Lista 4 - Mineração

Victor Alves Dogo Martins, RA: 744878 Ana Beatriz Alves Monteiro, RA: 727838
Larissa Torres, RA: 631914

11-09-2022

Item A anabe

```
### Carregando pacotes

library(tidyverse)
library(knitr)
library(kableExtra)
library(patchwork)
library(rsample)
library(glmnet)
library(caret)
library(rpart)
library(rpart.plot)
library(randomForest)
library(neuralnet)
library(xgboost)
library(ROCR)
library(Matrix)
library(pdp)

### Lendo dados

df <- readr::read_csv('dados_covid.csv') |>
  rename(result=1, age_quant=2, hct=3, hgb=4,
         plat=5, mean_plat=6, rbc=7, lym=8,
         mchc=9, wbc=10, baso=11, mch=12,
         eos=13, mcv=14, mono=15, rdw=16)

### ITEM A: estimações de densidade continua das
### variaveis divididas por diagnostico

age_quant <- df |>
  ggplot()+
  aes(x=age_quant, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Quantil de Idade', y='Densidade',
       title='Densidade de age_quant',
```

```
       subtitle = 'agrupada pela variável de resultado')

hct <- df |>
  ggplot()+
  aes(x=hct, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Hematócritos', y='Densidade',
       title='Densidade de hct',
       subtitle = 'agrupada pela variável de resultado')

hgb <- df |>
  ggplot()+
  aes(x=hgb, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Hemoglobinas', y='Densidade',
       title='Densidade de hgb',
       subtitle = 'agrupada pela variável de resultado')

plat <- df |>
  ggplot()+
  aes(x=plat, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'right')+
  labs(x='Plaquetas', y='Densidade',
       title='Densidade de plat',
       fill='Resultado',
       subtitle = 'agrupada pela variável de resultado')

(age_quant+hct)/(hgb+plat)
```

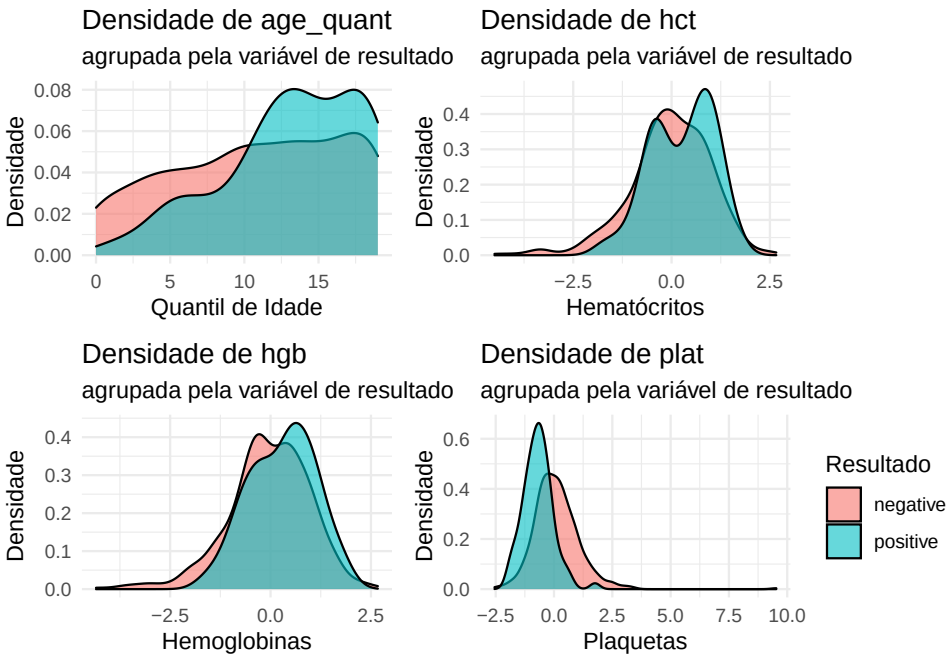

Figure 1: Densidade das variáveis agequant, hct, hgb e plat agrupadas por diagnóstico.

```
mean_plat <- df |>
  ggplot()+
  aes(x=mean_plat, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Média de Plaquetas', y='Densidade',
       title='Densidade de mean_plat',
       subtitle = 'agrupada pela variável de resultado')

rbc <- df |>
  ggplot()+
  aes(x=rbc, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Células Vermelhas', y='Densidade',
       title='Densidade de rbc',
       subtitle = 'agrupada pela variável de resultado')

lym <- df |>
  ggplot()+
  aes(x=lym, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Linfócitos', y='Densidade',
       title='Densidade de lym',
       subtitle = 'agrupada pela variável de resultado')
```

```
mchc <- df |>
  ggplot()+
  aes(x=mchc, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'right')+
  labs(x='Concentração Corpuscular Média de Hemoglobina', y='Densidade',
       title='Densidade de mchc',
       fill='Resultado',
       subtitle = 'agrupada pela variável de resultado')

(mean_plat+rbc)/(lym+mchc)
```

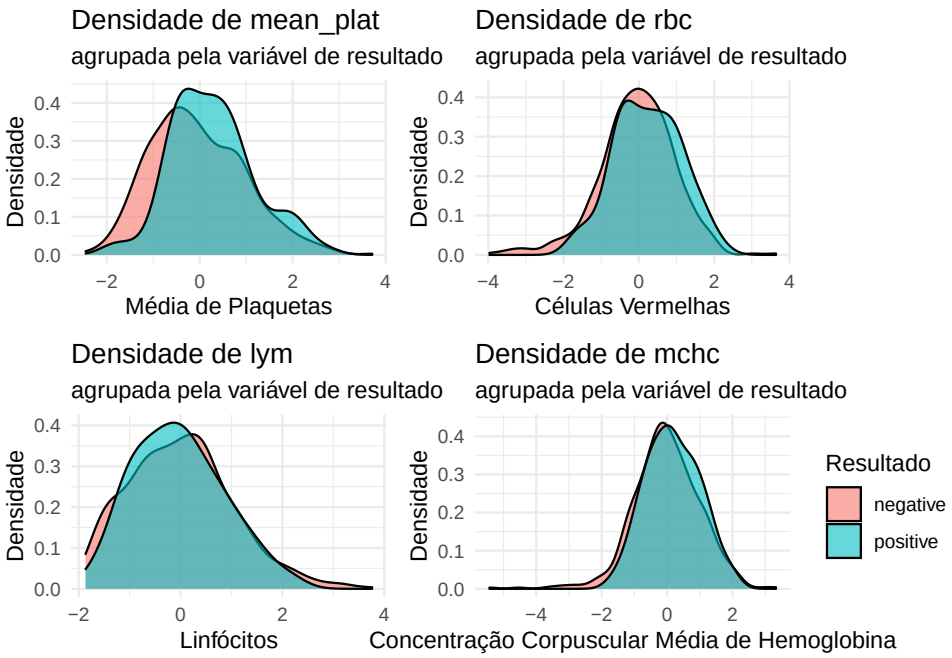

Figure 2: Densidade das variáveis meanplat, rbc, lym e mchc agrupadas por diagnóstico.

```
wbc <- df |>
  ggplot()+
  aes(x=wbc, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Leucócitos', y='Densidade',
       title='Densidade de wbc',
       subtitle = 'agrupada pela variável de resultado')

baso <- df |>
  ggplot()+
  aes(x=baso, fill=result)+
  geom_density(alpha=0.6)+
```

```
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Granulócito Basófilo', y='Densidade',
       title='Densidade de baso',
       subtitle = 'agrupada pela variável de resultado')

mch <- df |>
  ggplot()+
  aes(x=mch, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Média Corpuscular de Hemoglobina', y='Densidade',
       title='Densidade de mch',
       subtitle = 'agrupada pela variável de resultado')

eos <- df |>
  ggplot()+
  aes(x=eos, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'right')+
  labs(x='Granulócito Eosinófilo', y='Densidade',
       title='Densidade de eos',
       fill='Resultado',
       subtitle = 'agrupada pela variável de resultado')

(wbc+baso)/(mch+eos)
```

