

# Avaliação de modelos de previsão

Baseado em: “*Practical Data Science With R*”

Por: Pedro de Araújo Ribeiro

## Objetivo:

Utilizar de métricas de performance e proporções de acerto para determinar qual modelo de previsão melhor atende uma determinada tarefa.

# Considerações:

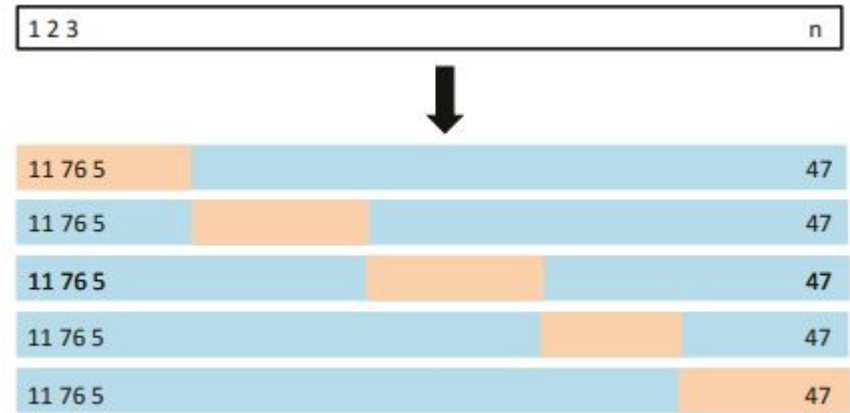
- Dataset usado: [Diabetes prediction](#)
- Todas iterações de *K-fold CV* utilizam  $K=10$
- Bibliotecas do R Studio usadas:

```
library(tidyverse)
library(class)
library(rpart)
library(rpart.plot)
library(wrapr)
library(WVPlots)
library(randomForest)
```

*R studio*

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

K-fold Cross Validation será aplicado em cada modelo de previsão que testaremos a fim de melhor estimar sua eficiência mas também para descobrir quais parâmetros levam a melhor performance no caso de modelos como KNN e florestas aleatórias.



# Análise de tarefa:

Diferentes modelos atendem diferentes tarefas e conjuntos de dados, especificamente temos 3 tipos distintos de modelos estatísticos:

- Pontuação, onde estimamos um valor quantitativo
- Classificação, onde atribuímos um valor qualitativo
- Agrupamento, onde buscamos padrões nos dados

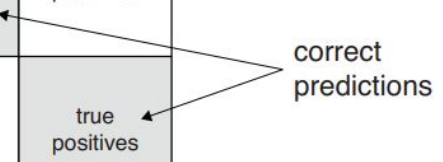
Para nossos estudos, focaremos apenas em modelos de classificação e suas aplicações.

# Avaliação de modelos: Matriz de confusão

A matriz de confusão é uma ferramenta de imensa importância pois quantifica não apenas a acurácia de um modelo mas também a natureza de suas previsões, o quão tendencioso ele é, e os tipos de erro que ele mais comete.

A partir dessas informações poderemos avaliar a performance em conjuntos de dados anormais onde há disparidades entre a proporção de certas classificações ou onde a acurácia geral não é a mais importante

		prediction	
		NEGATIVE	POSITIVE
true label	NEGATIVE	true negatives	false positives
	POSITIVE	false negatives	true positives



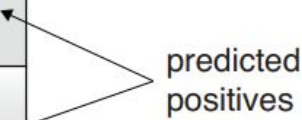
The diagram illustrates the confusion matrix with labels for true labels (rows) and predictions (columns). The cells are labeled: true negatives, false positives, false negatives, and true positives. Arrows point from the text 'correct predictions' to the 'true negatives' and 'true positives' cells, indicating that these represent correct classifications.

# Quadrantes da matriz: Precision

Precisão é a medida de acertos dentre as observações que foram classificadas como positivas.

