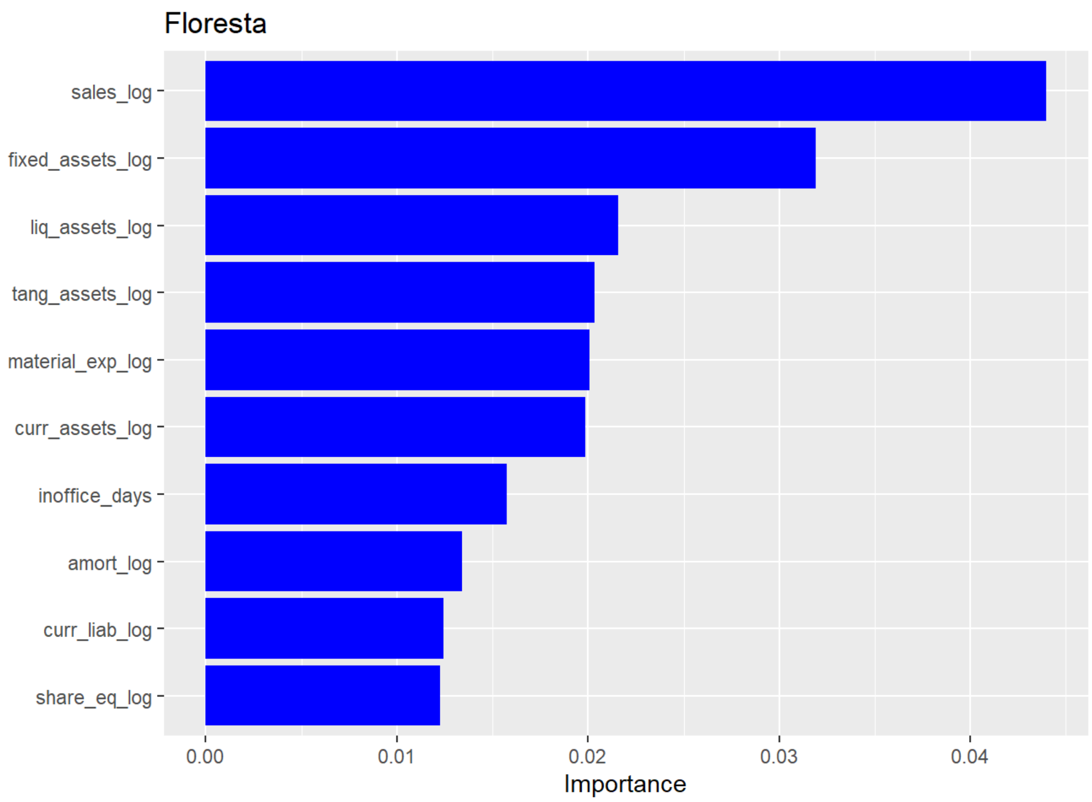
```
##              0           1
## [1,] 0.9986286 0.001371429
## [2,] 0.9226095 0.077390476
## [3,] 0.6619429 0.338057143
## [4,] 0.8997111 0.100288889
## [5,] 0.5509270 0.449073016
## [6,] 0.8750016 0.124998413
```

```
tab_AUC$AUC[tab_AUC$metodo == "forest"] <- roc_floresta$auc
tab_AUC
```

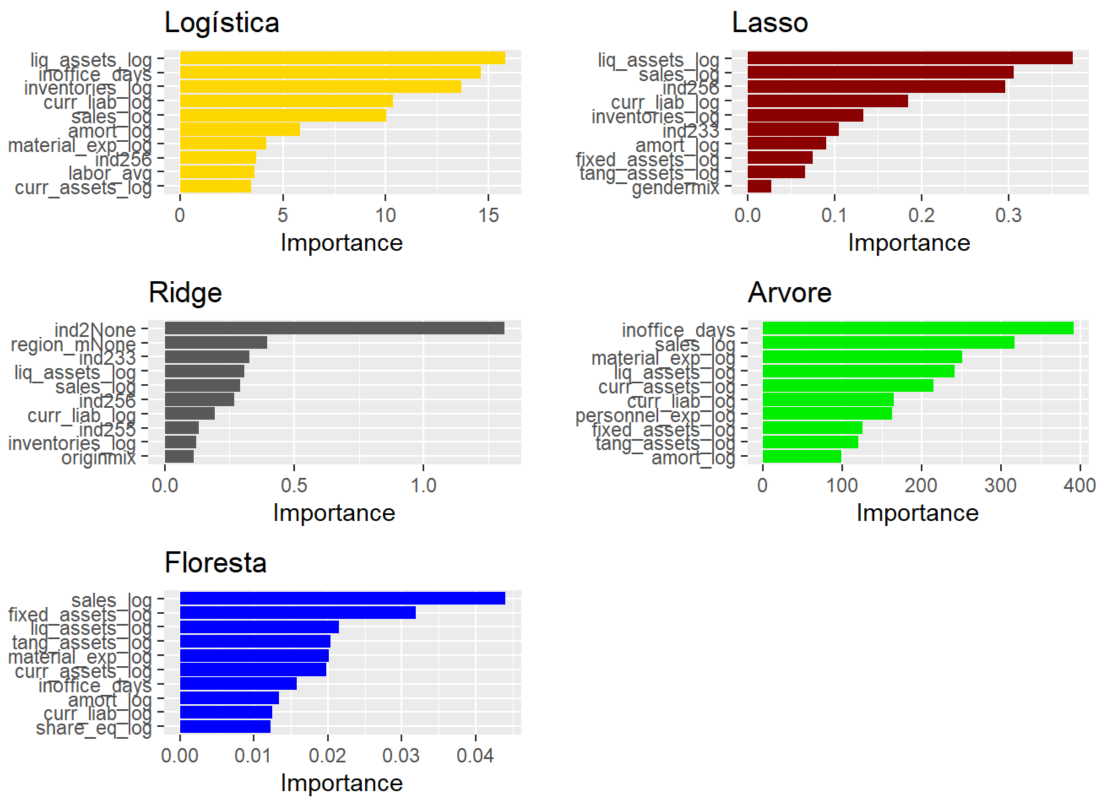
```
## # A tibble: 5 × 2
##   metodo   AUC
##   <chr>  <dbl>
## 1 log    0.776
## 2 ridge  0.771
## 3 lasso  0.769
## 4 árvore 0.762
## 5 forest 0.793
```

#Técnicas de importânica de váriavel e interpretabilidade

```
vip_floresta<-vip(floresta,aesthetics=list(fill="blue")) + ggtitle("Floresta")
vip_floresta
```



```
vip_arvore<-vip(arvore, aesthetics = list(fill = "green2")) + ggtitle("Arvore")

vip_log <- vip(fit_log, aesthetics = list(fill = "gold")) + ggtitle("Logística")

vip_ridge <- vip(fit_ridge) + ggtitle("Ridge")

vip_lasso <- vip(fit_lasso, aesthetics = list(fill = "dark red")) + ggtitle("Lasso")

grid.arrange(vip_log, vip_lasso, vip_ridge, vip_arvore, vip_floresta)
```




```
# Pacote DALEX ----------------------------------------------------------

library(DALEX)

explicador <- explain(model = floresta, #objeto modelo
                      data = dados, #base de teste - probabilidade
                      y=dados$default #base de teste - var resposta
                     )
```

```