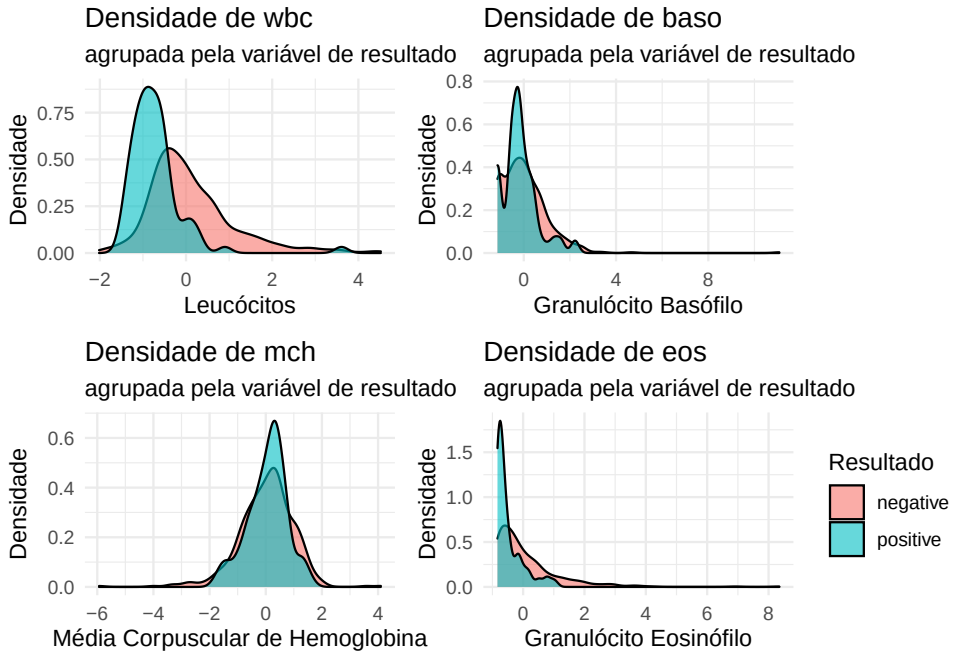

Figure 3: Densidade das variáveis wbc, baso, mch e eos agrupadas por diagnóstico.

```
mcv <- df |>
  ggplot()+
  aes(x=mcv, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Volume Corpuscular Médio', y='Densidade',
       title='Densidade de mcv',
       subtitle = 'agrupada pela variável de resultado')

mono <- df |>
  ggplot()+
  aes(x=mono, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Monócitos', y='Densidade',
       title='Densidade de mono',
       subtitle = 'agrupada pela variável de resultado')

rdw <- df |>
  ggplot()+
  aes(x=rdw, fill=result)+
  geom_density(alpha=0.6)+
  theme_minimal()+
  theme(legend.position = 'none')+
  labs(x='Distribuição da Amplitude de Células Vermelhas', y='Densidade',
       title='Densidade de mch',
       subtitle = 'agrupada pela variável de resultado')

(mcv+mono)/rdw
```

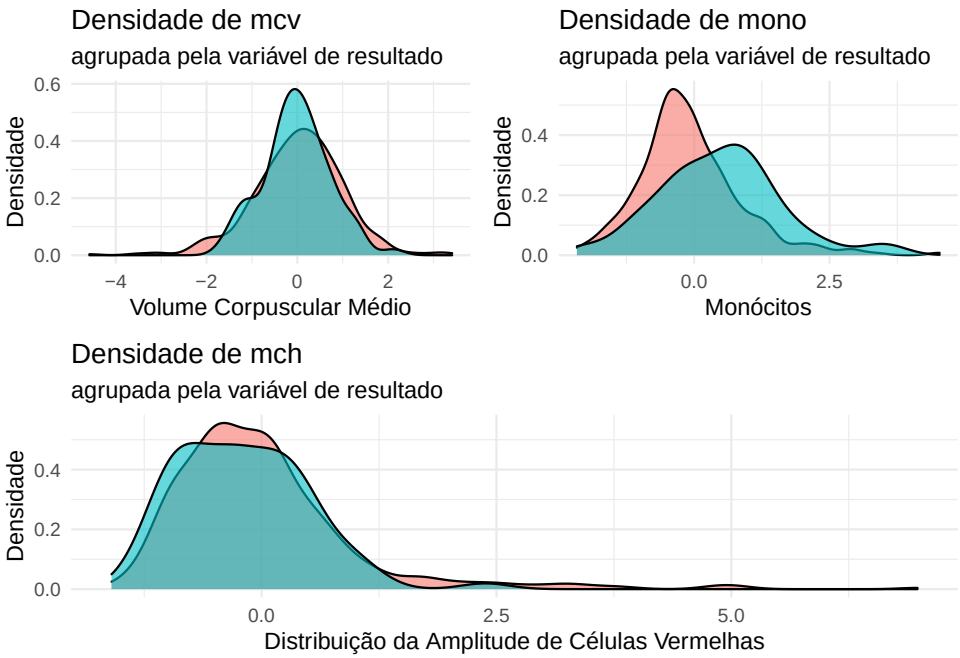

Figure 4: Densidade das variáveis mcv, mono e mcg agrupadas por diagnóstico.

Item B anabe

```
### ITEM B: divisao dos dados

df <- df |>
  mutate(result=as.factor(ifelse(result=='negative',0,1)))

set.seed(57)

split <- initial_split(df, prop=0.6) # proporcao de 0.6 para treino

tre <- training(split)
tes <- testing(split)

x_tre <- model.matrix(result~., tre)
y_tre <- pull(tre[,1])

x_tes <- model.matrix(result~., tes)
y_tes <- pull(tes[,1])
```

Item C

Dado que tratamos de uma situação em que o diagnóstico de falso negativo (isto é, dizer que uma pessoa não tem COVID-19 dado que ela tem) é muito mais grave do que o de falso positivo, é de interesse nosso utilizarmos métricas com isto em mente.

Para isso, utilizaremos a **Sensibilidade** (dos pacientes doentes, quantos foram corretamente identificados) e o **Valor Preditivo Positivo** (dos pacientes classificados como doentes, quantos foram corretamente identificados).