Acertos em positivos/Total de positivos previstos

		prediction	
		NEGATIVE	POSITIVE
true label	NEGATIVE	true negatives	false positives
	POSITIVE	false negatives	true positives

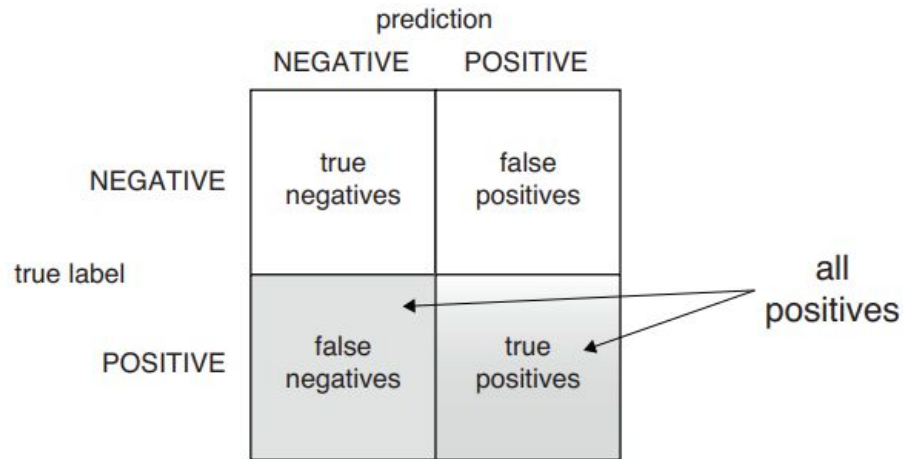


predicted positives

# Quadrantes da matriz: Recall

Recall é a medida de acertos dentre quantidade total de positivos reais.

Acertos em positivos/Total de positivos reais





# Quadrantes da matriz: Sensitivity

Sensitividade é a medida de acertos dentre as observações que foram classificadas como negativas.

Acertos em negativos/Total de negativos previstos

# Quadrantes da matriz: Specificity

Especificidade é a medida de acertos dentre quantidade total de negativos reais.

Acertos em negativos/Total de negativos reais

		prediction	
		NEGATIVE	POSITIVE
true label	NEGATIVE	true negatives	false positives
	POSITIVE	false negatives	true positives

all negatives

# Exemplo prático: Conjunto de dados e objetivo

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
1	Female	80	0	1	never	25.19	6.6	140	0
2	Female	54	0	0	No Info	27.32	6.6	80	0
3	Male	28	0	0	never	27.32	5.7	158	0
4	Female	36	0	0	current	23.45	5.0	155	0
5	Male	76	1	1	current	20.14	4.8	155	0
6	Female	20	0	0	never	27.32	6.6	85	0
7	Female	44	0	0	never	19.31	6.5	200	1
8	Female	79	0	0	No Info	23.86	5.7	85	0
9	Male	42	0	0	never	33.64	4.8	145	0
10	Female	32	0	0	never	27.32	5.0	100	0
11	Female	53	0	0	never	27.32	6.1	85	0
12	Female	54	0	0	former	54.70	6.0	100	0
13	Female	78	0	0	former	36.05	5.0	130	0
14	Female	67	0	0	never	25.69	5.8	200	0
15	Female	76	0	0	No Info	27.32	5.0	160	0

## Exemplo prático: Conjunto de dados e objetivo

O conjunto Diabetes possui majoritariamente pessoas que não possuem a doença, ou seja, observações cuja variável diabetes é negativa.

Portanto, para nossas análises de performance não mediremos a acurácia geral, que é o *total de previsões corretas/total de observações*, e sim a capacidade do modelo de identificar pessoas com diabetes, que são os *acertos em positivos/total de positivos reais*.

De agora pra frente, a capacidade em questão será chamada de ‘acurácia relevante’

# Exemplo prático: Null Model

Inicialmente não utilizamos nenhum modelo real e sim um chamado *null model*, que consiste de uma previsão estática para todas as observações

```
actual <- teste$diabetes  
pred <- rep(0, nrow(teste))  
t1 <- table(actual, pred)
```

	pred
actual	0
0	9200
1	800

R studio

Neste caso, a acurácia relevante será 0

# Exemplo prático: KNN

Para KNN realizamos K-fold CV para identificar a quantidade de vizinhos que resulta na melhor acurácia relevante.

No final, faremos uma previsão com o melhor parâmetro encontrado.

