Titanic: Machine Learning from Disaster with KNN

Vanderlei Kleinschmidt

21/10/2020

1. Introdução

Quando você acessa o Kaggle, famoso site de competição em ciência de dados, e escolhe o menu "Compete", o site propõe que você comece a sua jornada com o Titanic, uma competição de aprendizado onde você deve construir um modelo de machine learning capaz de prever se um passageiro morreu ou não no naufrágio.

É um ótimo ponto de partida para aprendizado de algorítimos de machine learning, e por isso decidi começar por aqui, usando o algoritmo KNN (k-Nearest Neighbour Classification).

O link para a competição está disponível no: https://www.kaggle.com/c/titanic/overview, juntamente com os dados utilizados.

2. Pacotes utilizados

```
library(tidyverse)
library(Amelia)
library(patchwork)
library(class)
library(gmodels)
```

3. Coletando os dados

Esses dados estão disponíveis no site do Kaggle. O primeiro contém os dados que vou usar para treinar o algoritmo e o segundo os dados de teste, utilizados para validar o resultado do modelo treinado.

```
train <- read.csv('train.csv', stringsAsFactors = F)
test  <- read.csv('test.csv', stringsAsFactors = F)
```

Além desses dois datasets, vou criar mais dois que serão úteis no final.

```
Survived <- train$Survived
survived <- as.factor(survived)
passengers <- test$PassengerId
```

4. Explorando e preparando os dados para análise

Começo dando uma olhada nos datasets, e nos tipos de dados importados.

```
View(train)
str(train)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Flo
rence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily
May Peel)" ...
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

```
View(test)
str(test)
```

```
## 'data.frame':    418 obs. of  11 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Pclass     : int  3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen Needs)" "Myl
es, Mr. Thomas Francis" "Wirz, Mr. Albert" ...
##  $ Sex        : chr  "male" "female" "male" "male" ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : chr  "" "" "" "" ...
##  $ Embarked   : chr  "Q" "S" "Q" "S" ...
```

O dataset de treino tem 891 observações, com 12 variáveis, sendo elas:

* PassengerId: integer => é um índice ou identificador de cada passageiro

* Survived : integer (variável target) => assume valor igual a 0 caso não tenha sobrevivido e valor igual a 1 caso tenha sobrevivido

* Pclass : integer => é uma proxy para o status econômico e social do passageiro, assumindo valor igua a 1 = classe superior; 2 = classe média; e 3 = classe inferior

* Name : character => nome do passageiro * Sex : character => masculino e feminino

* Age : number => idade do passageiro em anos * SibSp : integer => número de irmãos, cuja relação familiar do passageiro no navio é definida como:

** Sibling= brother, sister, stepbrother, stepsister

** Spouse= husband, wife (mistresses and fiancés were ignored)

* Parch : integer => número de pais/crianças abordo, cuja relação familiar do passageiro no navio é definida como:

** Parent= mother, father

** Child= daughter, son, stepdaughter, stepson

** Some children traveled only with a nanny, therefore parch=0 for them.

* Ticket : character => número da passagem

* Fare : number => tarifa do passageiro

* Cabin : character => cabine

* Embarked : character => porto de embarque: C = Cherbourg, Q = Queenstown, S = Southampton

5. Agrupando e analisando os dados

Como temos dois conjuntos de dados, um de treino e outro de teste, antes de iniciar a análise de dados vamos unificar os conjuntos. Com os conjuntos de dados agrupados eu posso entender melhor quais as características dos sobreviventes e quem sabe tirar algum insight.

Porém, depois de unificar eu vou precisar saber quais informações pertencem ao dataset de treino e quais pertencem ao de teste. Por isso eu crio uma nova variável, chamada 'isTrainSet', da seguinte forma:

```
train$isTrainSet <- TRUE
test$isTrainSet <- FALSE

names(train)

##  [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
##  [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
## [11] "Cabin"       "Embarked"    "isTrainSet"

names(test)

##  [1] "PassengerId" "Pclass"      "Name"        "Sex"         "Age"
##  [6] "SibSp"       "Parch"       "Ticket"      "Fare"        "Cabin"
## [11] "Embarked"    "isTrainSet"
```

É preciso também incluir a variável "Survived" nos dados de teste, para que ambos os datasets tenham o mesmo número de colunas, porque no momento, os dados de treino tem 13 colunas e o de teste tem 12. Essa nova variável recebe valor NA, porque no dataset de teste esse valor sairá da previsão obtida pelo algorítmo de machine learning.

