# K-fold Cross-Validation: Aplicação no modelo KNN

Baseado em: “*An introduction to statistical learning with applications in R*”

# Objetivo:

Utilizar o método de *K-fold CV* estudado anteriormente junto da análise de variáveis para determinar a melhor quantidade de vizinhos no modelo de previsão *KNN*.

# Considerações:

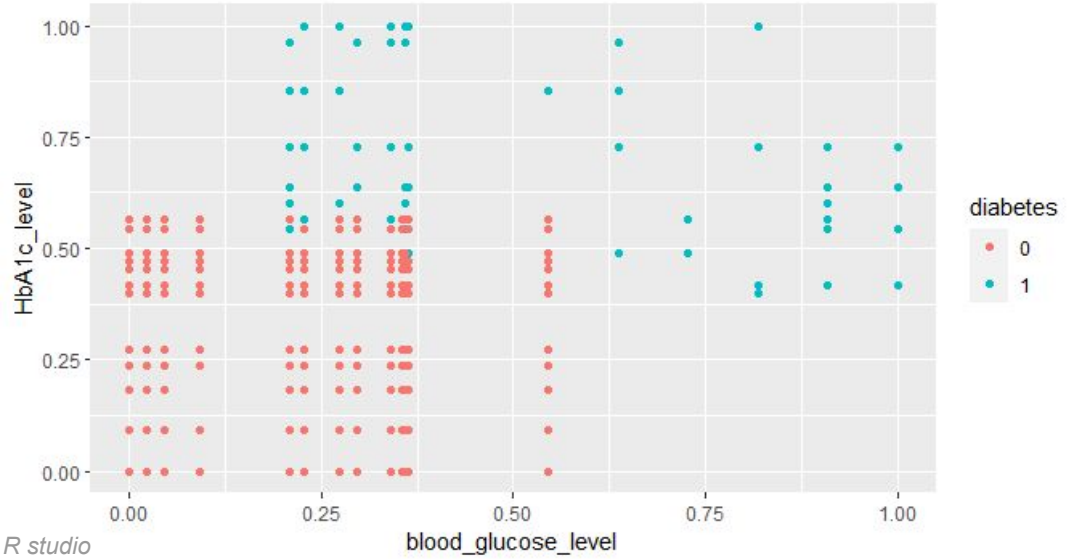
- Dataset usado: Diabetes prediction e Abalone
- Todas iterações de *K-fold CV* utilizam *K*=10
- Bibliotecas do R Studio usadas:

```
library(tidyverse)
library(class)
library(MASS)
library(reshape2)
library(reshape)
library(corrplot)
library(rpart)
library(rpart.plot)
```

*R studio*

# Revisão: Modelo de Previsão *K-Nearest-Neighbours*

Para cada observação que deve ser classificada, comparamos sua “distância” das observações do conjunto de treinamento baseado em variáveis específicas, a classificação da observação será baseada na classificação das *K* observações mais próximas.




*R studio*

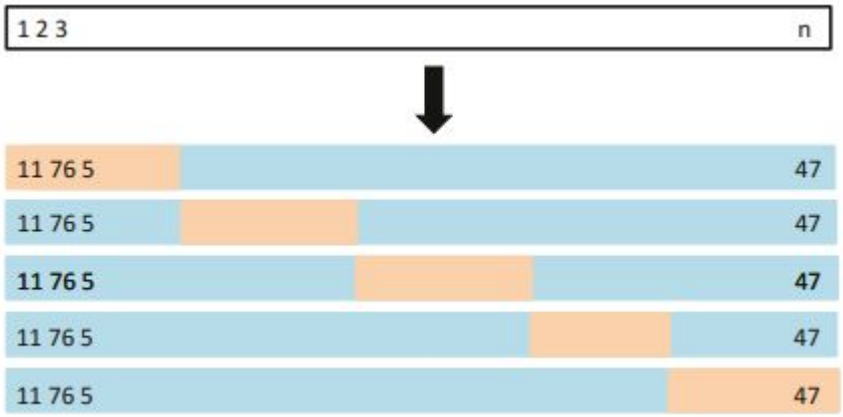
# Revisão: Modelo de Previsão *K-Nearest-Neighbours*

```r
nor <-function(x) {(x -min(x))/(max(x)-min(x))}
distancia <- function(x,y) {return(sqrt(sum((x-y)**2)))}

dados0 <- data.frame(lapply(dados1[,c(2,6,7,8)], nor),diabetes)

treino <- dados0[1:1000,]
teste <- dados0[1001:1100,]

previsao <- c()

for (k in 1:nrow(teste)) {
  distancias <- c()
  for (j in 1:nrow(treino)) {
    distancias[j] <- distancia(teste[k,1:4],treino[j,1:4])
  }
  previsao[k] <- treino$diabetes[order(distancias)[1]]
}

previsao <- previsao-1
mean(previsao == teste$diabetes)
```

*R studio*

# Revisão: Método *K-fold Cross-Validation*

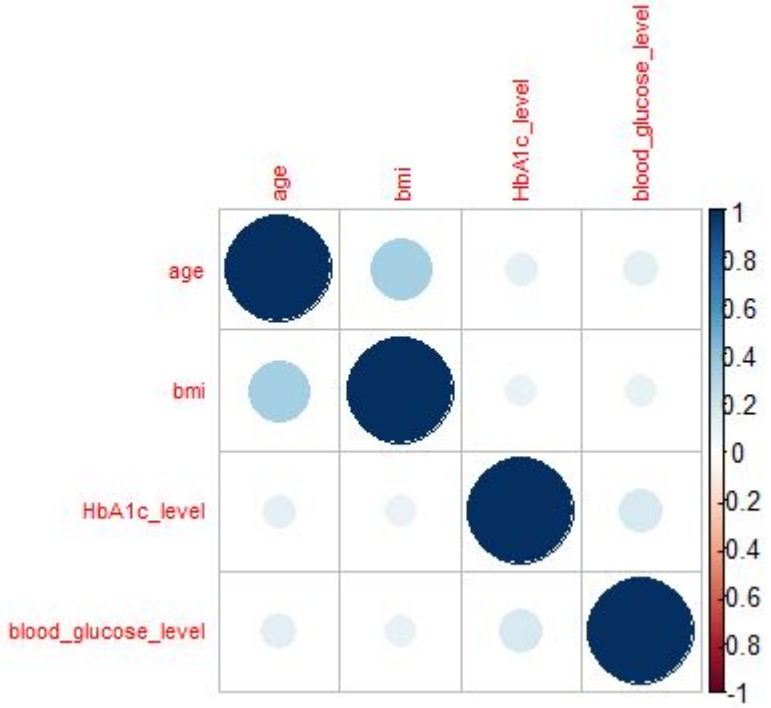
Consiste em dividir o conjunto de dados em *K* subconjuntos com observações retiradas aleatoriamente do conjunto principal. Para cada subconjunto o modelo é treinado nas demais observações e a previsão é realizada no subconjunto, a média das *K* taxas de acerto é a taxa de acerto total do modelo.