R studio

```
dados_norm = scale(dados1[-c(1,5,9)])

Ks <- seq(from = 10, to = 100, by = 10)
det2 <- c()

for(j in Ks){

  detection <- c()

  for(i in 1:10){
    teste_knn <- kFoldPartitionTeste(dados_norm,10,i)
    treino_knn <- kFoldPartitionTreino(dados_norm,10,i)
    actual_knn <- kFoldPartitionTeste(dados1,10,i)$diabetes
    treino2 <- kFoldPartitionTreino(dados1,10,i)

    test_pred <- knn(
      train = treino_knn,
      test = teste_knn,
      cl = treino2$diabetes,
      k=j
    )

    cm <- table(actual_knn,test_pred)
    detection <- c(detection,cm[2,2]/(cm[2,2]+cm[2,1]))

  }

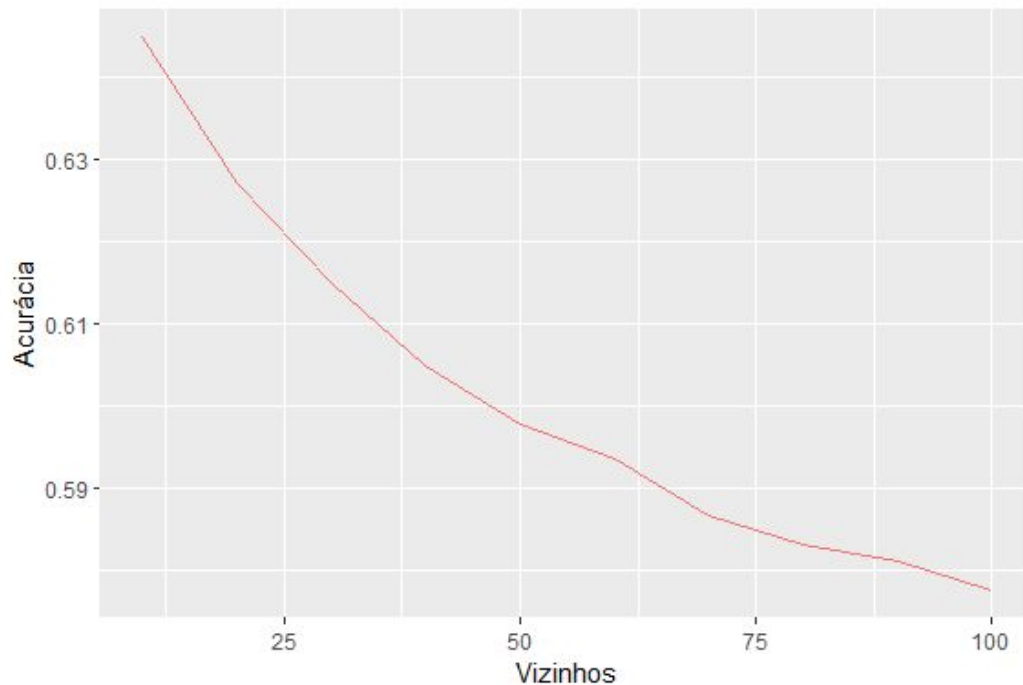
  det2 <- c(det2,mean(detection))
}
```

# Exemplo prático: KNN

Aqui concluímos que uma quantidade maior de vizinhos leva a uma acurácia pior.

```
> max(det2)
[1] 0.6448449
> ks[which(det2==max(det2))]
[1] 10
```