```
ncol(train)

## [1] 13

ncol(test)

## [1] 12

test$Survived <- NA
ncol(test)

## [1] 13
```

Se eu juntar os dados de treino e teste em um só dataset, vou perceber que os dados de treino representam cerca de 68% do total do conjunto de dados. Se eu tivesse informações

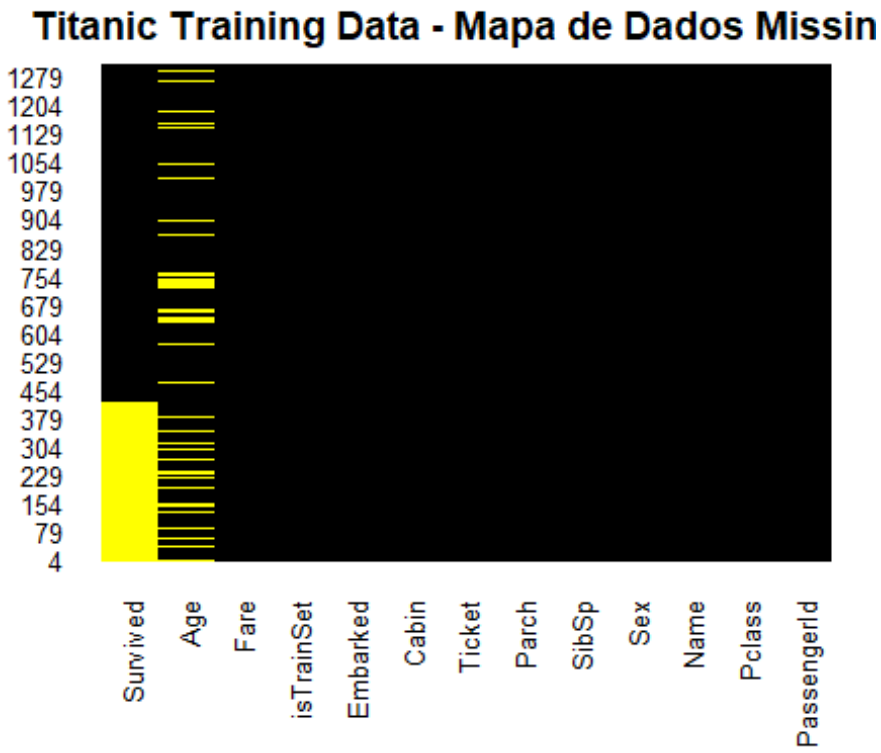
sobre os sobreviventes nos dados de teste eu poderia aumentar esse percentual de dados de treino, para melhorar a acurácia do meu modelo.

Mesmo não tendo essa informação, para analisar melhor as características da amostra, vou juntar os dois conjuntos em um único dataset.

```
fulldata <- rbind(train, test)
round(prop.table(table(fulldata$isTrainSet)), 4)

##
##   FALSE    TRUE
## 0.3193 0.6807
```

Vamos verificar se tem algum valor NA nas demais variáveis, criando um mapa de dados faltantes com o pacote “Amelia”.