Também apresentaremos a **Acurácia** do modelo (quantas observações foram identificadas corretamente) e a **Especificidade** (dos pacientes não doentes, quantos foram corretamente identificados). No entanto, o foco da avaliação não será nestas duas e sim nas outras duas citadas anteriormente, que identificam da melhor forma se o diagnóstico de COVID-19 positivo está sendo realizado da melhor forma (tendo em mente a maior quantidade de observações com COVID-19 negativo).

Item D larissa

```
### ITEM D

## Verificando balanceamento

n <- nrow(df)

df$result |>
  table() |>
  kable(align = 'cc', col.names = c('result', 'Frequência'),
        caption = 'Verificação de balanceamento do banco de dados') |>
  kable_styling(position="center",
                latex_options="HOLD_position")
```

Table 1: Verificação de balanceamento do banco de dados

result	Frequência
0	517
1	81

```
## Ajuste Lasso

cv_lasso <- cv.glmnet(x_tre, y_tre, alpha=1, family='binomial')
ajuste_lasso <- glmnet(x_tre, y_tre, alpha=1, lambda = cv_lasso$lambda.1se,
                       family='binomial')

## Erro x Lambda Lasso

tibble(
  lambda=cv_lasso$lambda,
  risco=cv_lasso$cvm
) |>
  ggplot()+
  aes(x=lambda, y=risco)+
  geom_line()+
  geom_vline(xintercept = cv_lasso$lambda.1se)+
  geom_vline(xintercept = cv_lasso$lambda.min)+
  annotate(geom = 'text', y=0.75, x=0.023,
           label=paste0('lambda.1se = ', round(ajuste_lasso$lambda,5)))+
```

```
  annotate(geom = 'text', y=0.7, x=0.017,
           label=paste0('lambda.min = ', round(cv_lasso$lambda.min,5)))+
  theme_minimal()+
  labs(title='Risco x Lambda estimado',
       subtitle = 'com lambda.1se e lambda.min')
```

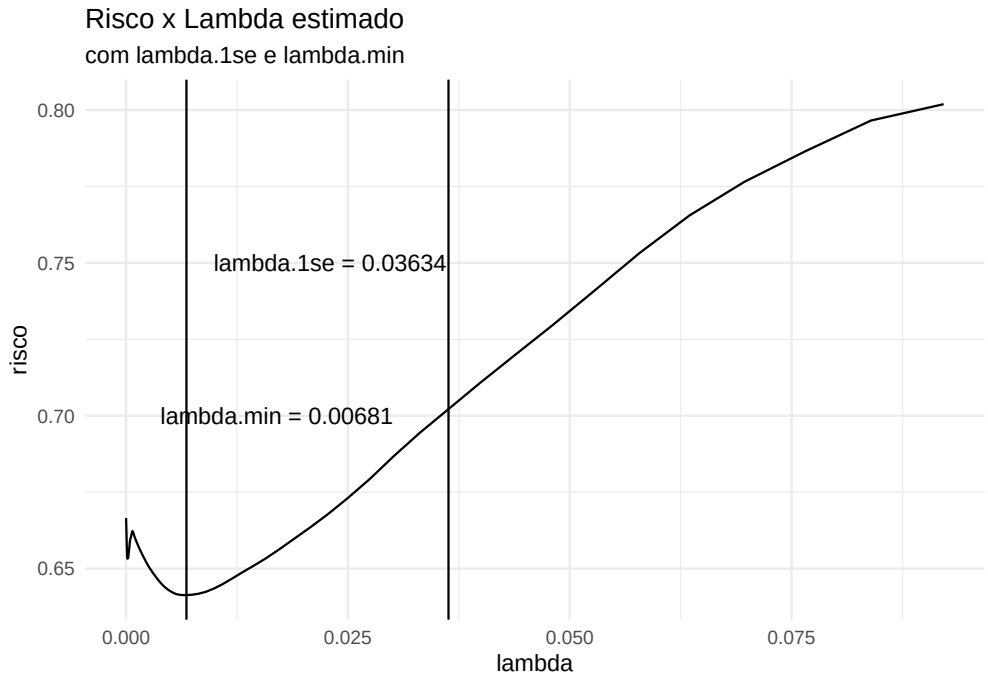

Figure 5: Risco estimado x Lambda para penalização da Regressão Logística via Lasso.

```
## Ajuste KNN

# Realizando calculo do melhor K

ajuste_knn <- train(
  x=x_tre,
  y=y_tre,
  method = 'knn',
  tuneLength = 20
)

# Plotando grafico de K vs Risco

plot(ajuste_knn)
```

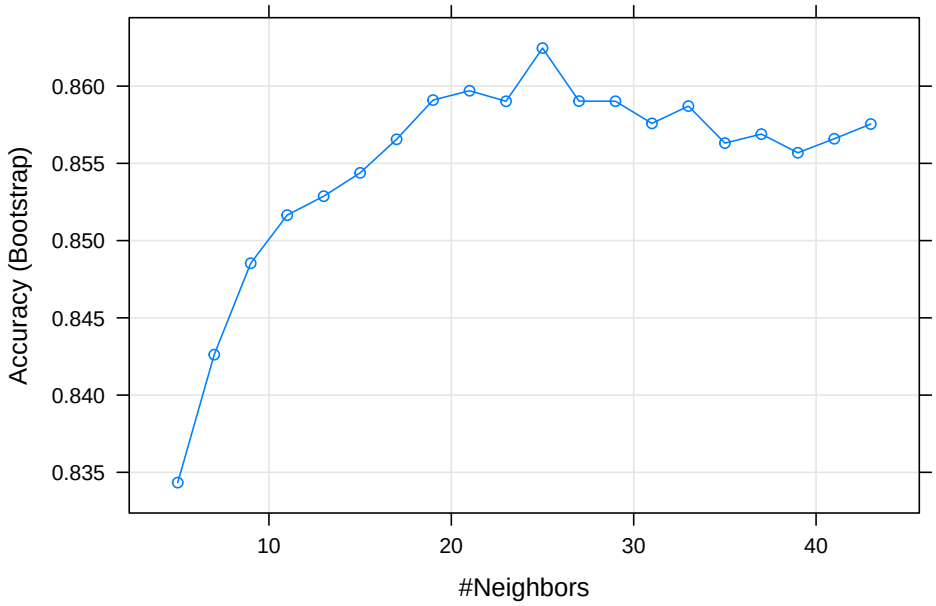

Figure 6: K x Estimativa de Risco do ajuste via KNN.

```
paste0('O melhor K é: ', ajuste_knn$bestTune)
```

```
## [1] "O melhor K é: 25"
```

```
## Arvore de Decisão

# Transformando dados apropriadamente
tre_arv <- data.frame(y_tre, x_tre[,-1])


# Ajustando árvore
ajuste_arv <- rpart(y_tre~., data=tre_arv)

# Podando ajuste
ajuste_arv <- prune(ajuste_arv, cp=0.03)

# Árvore resultante
rpart.plot(ajuste_arv)
```