*R studio*

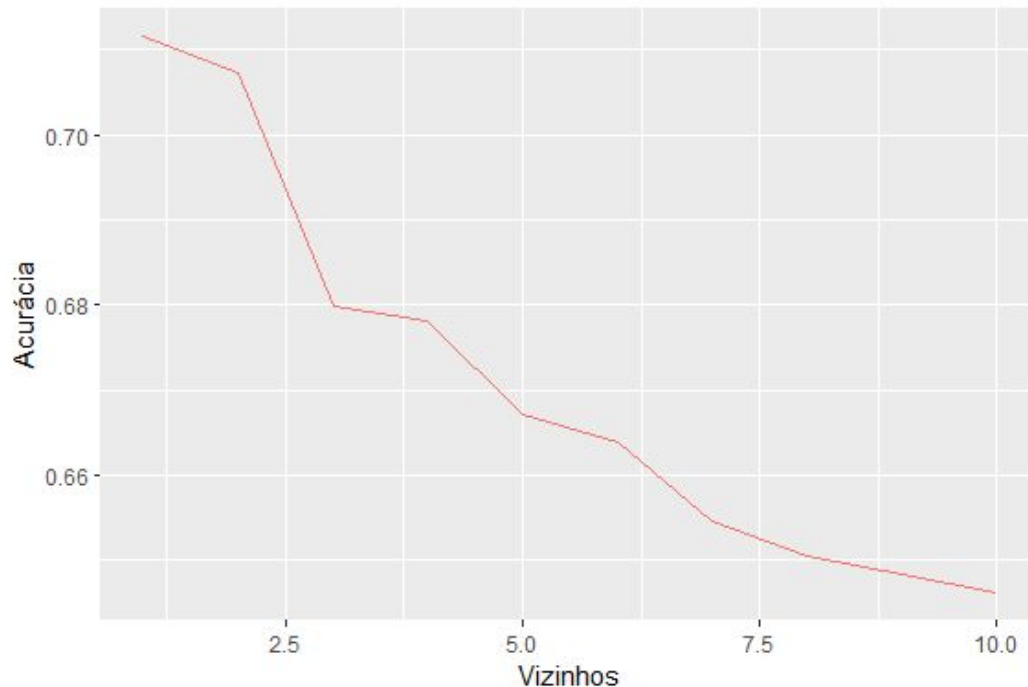


# Exemplo prático: KNN

Ao observar a tendência anterior o experimento foi replicado para o intervalo de vizinhos 1 a 10 e observamos que 1 vizinho resultou na acurácia mais alta

```
> max(det2)
[1] 0.7115987
> Ks[which(det2==max(det2))]
[1] 1
```

*R studio*





# Exemplo prático: Árvore

Como árvores não possuem parâmetros a serem testados do mesmo jeito que KNN e floresta aleatória, apenas realizamos o K-fold CV para obter uma média de acurácia que será comparada as demais.

```
> mean(detection)
[1] 0.7107549
```

*R studio*

```
detection <- c()

for(i in 1:10){

  teste_arv <- kFoldPartitionTeste(dados1,10,i)
  treino_arv <- kFoldPartitionTreino(dados1,10,i)
  actual_arv <- teste_arv$diabetes

  modelo <- rpart(diabetes ~ .,data = treino_arv,
                  method="class",control=rpart.control(cp=0))

  pred1 <- predict(modelo,teste_arv,"class")

  tab <- table(actual_arv,pred1)

  detection <- c(detection,tab[2,2]/(tab[2,1]+tab[2,2]))
}

mean(detection)
```

# Exemplo prático: Florestas aleatórias

Para florestas aleatórias realizaremos o mesmo processo que KNN para identificar a quantidade ideal de árvores.

```
fs <- seq(from = 300, to = 1500, by = 200)
det4 <- c()

for(j in fs){

  detection <- c()

  for(i in 1:10){
    teste_fore <- kFoldPartitionTeste(dados1,10,i)
    treino_fore <- kFoldPartitionTreino(dados1,10,i)
    actual_fore <- teste_fore$diabetes

    modelo2 <- randomForest(diabetes ~.,data=treino_fore,ntree=j)

    pred2 <- predict(modelo2,teste_fore,"class")

    cm2 <- table(actual_fore,pred2)
    detection <- c(detection,cm2[2,2]/sum(cm2[,2]))
  }
  print(j)
  det4 <- c(det4,mean(detection))
}
```