*ISRLv2 - sessão 5.1.3*

# Preparação: variáveis relevantes

*KNN* é um modelo que funciona melhor para variáveis numéricas, portanto utilizaremos somente 4 das 8 variáveis do dataset, dito isso é importante verificar se tanto as variáveis que estamos descartando quanto as que estamos incluindo são relevantes para o modelo.




*R studio*

# Preparação: variáveis relevantes

Para variáveis classificatórias ((fatores)), podemos realizar o cálculo do gini de sua partição no dataset para determinar sua relevância. A fórmula para calcular o gini é:

G1=1-(S1/V1)^2-(N1/V1)^2 G1=gini da partição V=1
G2=1-(S0/V0)^2-(N0/V0)^2 G2=gini da partição V=0
GT=G1*(V1/T)+G2*(V0/T) GT=gini total da variável

S1=quantidade de observações classificadas como “sim” e com a variável V=1
N1=quantidade de observações classificadas como “não” e com a variável V=1
S0=quantidade de observações classificadas como “sim” e com a variável V=0
N0=quantidade de observações classificadas como “não” e com a variável V=0
V1=quantidade de observações com variável V=1
V0=quantidade de observações com variável V=0
T=quantidade total de observações

# Preparação: variáveis relevantes

Addendum: quanto menor o gini, melhor é a variável

```
> giniT
[1] 0.1494627
```

*R studio*

```
diabeticos <- dados1[dados1$diabetes==1,]
Ndiabeticos <- dados1[!dados1$diabetes==1,]

#gini da partição hypertension=1

gini1 <- (1-(sum(diabeticos$hypertension==1)/sum(dados1$hypertension==1))**2
        -(sum(Ndiabeticos$hypertension==1)/sum(dados1$hypertension==1))**2)

#gini da partição hypertension=0

gini2 <- (1-(sum(diabeticos$hypertension==0)/sum(dados1$hypertension==0))**2
         -(sum(Ndiabeticos$hypertension==0)/sum(dados1$hypertension==0))**2)

#gini total da variável

giniT <- gini1*sum(dados1$hypertension==1)/nrow(dados1) +
  gini2*sum(dados1$hypertension==0)/nrow(dados1)
```

# Preparação: variáveis relevantes

Heart Disease:

```
> giniT
[1] 0.1509628
```

Gender:

```
> giniT
[1] 0.1553272
```

Smoking History:

```
> giniT
[1] 0.1525073
```

Hypertension:

```
> giniT
[1] 0.1494627
```

Age:

```
0.14253961
```

BMI:

```
0.14524750
```

HbA1c Lvl*:

```
0.07482132
```

Blood Glucose Lvl:

```
0.11439837
```

```
> gini(dados1[1:1000,c(2,6,7,8,9)])
[1] 0.14253961 0.14524750 0.07482132 0.11439837
```

*R studio*

# Preparação: variáveis relevantes

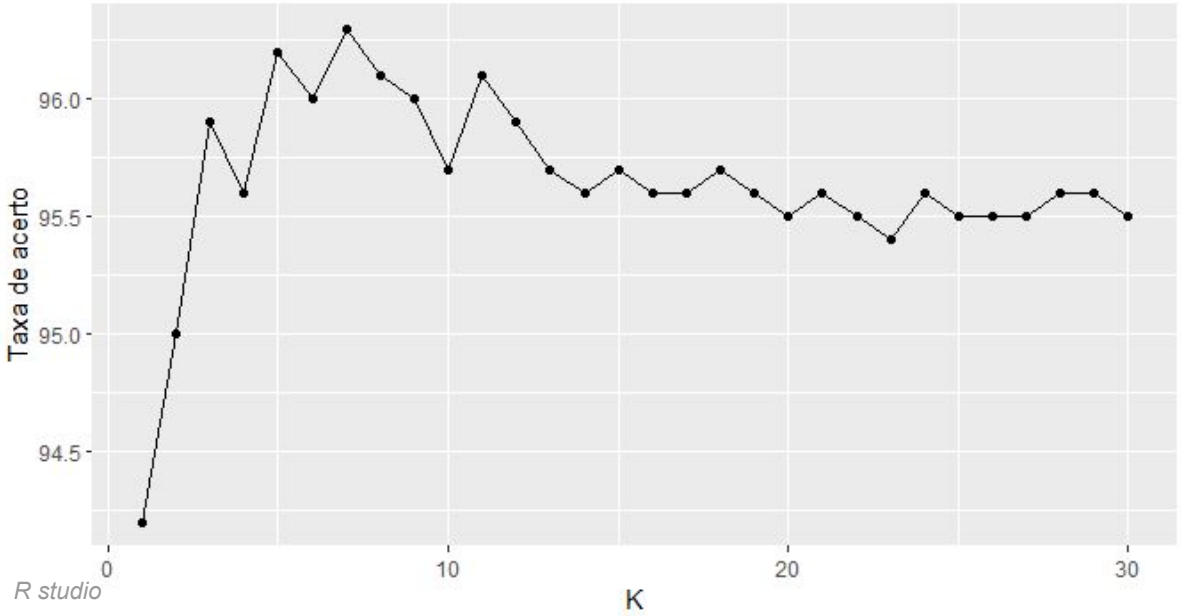
```
gini(dados1[1:1000,c(2,6,7,8,9)])
```