```
missmap(fulldata,
        main = "Titanic Training Data - Mapa de Dados Missing",
        col = c("yellow", "black"),
        legend = FALSE)
```



Já era esperado encontrar dados missing na variável Survived, porque juntei os dois conjuntos de dados e nos dados de teste essa variável não existia. Portanto, só tenho o problema de dados missing na variável “Age”.

Vamos ver qual o tamanho desse problema:

```
table(is.na(fulldata$Age))
```

```
##
## FALSE  TRUE
##  1046   263
```

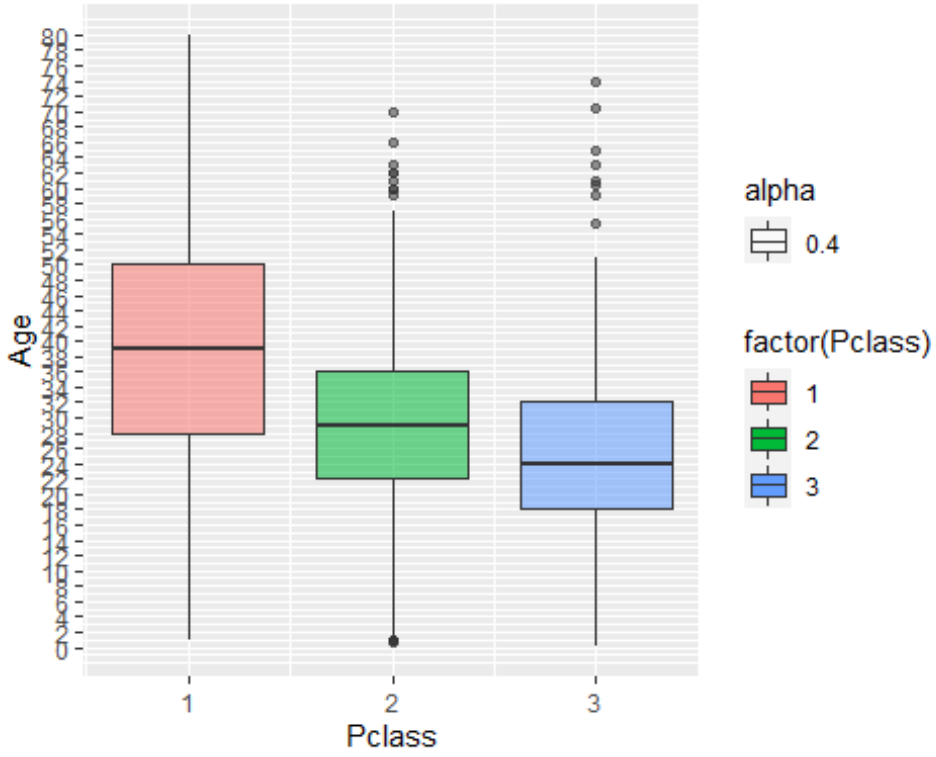
Temos 263 dados faltando (NA) nessa variável, e portanto é um resultado que não pode ser desconsiderado! Para resolver esse problema eu vou substituir os dados missing pela mediana das idades de cada classe de passageiro (1a, 2a e 3a classe). Por que usar a mediana e não a média?

Porque a média é sensível a valores extremos, outliers, e quando você faz um boxplot dos dados vai constatar que há alguns valores extremos, principalmente na segunda e terceira classes, o que pode enviesar o cálculo da média.

```
pl <- ggplot(fulldata, aes(Pclass,Age)) + geom_boxplot(aes(group = Pclass, fill =
factor(Pclass), alpha = 0.4))
```

Vou adicionar uma escala contínua no eixo Y pra melhorar o entendimento do que está sendo visto.

```
pl + scale_y_continuous(breaks = seq(min(0), max(80), by = 2))

## Warning: Removed 263 rows containing non-finite values (stat_boxplot).
```



No boxplot é possível ver que cada classe tem media de idade diferente, sendo que na primeira classe a média de idade é maior do que nas demais, e a terceira classe é a que tem a menor média de idade.

Calculando a mediana de cada classe de passageiros:

```
ageMedian_1 <- fulldata %>% group_by(Pclass) %>% filter(Pclass == 1)
median_1 <- median(ageMedian_1$Age, na.rm = TRUE)

ageMedian_2 <- fulldata %>% group_by(Pclass) %>% filter(Pclass == 2)
```

```
median_2 <- median(ageMedian_2$Age, na.rm = TRUE)

ageMedian_3 <- fulldata %>% group_by(Pclass) %>% filter(Pclass == 3)
median_3 <- median(ageMedian_3$Age, na.rm = TRUE)

rotulos <- c("1a classe", "2a classe", "3a classe")
mediana <- c(median_1, median_2, median_3)
medianas <- rbind(rotulos, mediana)
medianas

##         [,1]        [,2]        [,3]
## rotulos "1a classe" "2a classe" "3a classe"
## mediana "39"        "29"        "24"
```

Vou criar uma função para imputar esses valores (medianas) no dataset:

```
impute_age <- function(age, class){
  out <- age
  for (i in 1:length(age)){

    if (is.na(age[i])){

      if (class[i] == 1){
        out[i] <- median_1

      }else if (class[i] == 2){
        out[i] <- median_2

      }else{
        out[i] <- median_3
      }
    }else{
      out[i]<-age[i]
    }
  }
  return(out)
}

fixed.ages <- impute_age(fulldata$Age, fulldata$Pclass)
fulldata$Age <- fixed.ages
```

Pronto, resolvido o problema de falta de inforação em relação à idade dos passageiros.

```
table(is.na(fulldata$Age))

##
## FALSE
##  1309
```

Vamos dar uma olhada nas demais variáveis pra ver se está tudo ok, antes de começar a análise.

```
table(is.na(fulldata$PassengerId))

##
## FALSE
##  1309
```

A variável Survived tem os valores NA inseridos na parcela oriunda dos dados de teste.

```r
table(is.na(fulldata$Survived))
```

```
##
## FALSE  TRUE
##   891   418
```

```r
table(is.na(fulldata$Pclass))
```

```
##
## FALSE
##  1309
```

```r
table(is.na(fulldata$Name))
```

```
##
## FALSE
##  1309
```

```r
table(is.na(fulldata$Sex))
```

```
##
## FALSE
##  1309
```

```r
table(is.na(fulldata$Age))
```

```
##
## FALSE
##  1309
```

```r
table(is.na(fulldata$SibSp))
```

```
##
## FALSE
##  1309
```

```r
table(is.na(fulldata$Parch))
```

```
##
## FALSE
##  1309
```

```r
table(is.na(fulldata$Ticket))
```

```
##
## FALSE
##  1309
```

Temos um passageiro que aparentemente não pagou a passagem!

```r
table(is.na(fulldata$Fare))
```

```
##
## FALSE  TRUE
##  1308     1
```

```r
table(is.na(fulldata$Cabin))
```

```
##
## FALSE
##  1309
```

```r
table(is.na(fulldata$Embarked))

##
## FALSE
##  1309

table(is.na(fulldata$isTrainSet))

##
## FALSE
##  1309
```

Vamos começar com a variável “Fare”, que apresenta um caso de NA. Vou resolver esse problema imputando a mediana dessa variável. Porém, primeiro eu preciso saber à qual classe pertence o passageiro cuja tarifa não consta no dataset.

Se eu calcular a mediana do dataset completo, sem usar “na.rm = TRUE”, eu obtenho um resultado NA.

```r
median(fulldata$Fare)

## [1] NA
```

Portanto, vou calcular a mediana de cada classe, e aquela que eu não conseguir obter diretamente é a que eu devo utilizar “na.rm = TRUE”.

```r
fareMedian_1 <- fulldata %>% group_by(Pclass) %>% filter(Pclass == 1)
median_f1 <- median(fareMedian_1$Fare)

fareMedian_2 <- fulldata %>% group_by(Pclass) %>% filter(Pclass == 2)
median_f2 <- median(fareMedian_2$Fare)

fareMedian_3 <- fulldata %>% group_by(Pclass) %>% filter(Pclass == 3)
median_f3 <- median(fareMedian_3$Fare)

rotulosf <- c("1a classe", "2a classe", "3a classe")
medianaf <- c(median_f1, median_f2, median_f3)
medianasf <- rbind(rotulosf, medianaf)
medianasf

##          [,1]        [,2]        [,3]
## rotulosf "1a classe" "2a classe" "3a classe"
## medianaf "60"        "15.0458"   NA
```

O passageiro é da terceira classe. Agora posso obter a medianda ignorando o valor NA.