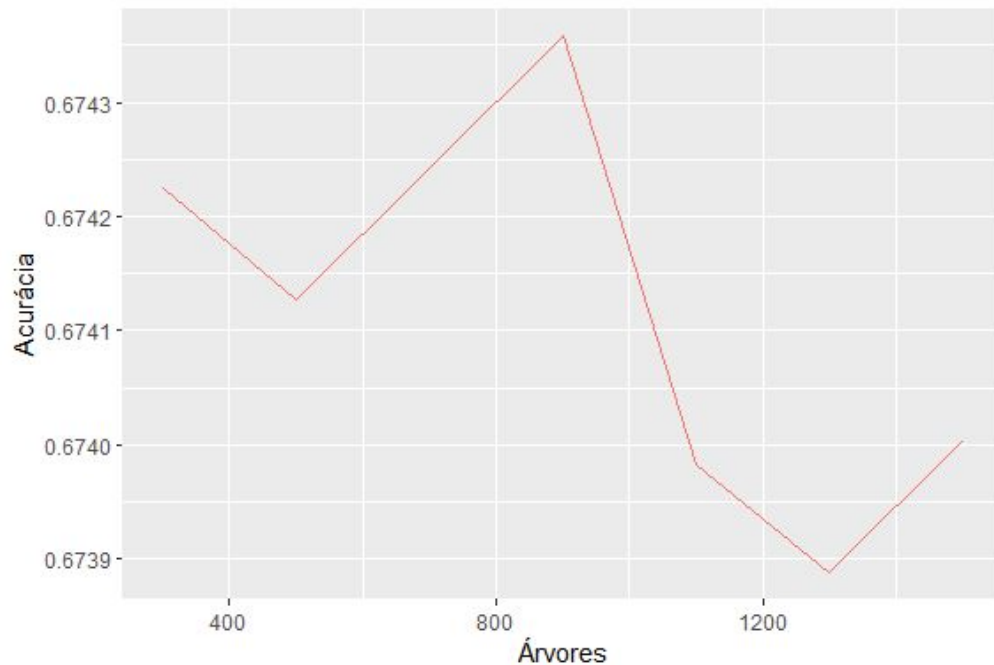
R studio

# Exemplo prático: Florestas aleatórias

Aqui observamos que 900 árvores produziram o melhor resultado.

```
> max(det4)
[1] 0.6743584
> fs[which(det4==max(det4))]  
[1] 900
```

*R studio*



FileEditCodeViewPlotsSessionBuildDebugProfileToolsHelp

Go to file/functionAddins

semana 19.R\*semana 19-2.R\*

Source on SaveRunSource

```
131
132
133 detection <- c()
134
135 for(i in 1:10){
136   teste_fore <- kFoldPartitionTeste(dados1,10,i)
137   treino_fore <- kFoldPartitionTreino(dados1,10,i)
138   actual_fore <- teste_fore$diabetes
139
140   modelo2 <- randomForest(diabetes ~.,data=treino_fore,ntree=j)
141
142   pred2 <- predict(modelo2,teste_fore,"class")
143
144   cm2 <- table(actual_fore,pred2)
145   detection <- c(detection,cm2[2,2]/sum(cm2[,2]))
146 }
147
148 det4 <- c(det4,mean(detection))
149 }
150
151 ggplot(mapping = aes(x=fs,y=det4,col="darkred"))+
152   geom_line() +
153   labs(x="Árvores",y="Acurácia")
154
155 #Logística
156
157 modelo3 <- glm(diabetes~., data = treino, family = binomial
164:66 (Top Level) :
```

EnvironmentHistoryConnectionsTutorial

Global Environment

treino290000 obs. of 9 variables

Values

actualFactor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...

actual\_arvFactor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...

actual\_knnFactor w/ 2 levels "0","1": 1 1 1 1 1 1 1 2 1 1 ...

cm'table' int [1:2, 1:2] 9121 298 40 541

det2num [1:10] 0.712 0.707 0.68 0.678 0.667 ...

det4NULL

1:100.894 0.903 0.868 0.871 0.886 ...

1:14200 300 400 500 600 700 800 900 1000 1100 ...

PresentationPublish

ConsoleTerminalBackground Jobs

R 4.3.0 - G:/Meu Drive/UFU/IC/repositorio/conjuntos/

+ modelo2 <- randomForest(diabetes ~.,data=treino\_fore,ntree=j)

+ pred2 <- predict(modelo2,teste\_fore,"class")

+ cm2 <- table(actual\_fore,pred2)


+ detection <- c(detection,cm2[2,2]/sum(cm2[,2]))

+ }

+ det4 <- c(det4,mean(detection))

+ }

Memory Usage Report (88% in use)



88%  
memory in use

Statistic	Memory	Source
Used by R objects	9,157 MiB	R
Used by session	9,306 MiB	Windows System
Used by system	5,071 MiB	Windows System
Free system memory	1,874 MiB	Windows System
Total system memory	16,253 MiB	Windows System