*R studio*

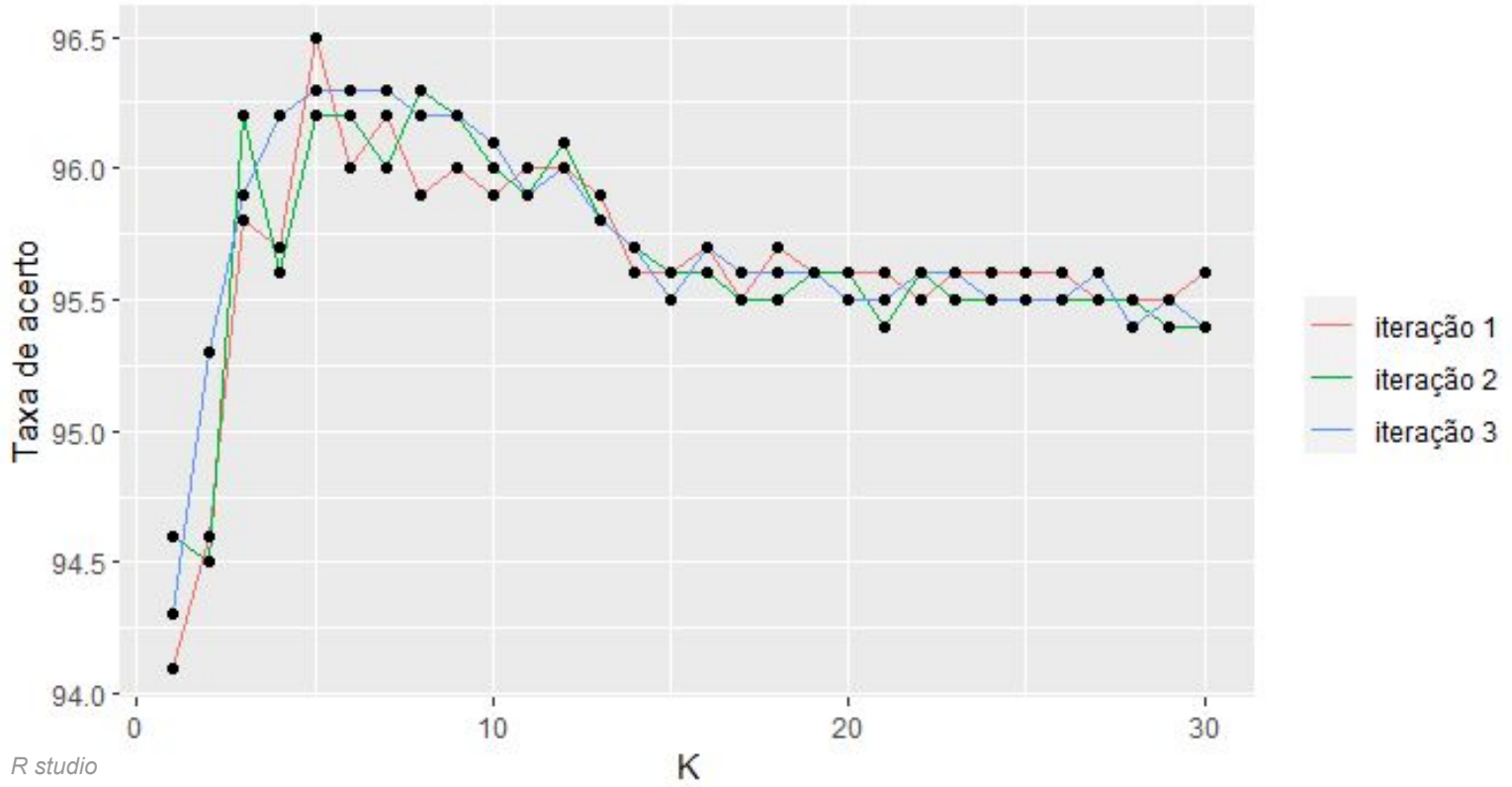
```
gini <- function(treino){
  resuls<-c()
  resulsGinis<-c()
  for(j in 1:4){
    ginis <- c()
    vet <- sort(treino[,j])

    for(i in 1:(nrow(treino)-1)){
      med <- (vet[i]+vet[i+1])/2
      a <-sum(treino[treino[,j]<med,]$diabetes==1)
      b <-sum(treino[treino[,j]<med,]$diabetes==0)
      c <- a+b
      d <-sum(treino[treino[,j]>=med,]$diabetes==1)
      e <-sum(treino[treino[,j]>=med,]$diabetes==0)
      f <- d+e
      g <- c+f

      if(c!=0){gi1 <- c/g * (1 - (a/c)**2 - (b/c)**2)
      }else{gi1 <- 0}

      if(f!=0){gi2 <- f/g * (1 - (d/f)**2 - (e/f)**2)
      }else{gi2 <- 0}

      ginis[i] <- gi1 + gi2
    }
    resulsGinis[j] <- min(ginis)
    resuls[j] <- vet[which(min(ginis) == ginis)]
  }
  return(resulsGinis)}
```

# Experimento: *K-fold CV* aplicado em *KNN*

Realizamos 30 iterações de *K-fold CV* para o modelo de previsão *KNN*, onde cada iteração tem um número *K* de vizinhos diferente, de 1 a 30.



# Experimento: *K-fold CV* aplicado em *KNN*




*R studio*

# Experimento: *K-fold CV* aplicado em *KNN*

```
vet2<-c()
diabetes <- dados1[,9]
dados0 <- data.frame(
  lapply(dados1[,c(2,6,7,8)],nor))
dados <- dados0[c(1:1000),]
dadosZ <- dados1[c(1:1000),]
```

