Machine Learning aplicado em previsões de insuficiência cardíaca

Alexandre Turatto Henrique

28/09/2021

Para esse projeto, usaremos um Dataset disponibilizado por Davide Chicco, no Kaggle.

A ideia é descobrir, através de machine learning, a chance de um paciente sobreviver a uma insuficiência cardíaca, utilizando apenas como variáveis a Creatinina Sérica no sangue e o Percentual de sangue saindo do coração por contração.

A fonte desse estudo está disponibilizada abaixo:

Dataset from Davide Chicco, Giuseppe Jurman: Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical Informatics and Decision Making 20, 16 (2020)

https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-020-1023-5

Agora, vamos entender as colunas do Dataset:

- Age: Idade
- Anaemia: Redução de Glóbulos Vermelhos (Hemoglobina)
- Diabetes: Se o paciente tem diabetes ou não
- creatinine_phosphokinase: Nível da enzima CPK no sangue (mcg/L)
- ejection_fraction: Percentual de sangue saindo do coração a cada Contração
- high_blood_pressure: Se o paciente é Hipertenso
- platelets: Plaquetas no sangue (Kiloplatelets/mL)
- serum_creatinine: Nível de creatinina sérica no sangue (mg/dL)
- serum_sodium: Level of serum sodium in the blood (mEq/L)
- sex: 1 Masculino, 0 Feminino
- smoking: Fumante ou não
- time: Período de acompanhamento(Follow-Up) em dias
- DEATH_EVENT: Se o paciente faleceu durante o período de acompanhamento

Vamos começar o processo de limpeza do dataset e classificar as colunas adequadamente, respeitando as características das variáveis.

```r
skim(df)
```

Table 1: Data summary

Name	df
Number of rows	299
Number of columns	13
Column type frequency:	
factor	5
logical	1
numeric	7
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
anaemia	0	1	FALSE	2	FAL: 170, TRU: 129
diabetes	0	1	FALSE	2	FAL: 174, TRU: 125
high_blood_pressure	0	1	FALSE	2	FAL: 194, TRU: 105
sex	0	1	FALSE	2	TRU: 194, FAL: 105
smoking	0	1	FALSE	2	FAL: 203, TRU: 96

Variable type: logical

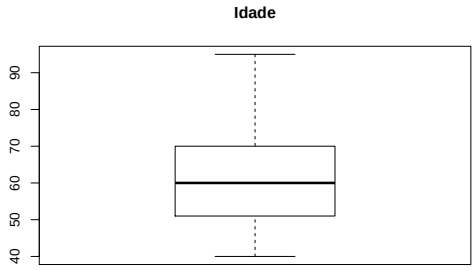
skim_variable	n_missing	complete_rate	mean	count
DEATH_EVENT	0	1	0.32	FAL: 203, TRU: 96

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75
age	0	1	60.83	11.89	40.0	51.0	60.0	70.0
creatinine_phosphokinase	0	1	581.84	970.29	23.0	116.5	250.0	582.0
ejection_fraction	0	1	38.08	11.83	14.0	30.0	38.0	45.0
platelets	0	1	263358.03	97804.24	25100.0	212500.0	262000.0	303500.0
serum_creatinine	0	1	1.39	1.03	0.5	0.9	1.1	1.4
serum_sodium	0	1	136.63	4.41	113.0	134.0	137.0	140.0
time	0	1	130.26	77.61	4.0	73.0	115.0	203.0

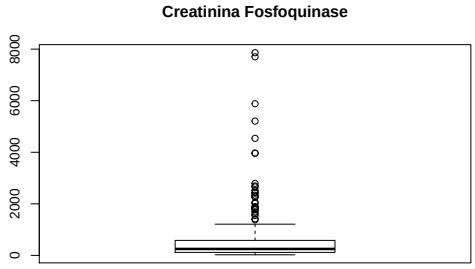
Agora iremos fazer uma análise dos outliers das variáveis numéricas do modelo, com o intuito de entender melhor a distribuição sem a interferência de valores extremos. Como a escala das variáveis é bem diferente, optei por plotar um a um para facilitar a visualização.

```r
boxplot(df$age, main = "Idade")
```