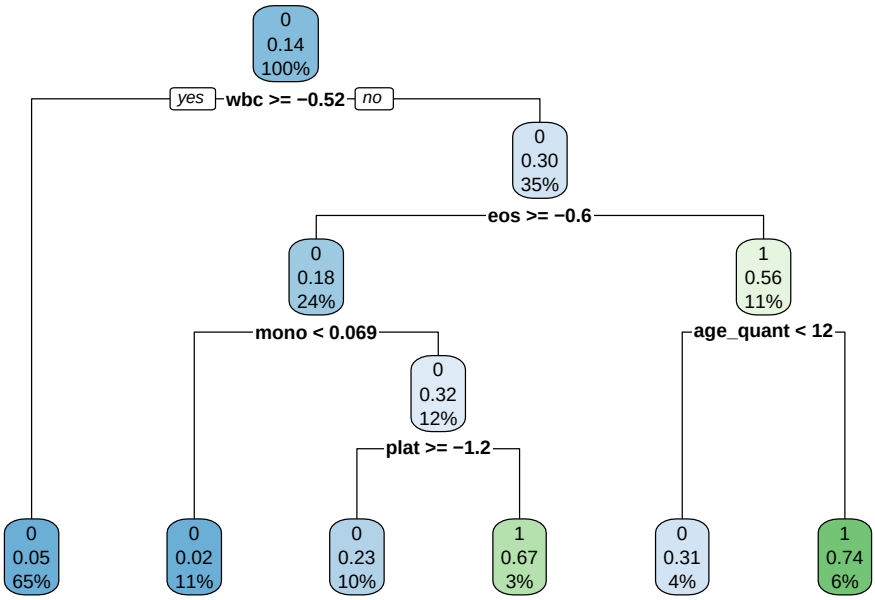

Figure 7: Árvore de Decisão ajustada após processo de poda.

```
## Floresta Aleatória

# Realizando Ajuste

ajuste_flor <- randomForest(x=x_tre, y=y_tre, mtry = round(ncol(df)/3),
                            importance=TRUE)

# Ajuste x Numero de Arvores

plot(ajuste_flor)
```

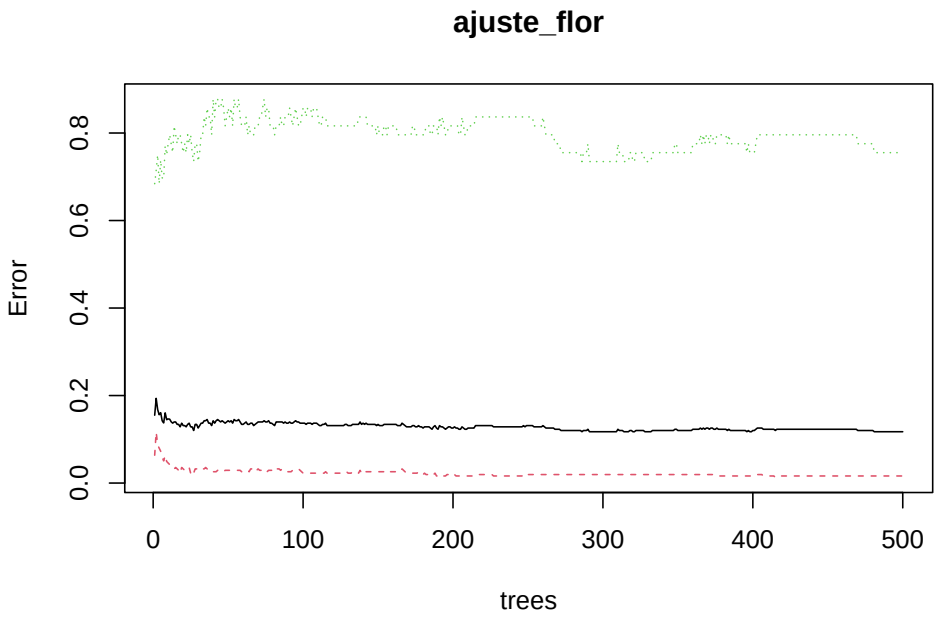

Figure 8: Número de Árvores x Risco dos ajustes via Florestas Aleatórias.

```
## Redes Neurais

# Adequando dados
tre_nn <- tre |>
  mutate(
    across(age_quant:rdw, .fns=scale),
    result=as.factor(ifelse(result==0, 'negativo', 'positivo'))
  )

tes_nn <- tes |>
  mutate(
    across(age_quant:rdw, .fns=scale),
    result=as.factor(ifelse(result==0, 'negativo', 'positivo'))
  )

# Parametros iniciais
nnetGrid <- expand.grid(.size=7,
                        .decay=c(0,.01,.1,.2,.3,.4,.5,1,2))

# Metodo de validação dentro do treino

ctrl <- trainControl(method='repeatedcv',
                     repeats=5,
                     classProbs = TRUE)
# Ajustando

ajuste_nn <- train(result~.,
                   data=tre_nn,
                   method='nnet',
```

```
                  metric='Accuracy',
                  tuneGrid=nnetGrid,
                  trControl=ctrl,
                  maxit=1000)

# Plotando acurácia vs weigth decay

plot(ajuste_nn)
```

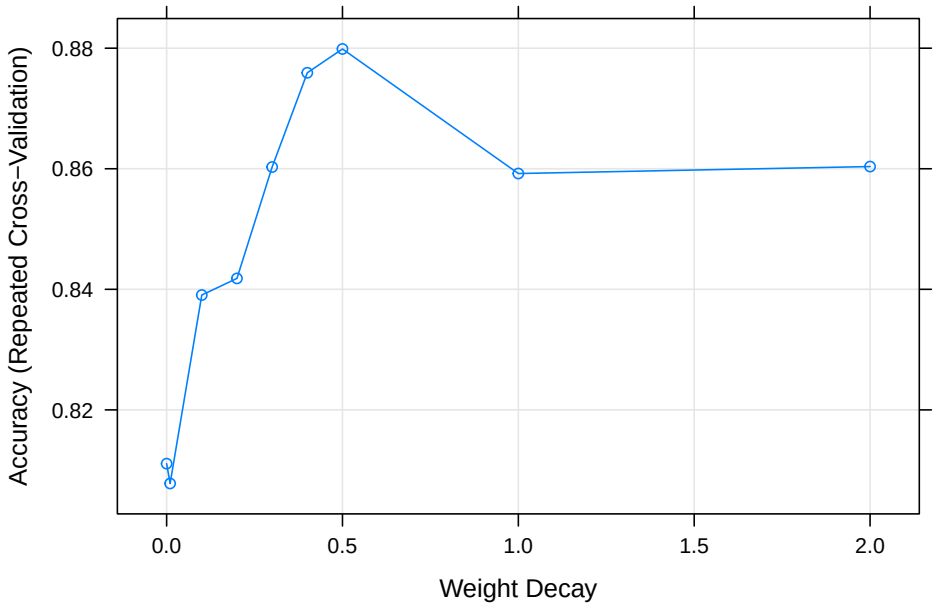

Figure 9: Acurácia x parâmetro Weight Decay dos ajustes via Redes Neurais

```
# Plotando arquitetura resultante da rede

devtools::source_url(paste0('https://gist.githubusercontent.com/fawda123/',
                            '7471137/raw/466c1474d0a505ff0',
                            '44412703516c34f1a4684a5/nnet_plot_update.r'))

plot(ajuste_nn$finalModel)
```