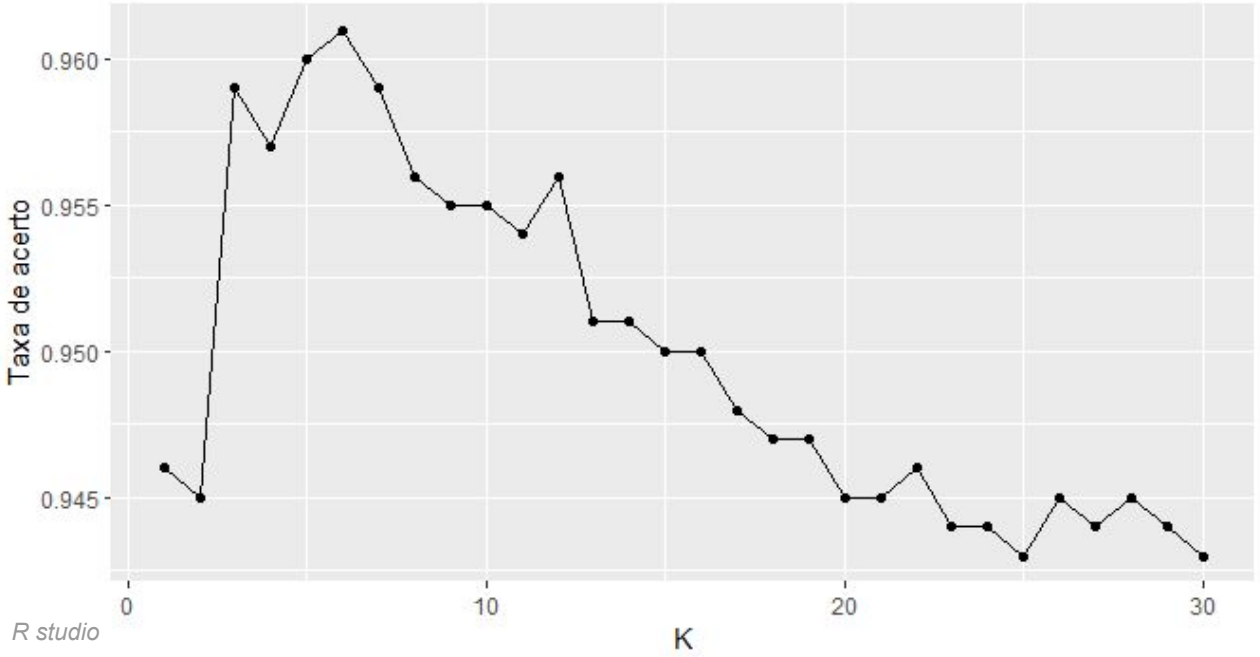
```
> max(vet2)
[1] 96.3
> which(vet2==(max(vet2)))
[1] 7
```

```
for(j in 1:30){

  ran<-sample(1:nrow(dados))

  dados <- dados[ran,]
  dadosZ <- dadosZ[ran,]

  k <- 10
  ct1 <- 1
  vet <- c()

  for(i in 1:k){
    treino <- dados[-(ct1:(ct1+(nrow(dados)/k))-1),]
    teste <- dados[ct1:(ct1+(nrow(dados)/k)-1),]
    treinoC <- dadosZ[-(ct1:(ct1+(nrow(dados)/k))-1),9]
    testeC <- dadosZ[ct1:(ct1+(nrow(dados)/k)-1),9]
    ct1 <- ct1 + nrow(dados)/k
    pr <- knn(treino,teste,cl=treinoC,k=j)
    tab <- table(pr,testeC)
    vet[i] <- accuracy(tab)
    remove(treino)
    remove(teste)
  }

  vet2[j] <- mean(vet)
}
```

*R studio*

# Experimento: *K-fold CV* aplicado em *KNN*

Por fim testamos o modelo em um novo subconjunto que não foi usado nos testes anteriores.




*R studio*

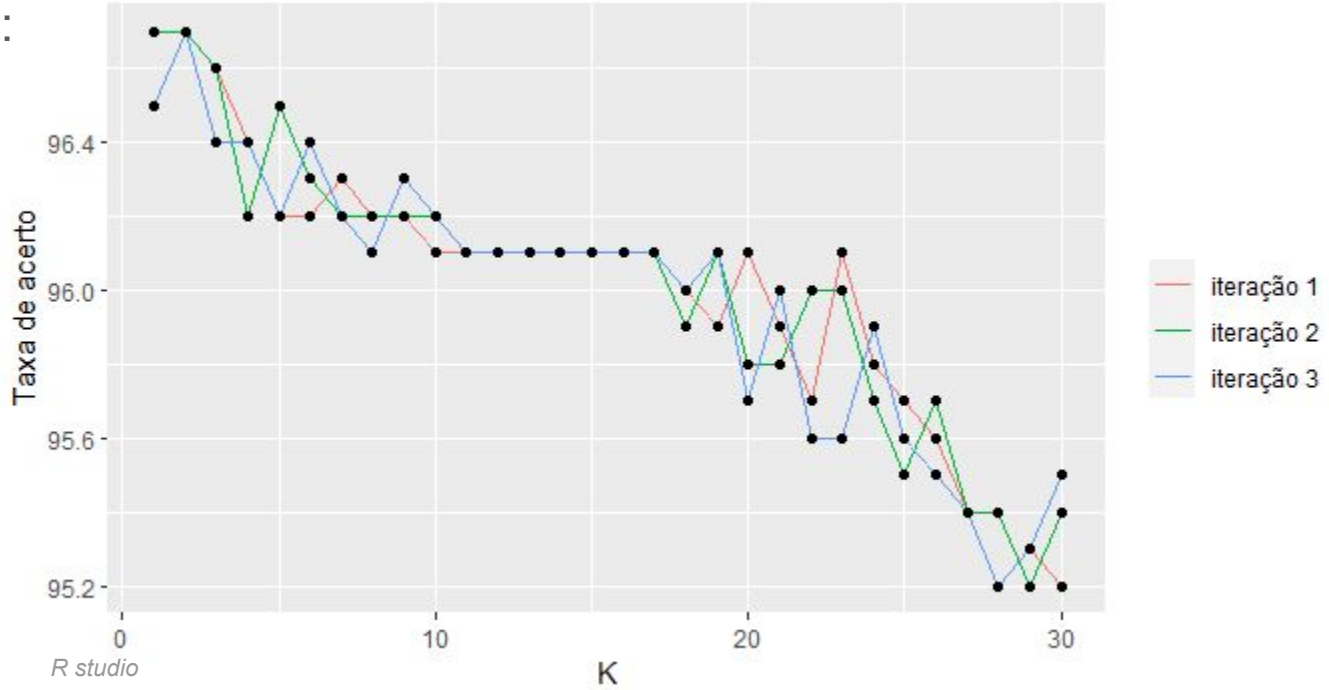
# Experimento: *K-fold CV* aplicado em *KNN*

```r
dados0 <- data.frame(lapply(dados1[,c(2,6,7,8)], nor))

vet <-c()

treino <- dados0[1:1000,]
teste <- dados0[1001:2000,]
treinoC <- dados1[1:1000,9]
testeC <- dados1[1001:2000,9]

for(i in 1:30){

pr <- knn(treino,teste,cl=treinoC,k=i)
tab <- table(pr,testeC)
vet[i] <- accuracy(tab)/100

}
```