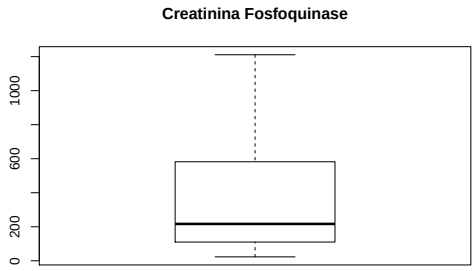
A variável age não possui outliers.

```
boxplot(df$creatinine_phosphokinase, main = "Creatinina Fosfoquinase")
```



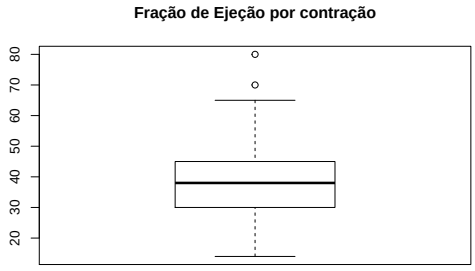

```
# A variável creatinine_phosphokinase possui outliers, então vamos retira-los para
# entender melhor as correlações.

df = df %>%
  filter(creatinine_phosphokinase <= 1300)
boxplot(df$creatinine_phosphokinase, main = "Creatinina Fosfoquinase")
```

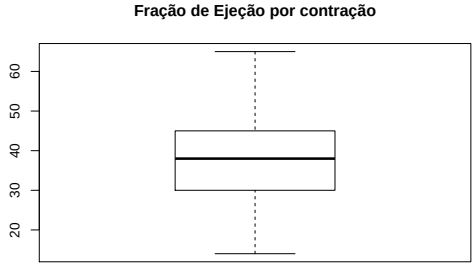


Ajustado.

```
boxplot(df$ejection_fraction, main = "Fração de Ejeção por contração")
```

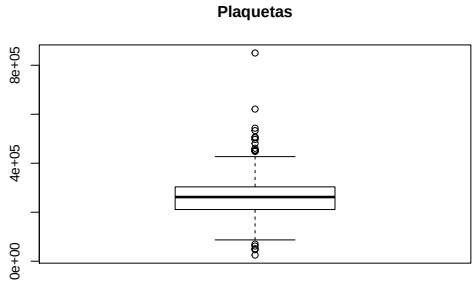



```
# Mesma coisa aqui.
df = df %>%
  filter(ejection_fraction <= 66)
boxplot(df$ejection_fraction, main = "Fração de Ejeção por contração")
```

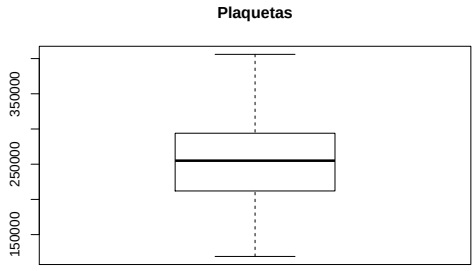



```
# Ejection_fraction Limpo.

boxplot(df$platelets, main = "Plaquetas")
```

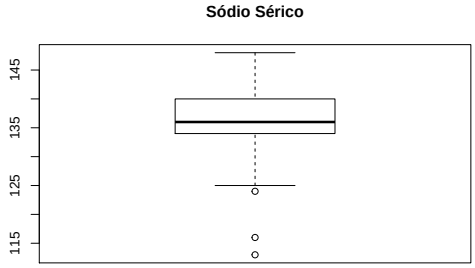



```
# A variável Plaquetas também
df = df %>%
  filter(platelets>=1e+05)
df = df %>%
  filter(platelets<=4.1e+05)
boxplot(df$platelets, main = "Plaquetas")
```

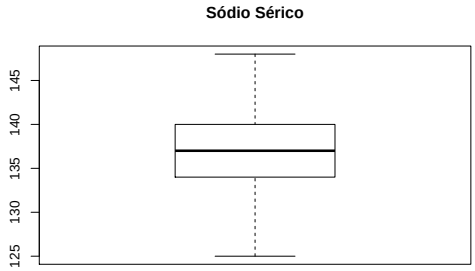



```
# Platelets Limpo.

boxplot(df$serum_sodium, main = "Sódio Sérico")
```




```
# Aqui também
df = df%>%
  filter(serum_sodium >=125)
boxplot(df$serum_sodium, main = "Sódio Sérico")
```




```
# serum_sodium Limpo.

# Após essas transformações, vamos entender o resumo dos nossos dados:
summary(df)
```