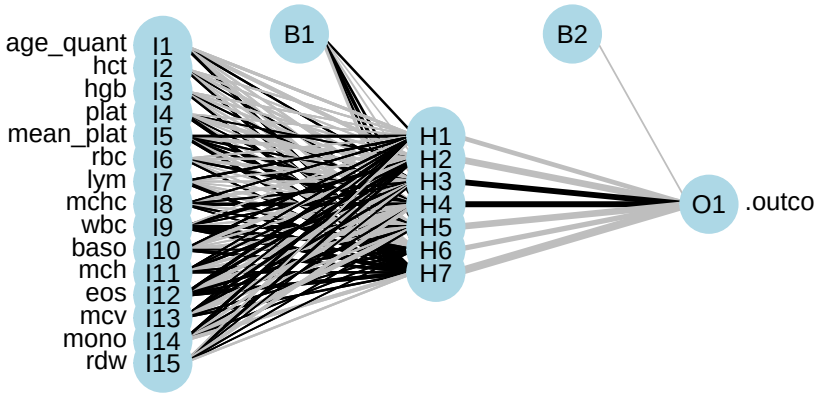

Figure 10: Arquitetura Resultante da melhor Rede Neural ajustada.

```
## XGBoost

set.seed(57)

# Montando dados de treinamento
X <- subset(tre, select = -result)
X.dgCMatrix <- as(data.matrix(X), "dgCMatrix")
y <- ifelse(tre$result == 1, 1, 0)

dtrain <- xgb.DMatrix(data = as.matrix(tre[,-1]),
                      label = (as.numeric(pull(tre[,1]))-1))

dtes <- xgb.DMatrix(data = as.matrix(tes[,-1]),
                      label = (as.numeric(pull(tes[,1]))-1))

# Encontrando melhor nrounds

params <- list(booster = "gbtree",
               objective = "binary:logistic", eta=0.3, gamma=0,
               max_depth=6, min_child_weight=1, subsample=1,
               colsample_bytree=1, verbose=0)

xgbcv <- xgb.cv(params = params, data = dtrain, nrounds = 100,
                          nfold = 5, showsd = T, stratified = T,
                          print_every_n = 10, early_stopping_rounds = 20,
                          maximize = F, eval_metric='error')

xgbcv$evaluation_log |>
  tibble() |>
```

```
  ggplot()+
  aes(x=iter, y=test_error_mean)+
  geom_line()+
  annotate('text', x=20, y=0.16,
           label=paste0('melhor iteracao: ', xgbcv$best_iteration))+
  theme_minimal()+
  labs(x='Iteração', y='Erro Médio no Treino',
       title = 'Iteração x Erro Médio no Treino',
       subtitle = 'no ajuste via XGBoost para nº de iterações')
```

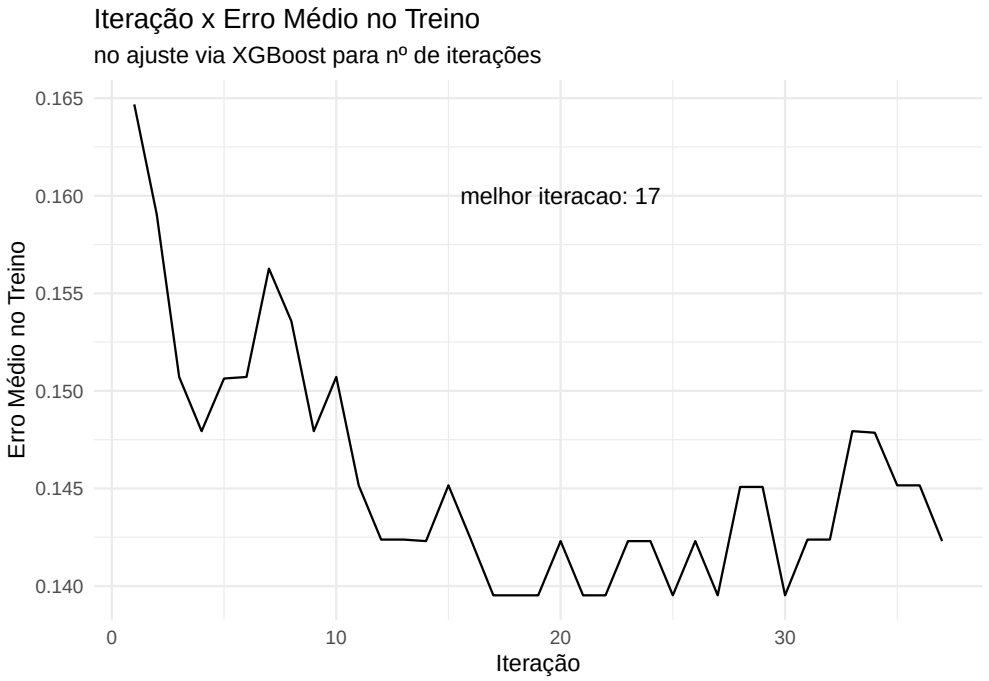

Figure 11: Melhor iteração x Erro médio dos ajustes via XGBoost.

```
# Encontrando melhor eta

eta <- seq(0,20,0.1)
conv_eta <-  data.frame(eta, error=numeric(length(eta)))
for(i in 1:length(eta)){
  set.seed(57)
  params=list(booster='gbtree', eta = eta[i], colsample_bytree=1,
              subsample = 1, max_depth = 6, verbose=0,
              objective='binary:logistic', gamma=0, eval.metric='error')
  xgb=xgboost(data=dtrain, nrounds = 17, params = params)
  conv_eta[i,2] = xgb$evaluation_log$train_error[17]
}
```

```
conv_eta |>
  tibble() |>
  ggplot()+
  aes(x=eta,y=error)+
```

```
  geom_line()+
  annotate('text', x=10, y=0.4, label=paste0('melhor eta: ',
                                             conv_eta[conv_eta$error==min(conv_eta$error),]$eta[1]))+
  theme_minimal()+
  labs(x='Eta', y='Erro Médio no Teste',
       title='Iteração x Erro Médio no Treino',
       subtitle='no ajuste via XGBoost para eta')
```