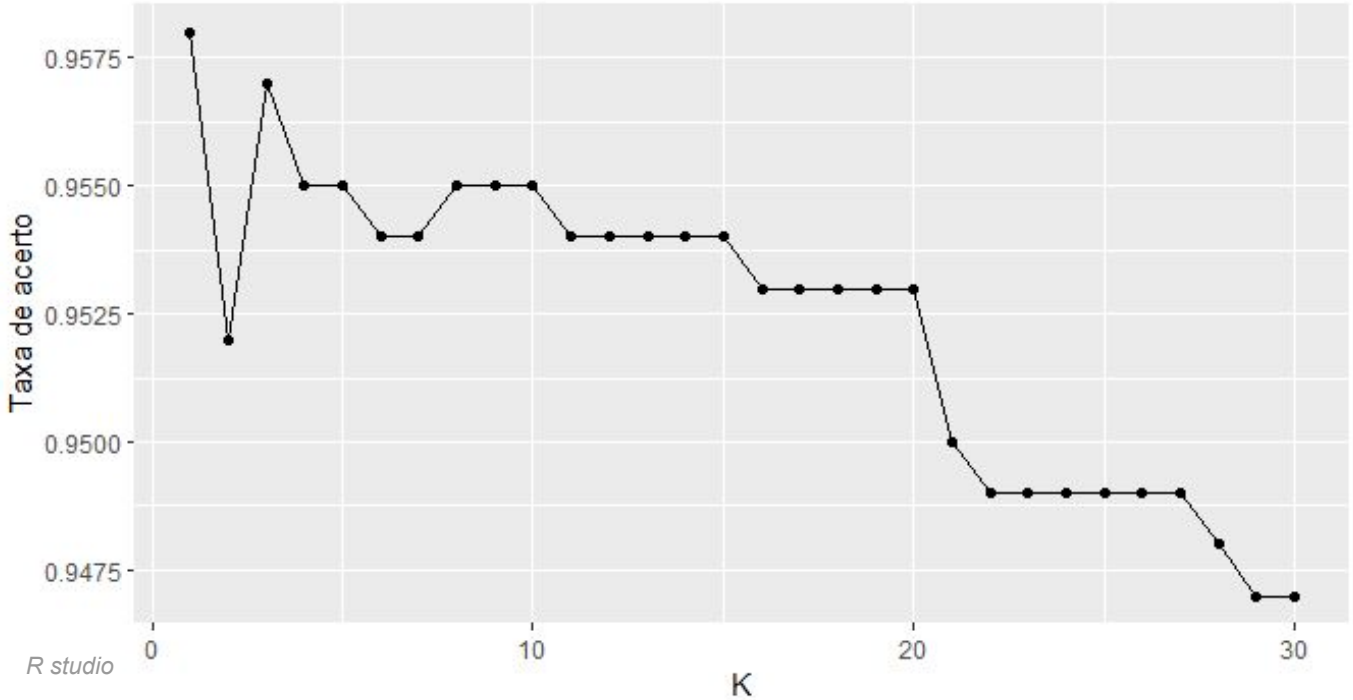
*R studio*

# Experimento: *K-fold CV* aplicado em *KNN*

Repetimos os dois experimentos anteriores usando apenas as duas variáveis de maior gini:



# Experimento: *K-fold CV* aplicado em *KNN*




*R studio*

# Conclusão:

- *K-fold Cross Validation* é um método válido de testar a eficiência de parâmetros de um modelo de previsão.
- Estudo de relevância de variáveis é indispensável na construção de modelos de previsão.
- *KNN* é um modelo melhor aplicado em datasets com apenas variáveis numéricas.

# Experimento 2: Conjunto Abalone - Preparação

```r
dados1 <- read.csv(file = "abalone.csv", header = TRUE)

Idade <- c()
for(i in 1:nrow(dados1)){
  if(dados1[i,9]<=5)
    Idade[i] <- "joven"
  else if(dados1[i,9]<=13)
    Idade[i] <- "adulto"
  else
    Idade[i] <- "velho"
}

dados1 <- data.frame(dados1[,-9],Idade)
```

*R studio*

# Experimento 2: Conjunto Abalone - Estudo das variáveis

```
> gini(dados1[,c(2:9)])
[1] 0.2467004 0.2455443 0.2509616 0.2461806 0.2481626 0.2453742 0.2461648
```