```r
median_f3 <- median(fareMedian_3$Fare, na.rm = TRUE)
median_f3

## [1] 8.05
```

Vou imputar diretamente, dispensando a necessidade de criar uma função.

```r
fulldata[is.na(fulldata$Fare), "Fare"] <- median_f3

table(is.na(fulldata$Fare))

##
## FALSE
##  1309
```

Vamos dar uma última olhada em algumas variáveis usando o “table”, para ter certeza de que não ficou nada de fora.

```
table(fulldata$Survived)

##
##   0   1
## 549 342

table(fulldata$Pclass)

##
##   1   2   3
## 323 277 709

table(fulldata$Sex)

##
## female   male
##    466    843

table(fulldata$Age)

##
## 0.17 0.33 0.42 0.67 0.75 0.83 0.92    1    2    3    4    5    6    7    8    9
##    1    1    1    1    3    3    2   10   12    7   10    5    6    4    6   10
##   10   11 11.5   12   13   14 14.5   15   16   17   18 18.5   19   20 20.5   21
##    4    4    1    3    5    8    2    6   19   20   39    3   29   23    1   41
##   22 22.5   23 23.5   24 24.5   25   26 26.5   27   28 28.5   29   30 30.5   31
##   43    1   26    1  255    1   34   30    1   30   32    3   46   40    2   23
##   32 32.5   33   34 34.5   35   36 36.5   37   38 38.5   39   40 40.5   41   42
##   24    4   21   16    2   23   31    2    9   14    1   59   18    3   11   18
##   43   44   45 45.5   46   47   48   49   50   51   52   53   54   55 55.5   56
##    9   10   21    2    6   14   14    9   15    8    6    4   10    8    1    4
##   57   58   59   60 60.5   61   62   63   64   65   66   67   70 70.5   71   74
##    5    6    3    7    1    5    5    4    5    3    1    1    2    1    2    1
##   76   80
##    1    1

table(fulldata$SibSp)

##
##   0   1   2   3   4   5   8
## 891 319  42  20  22   6   9

table(fulldata$Parch)

##
##    0    1    2    3    4    5    6    9
## 1002  170  113    8    6    6    2    2

table(fulldata$Embarked)

##
##       C   Q   S
##   2 270 123 914

table(fulldata$isTrainSet)
```

```
##
## FALSE  TRUE
##   418   891
```

É importante dar uma olhada na variável "Embarked". São três os portos de origem dos passageiros, C = Cherbourg, Q = Queenstown, S = Southampton. No entanto, o comando "table" revelou que temos dois passageiros que não tem o rótulo do porto de partida.

Por isso, vamos alocá-los para Southampton, que é de onde a grande maioria dos passageiros partiu.

```
fulldata[fulldata$Embarked == "", "Embarked"] <- 'S'
table(fulldata$Embarked)
```

```
##
##   C   Q   S
## 270 123 916
```

O último passo antes de partir efetivamente para a análise, é definir quais variáveis são categóricas. Temos pelo menos 4 variáveis categóricas nesse dataset: Survived, Pclass, Sex e Embarked.

Como a fórmula da distância Euclideana (empregada no knn) não está definida para variáveis nominais, vou criar novas variáveis (dummy) para essas quatro categorias.