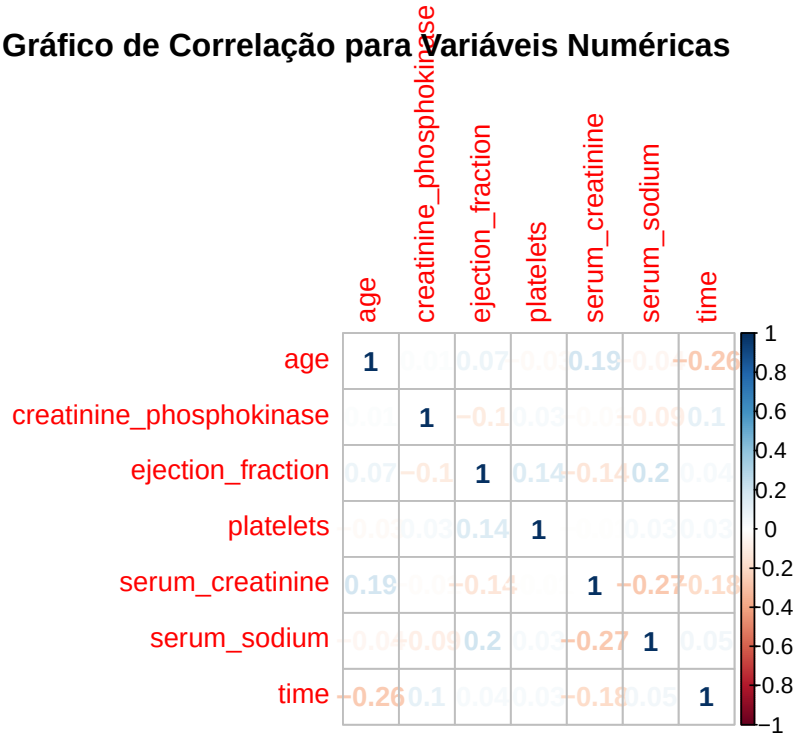
```
##       age         anaemia    creatinine_phosphokinase  diabetes
```

```
##  Min.   :40.00   FALSE:131   Min.   :  23.0          FALSE:143
##  1st Qu.:52.00   TRUE :112   1st Qu.: 109.5          TRUE :100
##  Median :60.00               Median : 211.0
##  Mean   :61.29               Mean   : 328.5
##  3rd Qu.:70.00               3rd Qu.: 582.0
##  Max.   :95.00               Max.   :1202.0
##  ejection_fraction high_blood_pressure   platelets      serum_creatinine
##  Min.   :14.00     FALSE:153           Min.   :119000   Min.   :0.60
##  1st Qu.:30.00     TRUE : 90           1st Qu.:211000   1st Qu.:0.90
##  Median :38.00                         Median :255000   Median :1.10
##  Mean   :37.98                         Mean   :252555   Mean   :1.36
##  3rd Qu.:45.00                         3rd Qu.:293500   3rd Qu.:1.40
##  Max.   :65.00                         Max.   :406000   Max.   :9.40
##   serum_sodium      sex       smoking         time       DEATH_EVENT
##  Min.   :125.0   FALSE: 84   FALSE:164   Min.   :  4.0   Mode :logical
##  1st Qu.:134.0   TRUE :159   TRUE : 79   1st Qu.: 74.0   FALSE:169
##  Median :137.0                           Median :112.0   TRUE :74
##  Mean   :136.8                           Mean   :129.3
##  3rd Qu.:140.0                           3rd Qu.:200.5
##  Max.   :148.0                           Max.   :285.0
```