Length:
0.2467004

Shucked weight:
0.2481626

Diameter:
0.2455443

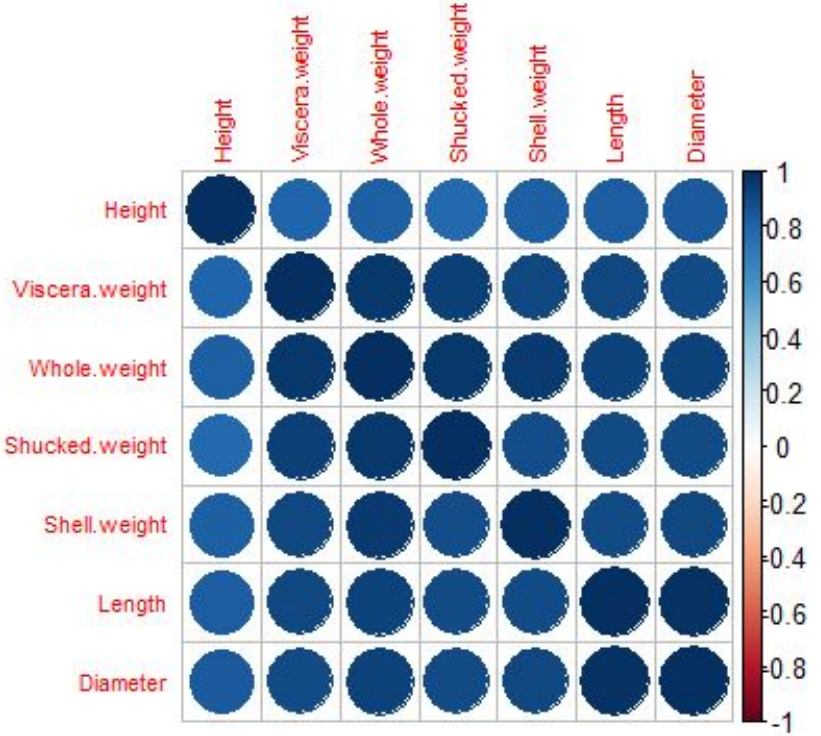
Viscera weight*:
0.2453742

Height:
0.2509616

Shell weight:
0.2461648

Whole weight:
0.2461806

*R studio*

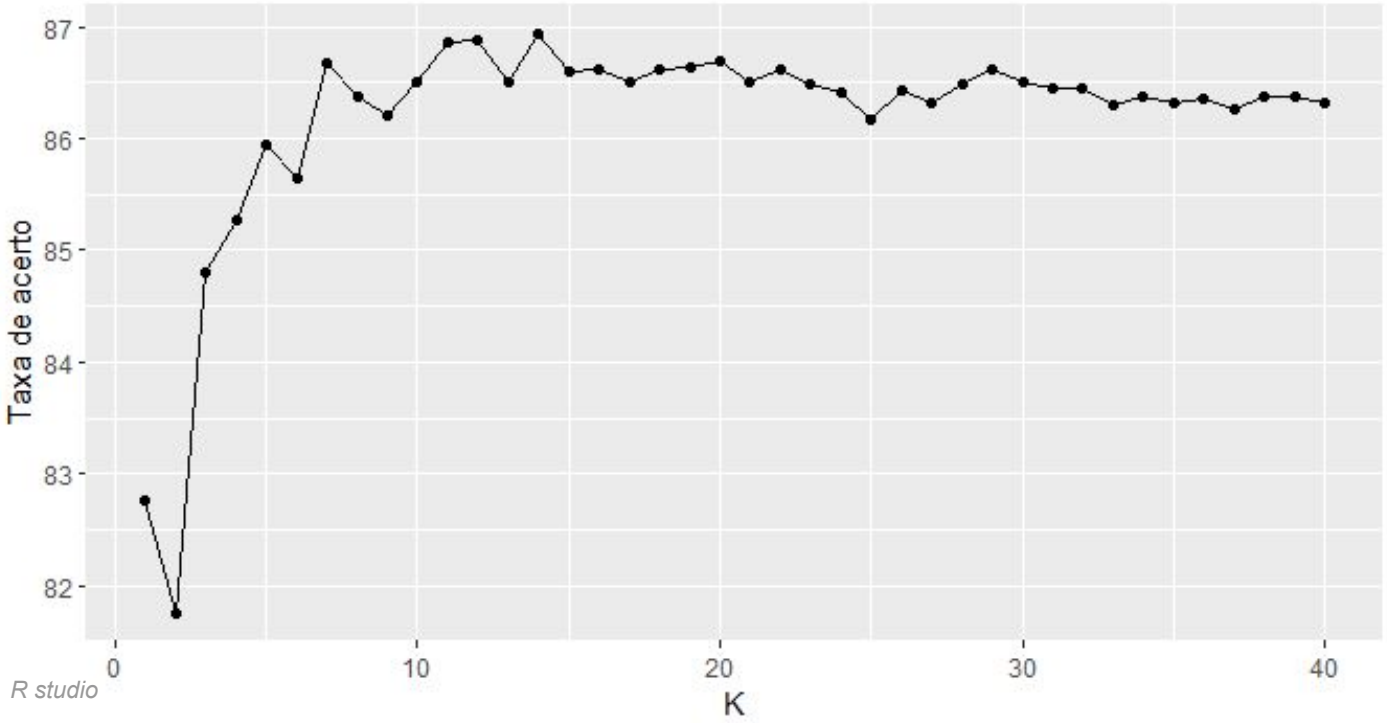


# Experimento 2: Conjunto Abalone - Estudo das variáveis

```r
gini <- function(treino){
  resuls<-c()
  resulsGinis<-c()
  for(j in 1:7){
    ginis <- c()
    vet <- sort(treino[,j])

    for(i in 1:(nrow(treino)-1)){
      med <- (vet[i]+vet[i+1])/2
      a <-sum(treino[treino[,j]<med,]$Idade=="velho")
      b <-sum(treino[treino[,j]<med,]$Idade=="adulto")
      h <-sum(treino[treino[,j]<med,]$Idade=="joven")
      c <- a+b+h
      d <-sum(treino[treino[,j]>=med,]$Idade=="velho")
      e <-sum(treino[treino[,j]>=med,]$Idade=="adulto")
      k <-sum(treino[treino[,j]>=med,]$Idade=="joven")
      f <- d+e+k
      g <- c+f

      if(c!=0){gi1 <- c/g * (1 - (a/c)**2 - (b/c)**2 - (h/c)**2)
      }else{gi1 <- 0}

      if(f!=0){gi2 <- f/g * (1 - (d/f)**2 - (e/f)**2 - (k/f)**2)
      }else{gi2 <- 0}

      ginis[i] <- gi1 + gi2
    }
    resulsGinis[j] <- min(ginis)
    resuls[j] <- vet[which(min(ginis) == ginis)]
  }
  return(resulsGinis)}
```