```
fulldata[fulldata$Sex == "female", "Female"] <- '1'
fulldata[fulldata$Sex == "male", "Female"] <- '0'
head(select(fulldata, Sex, Female), 10)
```

```
##       Sex Female
## 1    male      0
## 2  female      1
## 3  female      1
## 4  female      1
## 5    male      0
## 6    male      0
## 7    male      0
## 8    male      0
## 9  female      1
## 10 female      1
```

```
fulldata[fulldata$Pclass == "1", "FClasse"] <- '1'
fulldata[fulldata$Pclass == "2", "FClasse"] <- '0'
fulldata[fulldata$Pclass == "3", "FClasse"] <- '0'

fulldata[fulldata$Pclass == "1", "SClasse"] <- '0'
fulldata[fulldata$Pclass == "2", "SClasse"] <- '1'
fulldata[fulldata$Pclass == "3", "SClasse"] <- '0'

fulldata[fulldata$Pclass == "1", "TClasse"] <- '0'
fulldata[fulldata$Pclass == "2", "TClasse"] <- '0'
fulldata[fulldata$Pclass == "3", "TClasse"] <- '1'
head(select(fulldata, Pclass, FClasse, SClasse, TClasse), 10)
```

```
##    Pclass FClasse SClasse TClasse
## 1       3       0       0       1
## 2       1       1       0       0
## 3       3       0       0       1
## 4       1       1       0       0
## 5       3       0       0       1
```

```
## 6       3       0       0       1
## 7       1       1       0       0
## 8       3       0       0       1
## 9       3       0       0       1
## 10      2       0       1       0
```

```
fulldata[fulldata$Embarked == "C", "CEmbarked"] <- '1'
fulldata[fulldata$Embarked == "Q", "CEmbarked"] <- '0'
fulldata[fulldata$Embarked == "S", "CEmbarked"] <- '0'

fulldata[fulldata$Embarked == "C", "QEmbarked"] <- '0'
fulldata[fulldata$Embarked == "Q", "QEmbarked"] <- '1'
fulldata[fulldata$Embarked == "S", "QEmbarked"] <- '0'

fulldata[fulldata$Embarked == "C", "SEmbarked"] <- '0'
fulldata[fulldata$Embarked == "Q", "SEmbarked"] <- '0'
fulldata[fulldata$Embarked == "S", "SEmbarked"] <- '1'
head(select(fulldata, Embarked, CEmbarked, QEmbarked, SEmbarked), 10)
```

```
##    Embarked CEmbarked QEmbarked SEmbarked
## 1         S         0         0         1
## 2         C         1         0         0
## 3         S         0         0         1
## 4         S         0         0         1
## 5         S         0         0         1
## 6         Q         0         1         0
## 7         S         0         0         1
## 8         S         0         0         1
## 9         S         0         0         1
## 10        C         1         0         0
```

A variável Survived já é uma dummy e por isso não preciso fazer nada.

```
head(select(fulldata, Survived), 10)
```

```
##    Survived
## 1         0
## 2         1
## 3         1
## 4         1
## 5         0
## 6         0
## 7         0
## 8         0
## 9         1
## 10        1
```

Agora que criei as novas variáveis dummy, preciso transformar o tipo de dado em categórico, porque atualmente estão como character.

```
str(fulldata)
```

```
## 'data.frame':    1309 obs. of  20 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Flo
```

```
rence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily
May Peel)" ...
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
##  $ isTrainSet : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ Female     : chr  "0" "1" "1" "1" ...
##  $ FClasse    : chr  "0" "1" "0" "1" ...
##  $ SClasse    : chr  "0" "0" "0" "0" ...
##  $ TClasse    : chr  "1" "0" "1" "0" ...
##  $ CEmbarked  : chr  "0" "1" "0" "0" ...
##  $ QEmbarked  : chr  "0" "0" "0" "0" ...
##  $ SEmbarked  : chr  "1" "0" "1" "1" ...
```

Para transformá-las em variáveis categóricas eu uso a seguinte expressão:

```
fulldata$Survived <- as.factor(fulldata$Survived)

fulldata$Sex <- as.factor(fulldata$Sex)
fulldata$Female <-  as.factor(fulldata$Female)

fulldata$Pclass <- as.factor(fulldata$Pclass)
fulldata$FClasse <- as.factor(fulldata$FClasse)
fulldata$SClasse <- as.factor(fulldata$SClasse)
fulldata$TClasse <- as.factor(fulldata$TClasse)

fulldata$Embarked <- as.factor(fulldata$Embarked)
fulldata$CEmbarked <- as.factor(fulldata$CEmbarked)
fulldata$QEmbarked <- as.factor(fulldata$QEmbarked)
fulldata$SEmbarked <- as.factor(fulldata$SEmbarked)

str(fulldata)