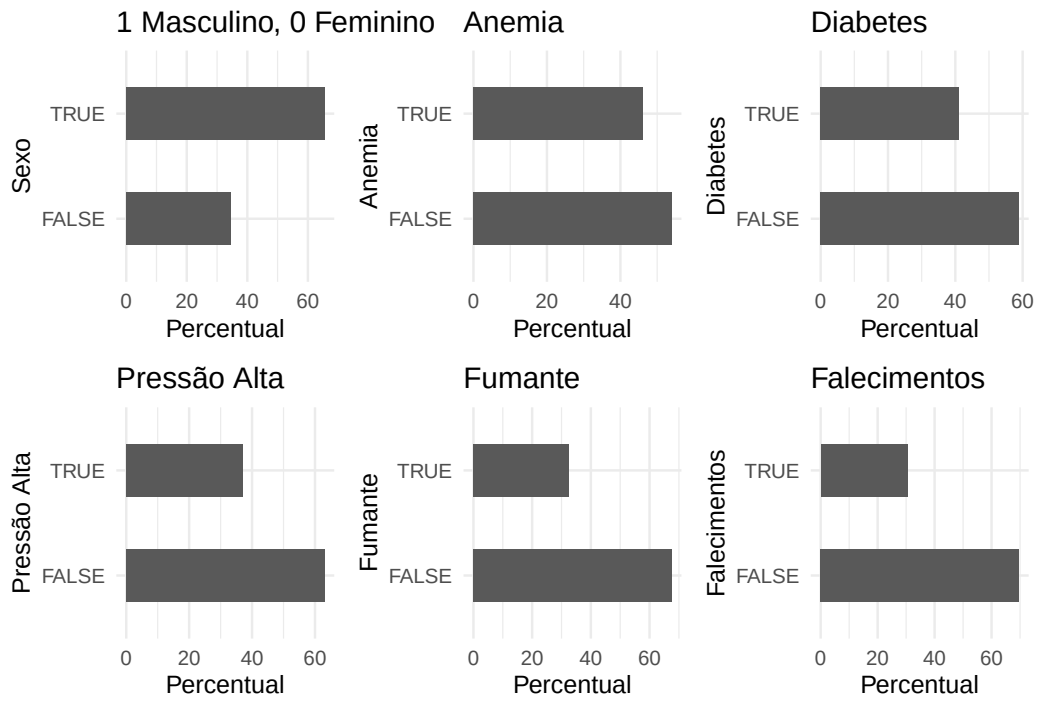
```
# Estamos trabalhando com um dataset que possui 243 registros e 13 colunas.
```

Vamos agora fazer um gráfico para ilustrar qual a correlação dentre as variáveis que são numéricas:

```
numeric.var <- sapply(df, is.numeric)
corr.matrix <- cor(df[,numeric.var])
corrplot(corr.matrix, main="\n\nGráfico de Correlação para Variáveis Numéricas", method="number")
```



Agora vamos analisar cada uma das variáveis categóricas:



Com os dados devidamente tratados, podemos iniciar a modelagem preditiva.

Modelagem preditiva

Primeiro, dividimos os dados em conjuntos de treinamento e testes

```
intrain <- createDataPartition(df$DEATH_EVENT,p=0.7,list=FALSE)
set.seed(2021)
training <- df[intrain,]
testing <- df[-intrain,]
```

Confirmando se a divisão está correta:

```
dim(df)
```

```
## [1] 243  13
```

```
dim(training); dim(testing)
```

```
## [1] 171  13
```

```
## [1] 72 13
```

```
# Conforme o esperado, treino corresponde a 70% do dataset e teste 30%
```

Criando um modelo de regressão logística:

```
LogModel <- glm(DEATH_EVENT ~ ., family=binomial(link="logit"), data=training)
anova(LogModel, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: DEATH_EVENT
##
## Terms added sequentially (first to last)
##
##
##                           Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                                        170     210.09
## age                        1   17.161       169     192.93 3.435e-05 ***
## anaemia                    1    2.686       168     190.24   0.10123
## creatinine_phosphokinase   1    0.096       167     190.15   0.75719
## diabetes                   1    0.034       166     190.11   0.85475
## ejection_fraction          1   19.522       165     170.59 9.945e-06 ***
## high_blood_pressure        1    1.015       164     169.58   0.31372
## platelets                  1    1.601       163     167.97   0.20575
## serum_creatinine           1    4.750       162     163.22   0.02929 *
```

```
## serum_sodium              1    1.915       161     161.31   0.16640
## sex                       1    0.054       160     161.25   0.81600
## smoking                   1    2.195       159     159.06   0.13843
## time                      1   57.754       158     101.31 2.971e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As variáveis mais relevantes são: age, ejection_fraction , serum_creatinine, time

Testando a Regressão logística no dataset de testes:

```
testing$DEATH_EVENT <- as.logical(testing$DEATH_EVENT)
fitted.results <- predict(LogModel,newdata=testing,type='response')
fitted.results <- ifelse(fitted.results > 0.5,1,0)
misClasificError <- mean(fitted.results != testing$DEATH_EVENT)
print(paste('Logistic Regression Accuracy',1-misClasificError))
```

```
## [1] "Logistic Regression Accuracy 0.791666666666667"
```

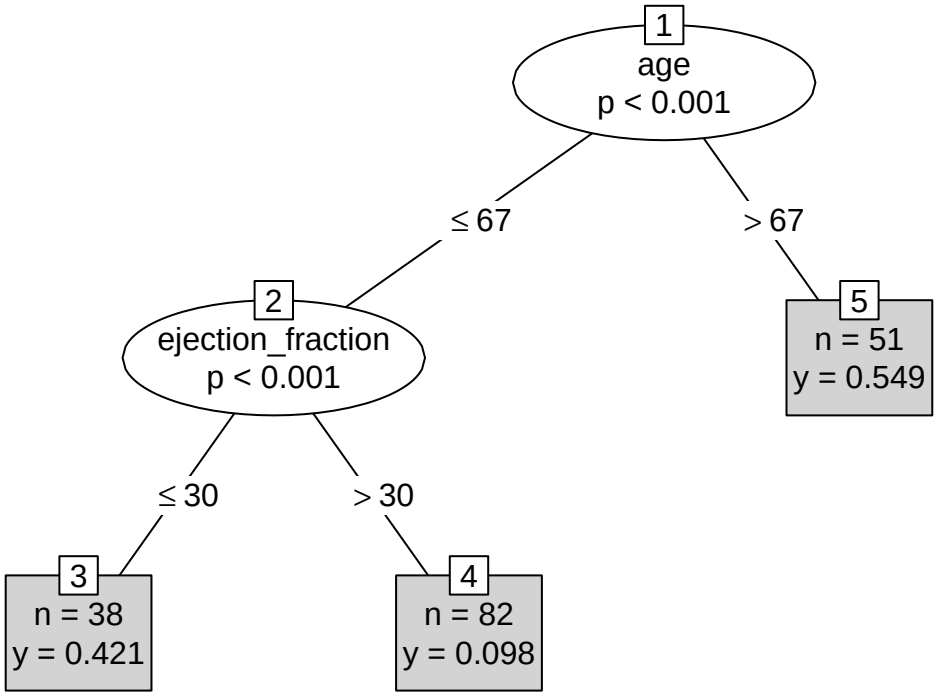
```
print("Confusion Matrix para Regressão Logística"); table(testing$DEATH_EVENT,fitted.results > 0.5)
```

```
## [1] "Confusion Matrix para Regressão Logística"

##
##         FALSE TRUE
##   FALSE    46    4
##   TRUE     11   11
```

Testando Random Forest

```
tree <- ctree(DEATH_EVENT ~ age + ejection_fraction + serum_creatinine ,training)
plot(tree, type='simple')
```




```
pred_tree <- predict(tree, testing)
print("Confusion Matrix Para Decision Tree"); table(Predicted = pred_tree, Actual = testing$DEATH_EVENT)
```

```
## [1] "Confusion Matrix Para Decision Tree"
```

```
##                     Actual
## Predicted            FALSE TRUE
##   0.0975609756097561    30    5
##   0.421052631578947      8    7
##   0.549019607843137     12   10
```

```
p1 <- predict(tree, training)
tab1 <- table(Predicted = p1, Actual = training$DEATH_EVENT)
tab2 <- table(Predicted = pred_tree, Actual = testing$DEATH_EVENT)
print(paste('Decision Tree Accuracy',sum(diag(tab2))/sum(tab2)))
```

```
## [1] "Decision Tree Accuracy 0.513888888888889"
```

Para este dataset, a Regressão Logística se mostrou com maior acurácia.

De acordo com o experimento que realizamos, podemos afirmar que as variáveis Creatinina Sérica no sangue e o Percentual de sangue saindo do coração por contração são suficientes para prever a chance do paciente de sofrer por insuficiência cardíaca com um grau de precisão de aproximadamente 80%.