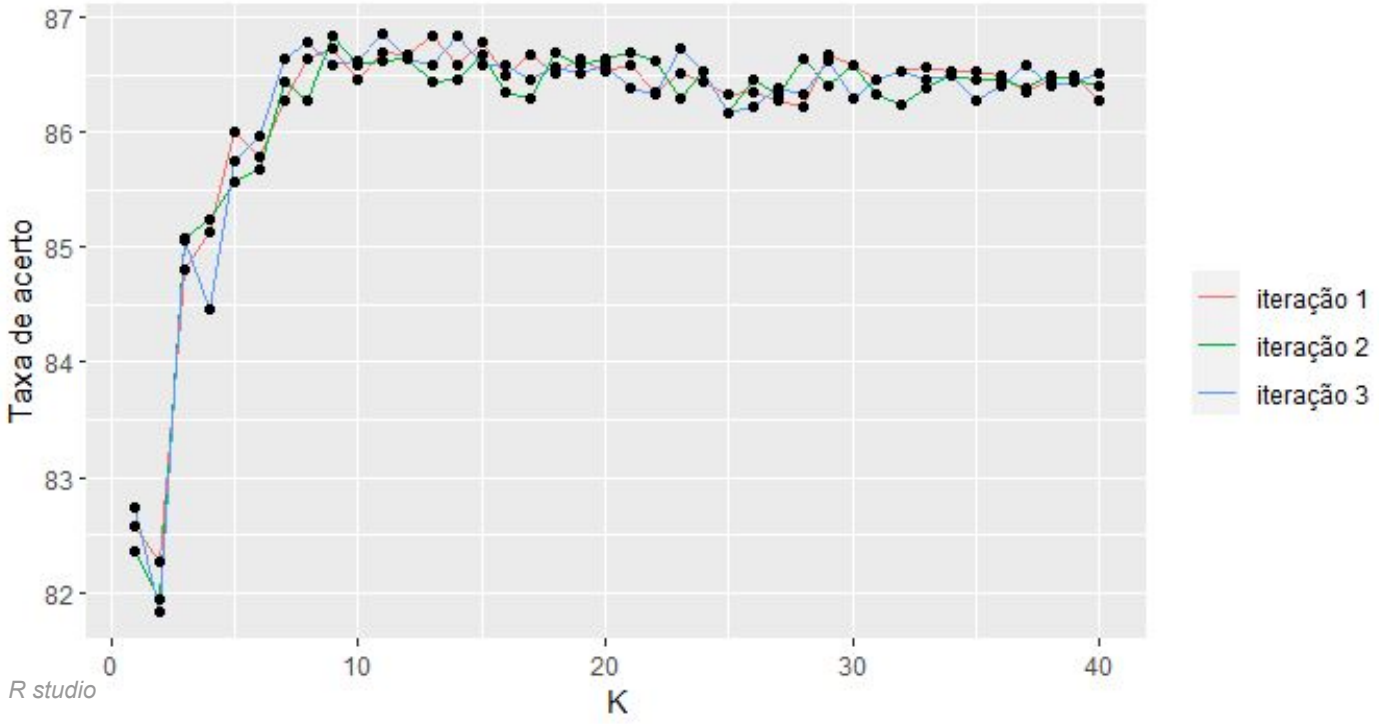
R studio

# Experimento 2: Conjunto Abalone - Aplicação



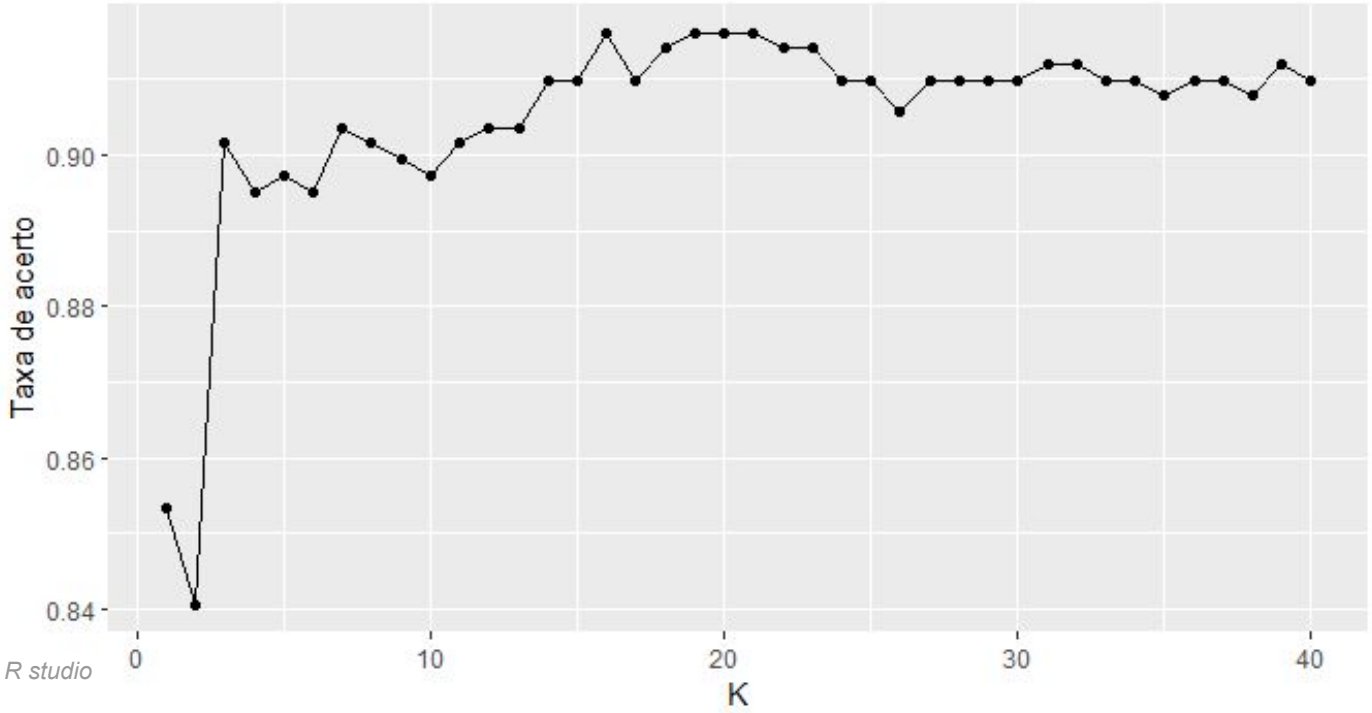

*R studio*

# Experimento 2: Conjunto Abalone - Aplicação



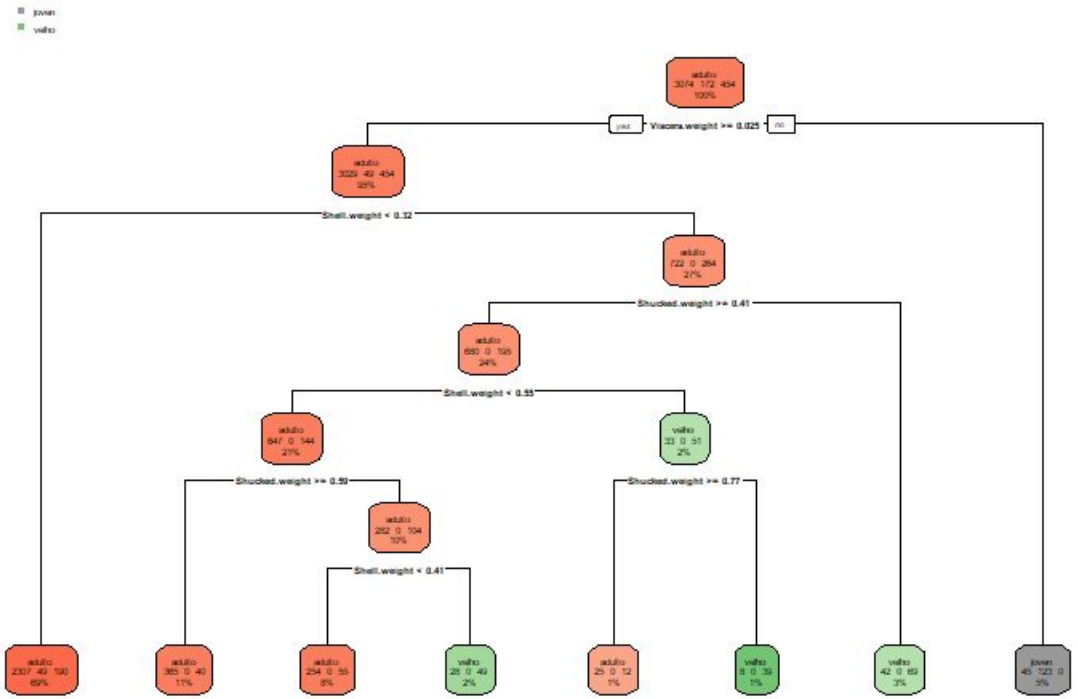

*R studio*

# Experimento 2: Conjunto Abalone - Aplicação*




*R studio*

# Experimento 2: Conjunto Abalone - Comparação*

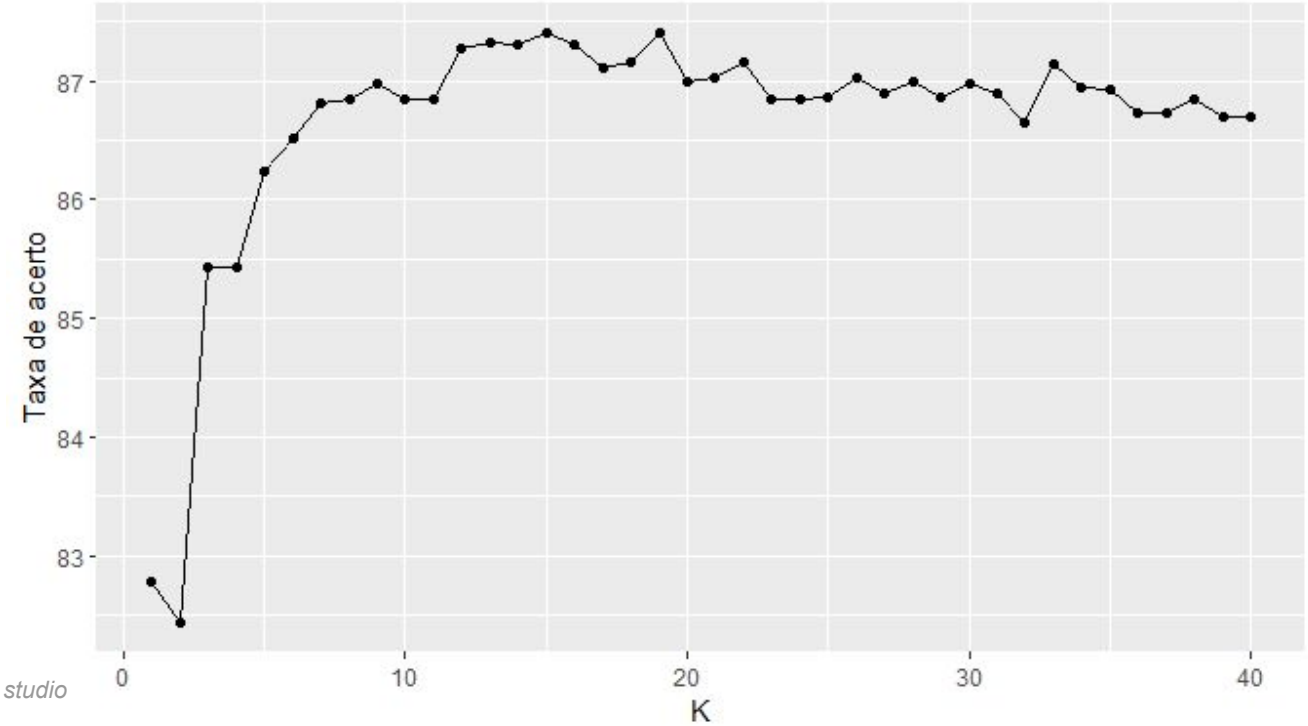



R studio

```
rpart(Idade ~ .,data = treino,
      method="class",
      control=rpart.control(cp=0.012))
```

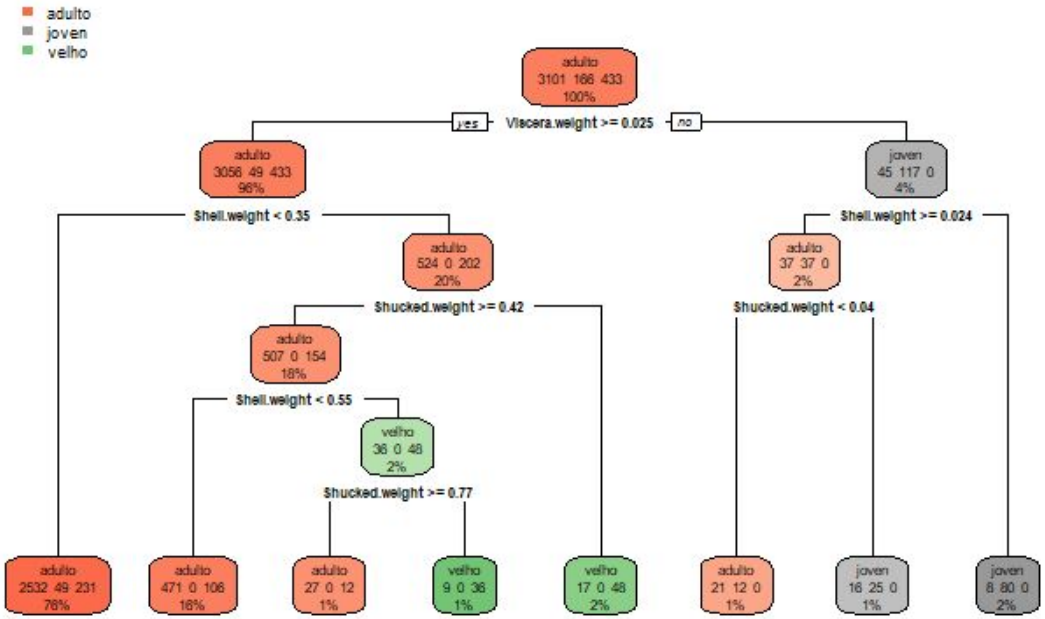
```
> sum(predict
[1] 0.9035639
```

# Experimento 2: Conjunto Abalone - Aplicação




*R studio*

# Experimento 2: Conjunto Abalone - Comparação




```
rpart(Idade ~ .,data = treino,
      method="class",
      control=rpart.control(cp=0.0071))
```

```
> sum(predict
[1] 0.8658281
```

```
randomForest(Idade ~.,
             data=treino,
             proximity=TRUE)
```

```
> sum(predict
[1] 0.8679245
```

*R studio*

# Conclusão:

- *K-fold CV* novamente se mostrou um método válido de testar e estimar parâmetros de um modelo de previsão.
- Um conjunto onde todas variáveis possuem alta correlação e gini ((relativamente)) alto produzirá um modelo menos que ideal.

FIM.