## Preparation of a new explainer is initiated
##   -> model label       :  ranger  (  default  )
##   -> data              :  21717  rows  31  cols
##   -> target variable   :  21717  values
##   -> predict function  :  yhat.ranger  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package ranger , ver. 0.13.1 , task classification (  default  )
##   -> model_info        :  Model info detected classification task but 'y' is a factor .  (  WARNING  )
##   -> model_info        :  By deafult classification tasks supports only numercical 'y' parameter.
##   -> model_info        :  Consider changing to numerical vector with 0 and 1 values.
##   -> model_info        :  Otherwise I will not be able to calculate residuals or loss function.
##   -> predicted values  :  numerical, min =  0 , mean =  0.2121263 , max =  0.9971746
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  NA , mean =  NA , max =  NA
##   A new explainer has been created!
```

```
# PDP ----------------------------------------------------------

pdp_rf <- model_profile(explainer = explicador, variables = c("sales_log", "fixed_assets_log","liq_assets_log"
))
pdp_rf
```

```
## Top profiles    :
##            _vname_ _label_        _x_     _yhat_ _ids_
## 1 fixed_assets_log  ranger 0.0000000 0.3393492     0
## 2   liq_assets_log  ranger 0.0000000 0.3949903     0
## 3   liq_assets_log  ranger 0.5686362 0.3822739     0
## 4   liq_assets_log  ranger 1.0457575 0.3564540     0
## 5   liq_assets_log  ranger 1.2676062 0.3489418     0
## 6   liq_assets_log  ranger 1.4717262 0.3414275     0
```

```
plot(pdp_rf)
```