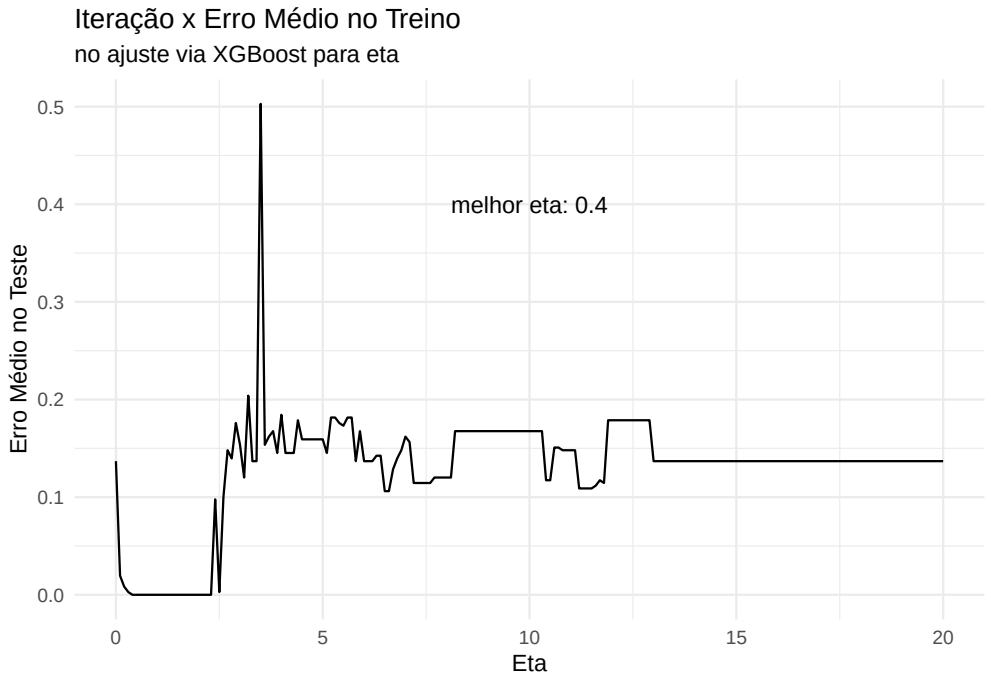

Figure 12: Apresentação do melhor eta encontrado via validação cruzada no treino para ajuste via XGBoost.

```
# Ajustando modelo final

set.seed(57)

params <- list(booster='gbtree', eta = 0.4, colsample_bytree=1,
               subsample = 1, max_depth = 6,
               objective='binary:logistic', gamma=0, eval.metric='error')

ajuste_xgb <- xgboost(data=X.dgCMatrix, nrounds = 17, params = params,
                      label = y, save_period = NULL)
```

Item E

Para construírmos a avaliação dos modelos ajustados via Curva ROC, fizemos:

1. O cálculo das probabilidades de COVID-19 positiva preditas para cada uma das observações do conjunto de teste;
2. Com o auxílio do pacote `{ROCR}`, computamos os valores da Curva ROC de cada um dos métodos;

3. Juntamos todas as curvas dos seis ajustes num gráfico só;
4. Calculamos, via distância euclidiana, qual ponto em cada uma das curvas está mais próximo do modelo perfeito (quanto Sensibilidade=1-Especificidade=1). O corte de probabilidade a ser utilizado para estes casos é a projeção destes pontos no Eixo X.

Os resultados obtidos, bem como o código utilizado, seguem abaixo.

```
### ITEM E

## Calculando Preditos

pred_lasso <- predict(ajuste_lasso, x_tes, type='response') |> c()

pred_knn <- predict(ajuste_knn, x_tes, type='prob')[,2] |> c()

pred_arv <- predict(ajuste_arv, tes, type='prob')[,2] |>
  c()

pred_flor <- predict(ajuste_flor, x_tes, type='prob')[,2] |>
  c()

pred_nn <- predict(ajuste_nn, tes_nn, type='prob')[,2] |>
  c()

pred_xgb <- predict(ajuste_xgb, dtes, type='prob')

pred_list <- list(lasso=pred_lasso, knn=pred_knn,
                  arv=pred_arv, flor=pred_flor,
                  nn=pred_nn, xgb=pred_xgb)

## Calculando

data_result <- list(
  x=NULL,
  y=NULL,
  method=NULL
)

for (ii in 1:length(pred_list)) {
  pred <- prediction(pred_list[[ii]], tes$result)
  perf <- performance(pred,"sens","spec")

  x_val <- unlist(perf@x.values)
  y_val <- unlist(perf@y.values)

  data_result$x <- append(data_result$x, x_val)
  data_result$y <- append(data_result$y, y_val)
  data_result$method <- append(data_result$method,
                               rep(names(pred_list)[ii],
                                length(x_val)))
}

data_result <- data_result |>
  data.frame() |> tibble()
```

```r
# Calculando pontos mais prox. do modelo perfeito

euc <- function(a, b) sqrt(sum((a - b)^2))

distancias <- numeric(nrow(data_result))

for (ii in 1:nrow(data_result)) {
  distancias[ii] <- euc(c(1,1), data_result[ii,-3])
}

data_result <- data_result |>
  mutate(
    dist=distancias
  )

mais_prox <- data_result |>
  group_by(method) |>
  slice_min(order_by = dist)

data_result |>
  ggplot()+
  aes(x=1-x,y=y, color=method)+
  geom_line()+
  geom_point(data=mais_prox,
             aes(x=1-x,y=y, color=method))+
  geom_vline(xintercept = 1-mais_prox$x,
             alpha=0.25, linetype='dashed')+
  theme_minimal()+
  labs(x='1-Especificidade',
       y='Sensibilidade',
       title='Curvas ROC dos modelos ajustados',
       color='Métodos')+
  scale_x_continuous(breaks = c(0,0.25,0.5,0.75,1,round(1-mais_prox$x,2)))+
  theme(legend.position = 'bottom',
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1, size=8))
```

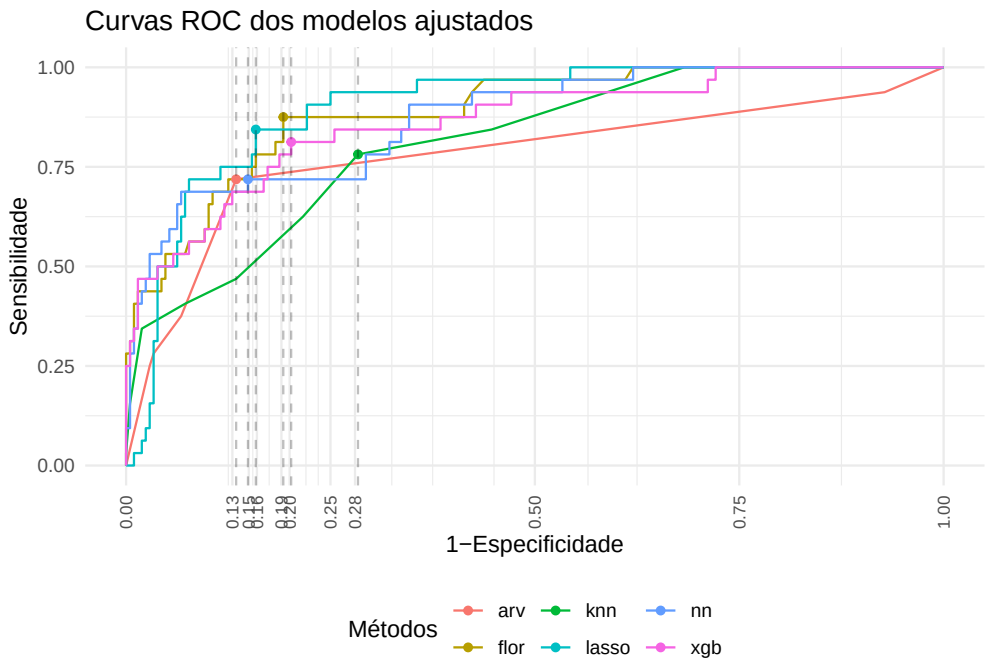

Figure 13: Curva ROC dos modelos ajustados com projeção de pontos mais próximos do modelo perfeito.