OK

Actual

0.66

2.55.07.510.0

Vizinhas

colour

darkred

Pesquisar

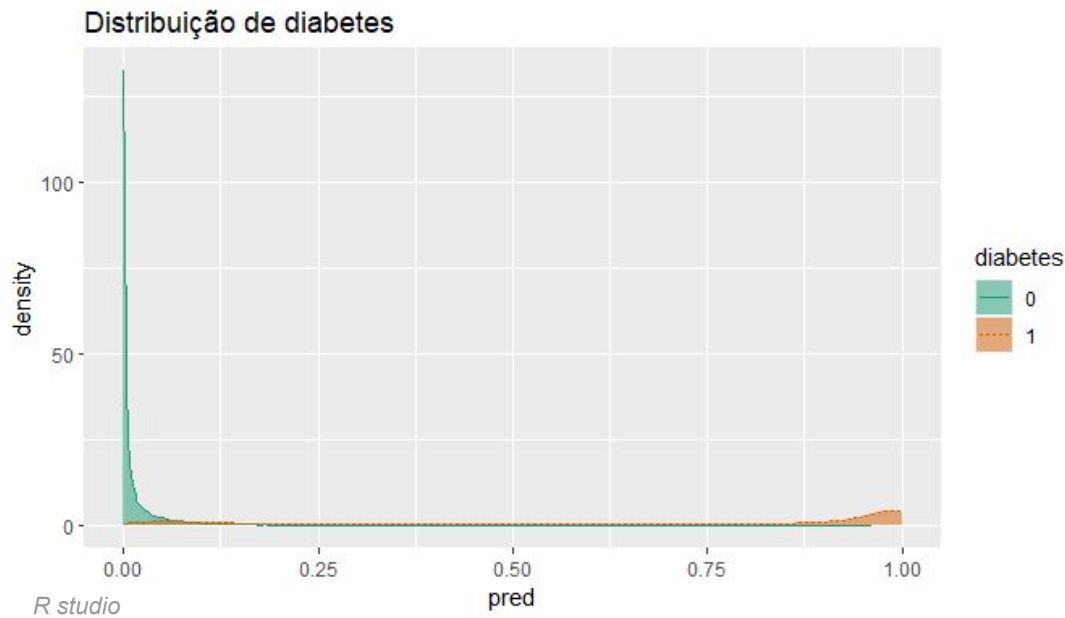
34°C Ensolarado

13:29 11/11/2023

# Exemplo prático: Regressão logística

Neste caso deveremos analisar não apenas o aumento na acurácia relevante mas também na acurácia para negativos, pois seria muito fácil simplesmente assumir todos como positivos.

Escolheremos arbitrariamente manter uma taxa de acerto de 90% para os negativos reais



# Exemplo prático: Regressão logística

```
for(j in 1:10){  
  teste_logit <- kFoldPartitionTeste(dados1,10,j)  
  treino_logit <- kFoldPartitionTreino(dados1,10,j)  
  
  modelo3 <- glm(diabetes~., data = treino_logit,  
                 family = binomial(link = "logit"))  
  
  teste1 <- teste_logit  
  
  teste1$pred <- predict(modelo3, newdata=teste_logit, type="response")  
  treino_logit$pred <- predict(modelo3, newdata=treino_logit, type = "response")  
  
  DoubleDensityPlot(treino_logit, "pred", "diabetes",  
                    title = "Distribuição de diabetes")  
  
  i <- 0  
  l <- 0.01  
  while(i < 0.90){  
    tab4 <- table(actual = teste1$diabetes, pred = teste1$pred > l)  
    l <- l + 0.01  
    i <- tab4[1,1]/(tab4[1,1]+tab4[1,2])  
  }  
  det5 <- c(det5, tab4[2,2]/(tab4[2,2]+tab4[2,1]))  
  cuts <- c(cuts, l)  
}
```

# Exemplo prático: Regressão logística

Para o experimento anterior obtivemos a acurácia relevante média e observamos que quase sempre o mesmo ponto de split levava aos melhores resultados, portanto o usaremos na análise final.

```
> cuts
[1] 0.11 0.11 0.11 0.10 0.11 0.11 0.11 0.10 0.11 0.12
> mean(det5)
[1] 0.8569157
```

*R studio*



# Exemplo prático: Análise final

Utilizando todos os parâmetros ideais obtidos, sendo eles:

- K=1 para KNN
- T=900 para florestas
- Split = 0.11 para regressão logística

Realizaremos uma previsão final para um subconjunto aleatório da base de dados e compararemos a performance dos modelos entre si e também com seus resultados no K-fold CV.

```
dados_fim <- shuffle(dados)
treino_fim <- kFoldPartitionTreino(dados_fim,10,5)
teste_fim <- kFoldPartitionTeste(dados_fim,10,5)
```



# Exemplo prático: Análise final

	Null	KNN	Arvore	Floresta	Logistica
Acuracia	0.9199	0.955100	0.9676000	0.9738000	0.9061000
Positivos	801.0000	801.000000	801.0000000	801.0000000	801.0000000
Negativos	9199.0000	9199.000000	9199.0000000	9199.0000000	9199.0000000
Falsos positivos	0.0000	217.000000	90.0000000	4.0000000	809.0000000
Falsos negativos	801.0000	232.000000	234.0000000	258.0000000	130.0000000
Verdadeiros positivos	0.0000	569.000000	567.0000000	543.0000000	671.0000000
Veradeiros negativos	9199.0000	8982.000000	9109.0000000	9195.0000000	8390.0000000
% de positivos detectados	0.0000	0.710362	0.7078652	0.6779026	0.8377029

*R studio*

# Exemplo prático: Análise final - KNN

```
> max(det2)
[1] 0.7115987
> Ks[which(det2==max(det2))]
[1] 1
```

```
> cm[2,2]/(cm[2,1]+cm[2,2])
[1] 0.710362
```

```
#KNN
treino_norm = scale(treino_fim[-c(1,5,9)])
teste_norm = scale(teste_fim[-c(1,5,9)])

test_pred <- knn(
  train = treino_norm,
  test = teste_norm,
  cl = treino_fim$diabetes,
  k=1
)

cm <- table(actual, test_pred)

matriz[,2]<-fillMatriz(cm)
```

# Exemplo prático: Análise final - Árvore

```
> mean(detection)
[1] 0.7107549
```

```
> tab[2,2]/(tab[2,1]+tab[2,2])
[1] 0.7067938
```

```
#Árvore
modelo <- rpart(diabetes ~ ., data = treino_fim,
                method="class", control=rpart.control(cp=0))

pred1 <- predict(modelo, teste_fim, "class")

tab <- table(actual, pred1)

matriz[,3] <- fillMatriz(tab)

tab[2,2]/(tab[2,1]+tab[2,2])
```

# Exemplo prático: Análise final - Floresta aleatória

```
> max(det4)
[1] 0.6743584
> fs[which(det4==max(det4))]
[1] 900
```

```
> tab2[2,2]/(tab2[2,1]+tab2[2,2])
[1] 0.6779026
```

*#Floresta*

```
modelo2 <- randomForest(diabetes ~.,
                        data=treino_fim, ntree=900)

pred2 <- predict(modelo2, teste_fim, "class")

tab2 <- table(actual, pred2)

matriz[,4] <- fillMatriz(tab2)
```

# Exemplo prático: Análise final - Regressão logística

```
> cuts  
[1] 0.11 0.11 0.11 0.10  
> mean(det5)  
[1] 0.8569157
```

```
> ctab.test[2,2]/  
[1] 0.8377029
```

*#logística*

```
modelo3 <- glm(diabetes~., data = treino_fim,  
               family = binomial(link = "logit"))  
  
teste1 <- teste_fim  
  
teste1$pred <- predict(modelo3,  
                       newdata=teste_fim, type="response")  
  
ctab.test <- table(actual = teste_fim$diabetes,  
                   pred = teste1$pred > 0.11)  
  
matriz[,5]<-fillMatriz(ctab.test)
```

## Conclusão:

- Quando realizamos previsões e analisamos a performance de modelos, devemos levar em consideração a natureza da variável resposta e o objetivo da previsão
- Métodos que priorizam apenas a acurácia geral e o ganho de pureza de partições como árvores e florestas não são capazes dessa nuance sem alterações no seu funcionamento
- Para conjuntos com uma disparidade muito grande na proporção de variáveis resposta, KNN com muitos vizinhos tende a perder acurácia para a variável em menor número
- Na regressão logística devemos encontrar um ponto de trade-off entre verdadeiros positivos e verdadeiros negativos

**FIM.**