No geral, temos que o ajuste via Regressão Logística com penalização Lasso demonstra melhor desempenho, já que sua curva (em azul claro) permanece mais próxima do ideal por mais tempo do que as outras. Depois deste ajuste, os ajustes via Florestas Aleatórias (em bege/marrom) e Redes Neurais (em azul escuro) parecem apresentar desempenho interessante.

Além disso, com as projeções, temos que:

- O corte de probabilidade no classificador de bayes para o ajuste via Regressão Logística com penalização lasso será de 0.16;
- O corte de probabilidade no classificador de bayes para o ajuste via KNN será de 0.28;
- O corte de probabilidade no classificador de bayes para o ajuste via Árvore de Classificação será de 0.13;
- O corte de probabilidade no classificador de bayes para o ajuste via Florestas Aleatórias será de 0.19;
- O corte de probabilidade no classificador de bayes para o ajuste via Redes Neurais será de 0.15;
- O corte de probabilidade no classificador de bayes para o ajuste via XGBoost será de 0.2.

Item F

Como dito no Item E, determinamos os cortes de probabilidade com base nos pontos mais próximos (através do cálculo da distância euclidiana) do modelo ideal. Assim, iremos classificar as observações, utilizando um corte p, utilizando o Classificador de Bayes:

$$g(x) = 1 \Leftrightarrow \mathbb{P}(Y = 1|x) \geq p$$

Ou seja, classificamos a observação como COVID-19 positiva se a probabilidade de que Y seja igual a 1 for maior do que o corte definido para o ajuste. Vale lembrarmos do que foi respondido no Item C: julgamos que

as métricas mais importantes são **Sensibilidade** e o **Valor Preditivo Positivo**. Também consideraremos, ainda que não com a mesma importância, a **Acurácia** e a **Especificidade**.

Os resultados foram obtidos através da função `confusionMatrix()` do pacote `{caret}`, e encontram-se dispostos à seguir:

```
### ITEM F

# Definindo cortes com base no Item E

cortes <- c(0.16, 0.28, 0.13, 0.19,
            0.15, 0.2)

class_pred <- list(
  lasso=NULL,
  knn=NULL,
  arv=NULL,
  flor=NULL,
  nn=NULL,
  xgb=NULL
)

for (ii in 1:length(class_pred)) {

  # Classificando com base nos cortes
  class_pred[[ii]] <- ifelse(pred_list[[ii]]>=cortes[ii],1,0)

}

metric_result <- list(
  lasso=NULL,
  knn=NULL,
  arv=NULL,
  flor=NULL,
  nn=NULL,
  xgb=NULL
)

tabl <- NULL

for (ii in 1:length(metric_result)) {

  tabl <- confusionMatrix(data=as.factor(class_pred[[ii]]),
                          reference = tes$result)

  metric_result[[ii]] <- c(tabl$byClass[1], tabl$byClass[3],
                           tabl$byClass[2], tabl$overall[1])

}

# Apresentando resultados

data.frame(metric_result) |>
  kable('latex', align='ccccccc',
        caption = 'Métricas dos Modelos Ajustados após definição dos cortes') |>
```

```
  kable_styling(position="center",
                latex_options="HOLD_position")
```

Table 2: Métricas dos Modelos Ajustados após definição dos cortes

	lasso	knn	arv	flor	nn	xgb
Sensitivity	0.7980769	0.9278846	0.8653846	0.8461538	0.7548077	0.9182692
Pos Pred Value	0.9707602	0.9103774	0.9523810	0.9513514	0.9457831	0.9317073
Specificity	0.8437500	0.4062500	0.7187500	0.7187500	0.7187500	0.5625000
Accuracy	0.8041667	0.8583333	0.8458333	0.8291667	0.7500000	0.8708333

O modelo com maior Sensibilidade foi o ajuste via KNN, em que 92.78846% dos pacientes doentes foram identificados corretamente (atingindo um patamar de detecção de COVID-19 muito bom). Seu Valor Preditivo Positivo apresentou o pior valor (91.03774% dos pacientes classificados como doentes foram identificados corretamente). No entanto, sua especificidade deixou a desejar, dectando corretamente apenas 40.635% dos pacientes não doentes.

Outros ajustes apresentam Sensibilidade pior, mas outras métricas maiores (como é o caso da Regressão Logística com Lasso, Árvore de Classificação, entre os outros ajustes). O que mais chama a atenção é o ajuste via XGBoost:

- Sua Sensibilidade possui desempenho similar ao do KNN, identificando corretamente 91.82692% dos pacientes doentes;
- Todas as outras métricas são maiores do que o ajuste via KNN.

Ainda que a acurácia do modelo ajustado via XGBoost é baixa (identifica corretamente apenas 56.25% dos casos), seria muito mais grave termos uma Sensibilidade ou VPP baixos em detrimento de uma Acurácia alta do que o contrário. Assim, para uma situação prática em que o modelo ajustado influencia diretamente em vidas humanas, é de interesse público que escolhamos o ajuste via XGBoost como aquele que deve ser utilizado para o diagnóstico da COVID-19.

Item G

```
### ITEM G

# Definindo variaveis que iremos utilizar

vars <- c('age_quant', 'plat', 'wbc')

list_graph <- list(
  age_quant=NULL, plat=NULL, wbc=NULL
)

# Função para plotagem do pdp do lasso

pred.fun <- function(object, newdata) {
  mean(predict(object, newx = newdata,
               s = cv_lasso$lambda.1se)[, 1L, drop = TRUE])
}
```

```r
# Encontrando graficos via for loop

for (ii in 1:length(list_graph)) {

  lasso <- partial(ajuste_lasso, pred.var = vars[ii],
                   pred.fun = pred.fun, train = x_tre) |>
    autoplot()+
    theme_minimal()+
    labs(title=paste0('lasso - ', vars[ii]))

  knn <- partial(ajuste_knn, pred.var=vars[ii]) |>
    autoplot()+
    theme_minimal()+
    labs(title=paste0('knn - ', vars[ii]))

  arv <- partial(ajuste_arv, pred.var=vars[ii]) |>
    autoplot()+
    theme_minimal()+
    labs(title=paste0('arv - ', vars[ii]))

  flor <-  partial(ajuste_flor, pred.var=vars[ii]) |>
    autoplot()+
    theme_minimal()+
    labs(title=paste0('flor - ', vars[ii]))

  nn <- partial(ajuste_nn$finalModel, pred.var=vars[ii],
                train=tre) |>
    autoplot()+
    theme_minimal()+
    labs(title=paste0('nn - ', vars[ii]))

  xgb <- partial(ajuste_xgb, pred.var = vars[ii], prob=TRUE,
                 train=X.dgCMatrix) |>
    autoplot()+
    theme_minimal()+
    labs(title=paste0('xgb - ', vars[ii]))

  list_graph[[ii]] <- (lasso+knn+arv)/(flor+nn+xgb)

}
```

```r
# Apresentando pdp da var. age_quant

list_graph[[1]]
```

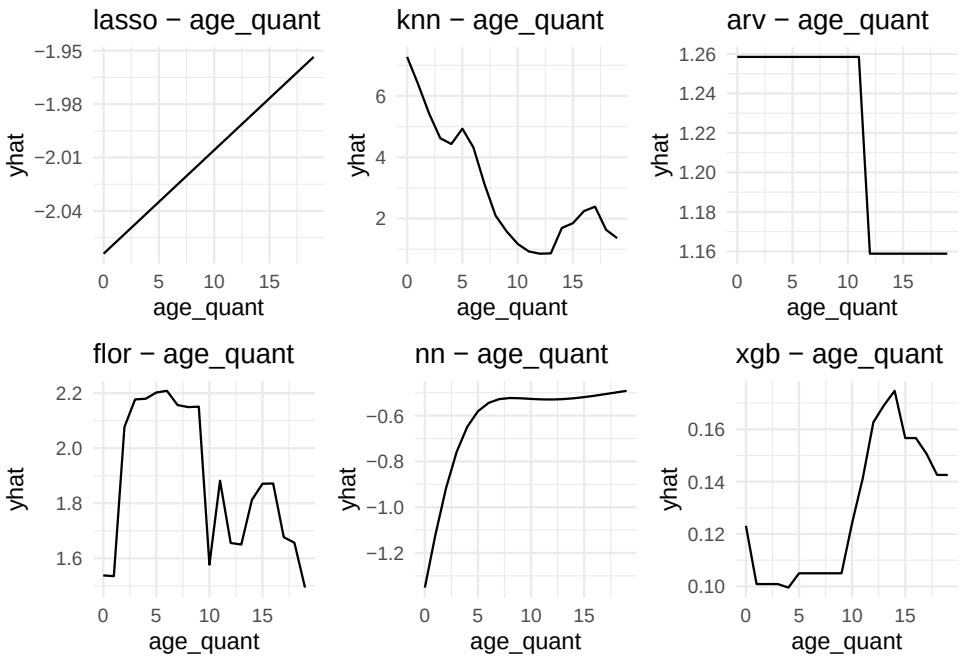

Figure 14: PDPs para os seis modelos ajustados da variável agequant

```
# Apresentando pdp da var. plat

list_graph[[2]]
```

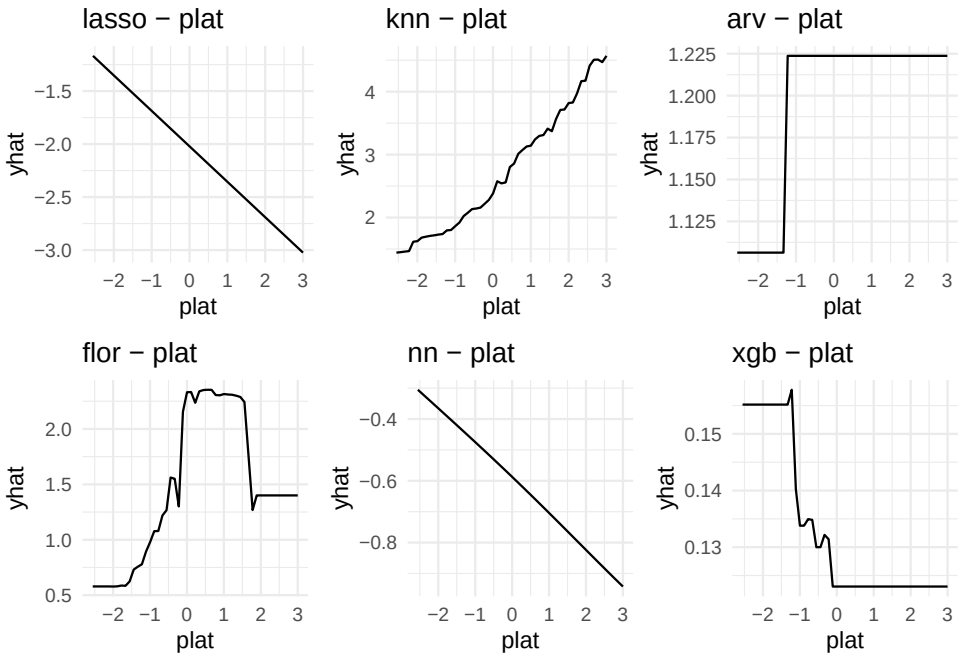

Figure 15: PDPs para os seis modelos ajustados da variável plat

```
# Apresentando pdp da var. wbc

list_graph[[3]]
```

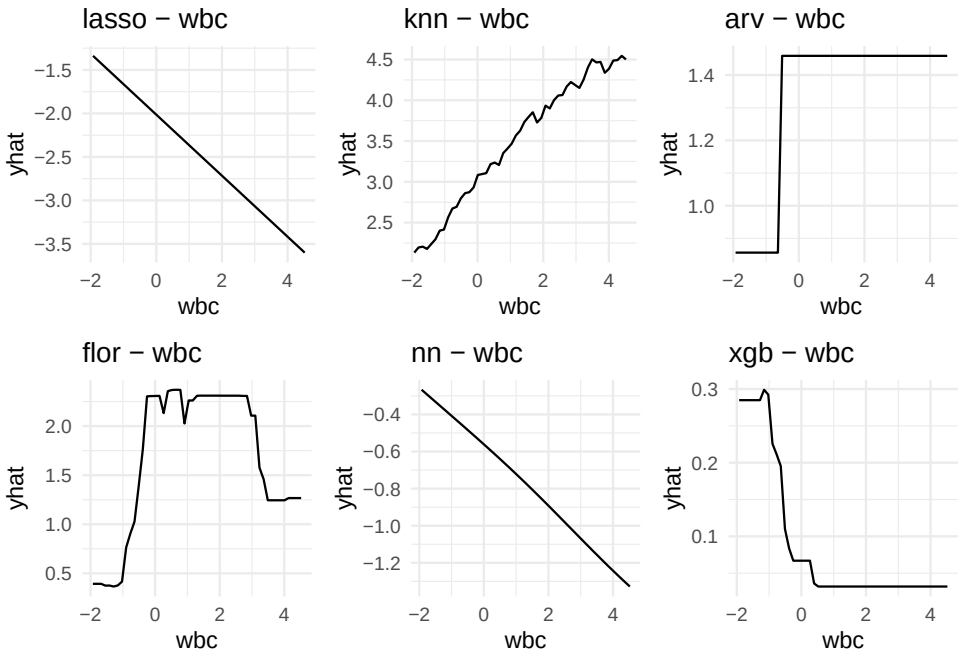

Figure 16: PDPs para os seis modelos ajustados da variável wbc