## 'data.frame':    1309 obs. of  20 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Flo
rence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily
May Peel)" ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
##  $ isTrainSet : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ Female     : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ FClasse    : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...
##  $ SClasse    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ TClasse    : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 1 2 2 1 ...
##  $ CEmbarked  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 2 ...
```

```
##  $ QEmbarked  : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
##  $ SEmbarked  : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 2 2 2 1 ...
```

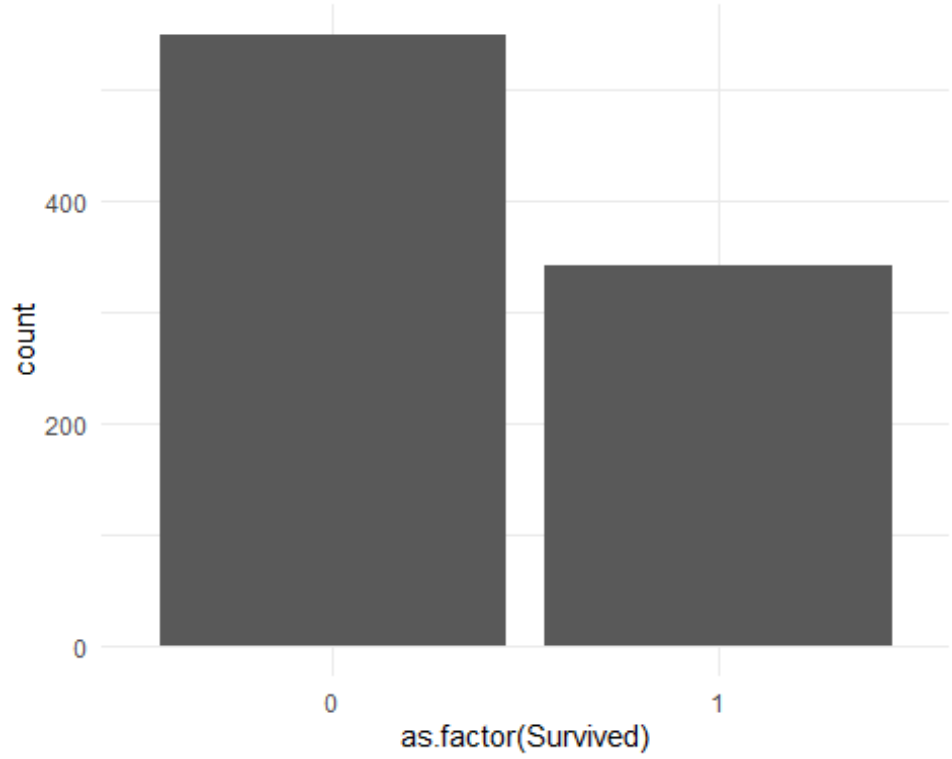
O que eu quero entender agora é, quais são as principais as características dos sobreviventes.

Sabemos que 61,62%% dos passageiros dessa amostra não sobreviveram.

```
sobreviventes <- table(train$Survived) # Utilizo os dados de treino porque não tenho a informação de quantos sobreviveram nos dados de teste.
round(prop.table(sobreviventes), 4)

##
##      0      1
## 0.6162 0.3838
```

Graficamente temos:

```
ggplot(train, aes(x = as.factor(Survived))) + geom_bar() + scale_x_discrete()+ theme_minimal()
```



Novamente eu usei os dados de treino porque a variável “Survived” recebe NA na parte do dataset que contem os dados de teste. O mesmo se repete para alguns gráficos abaixo.

Sabemos também que os homens são maioria na amostra completa, representando 64% dela.

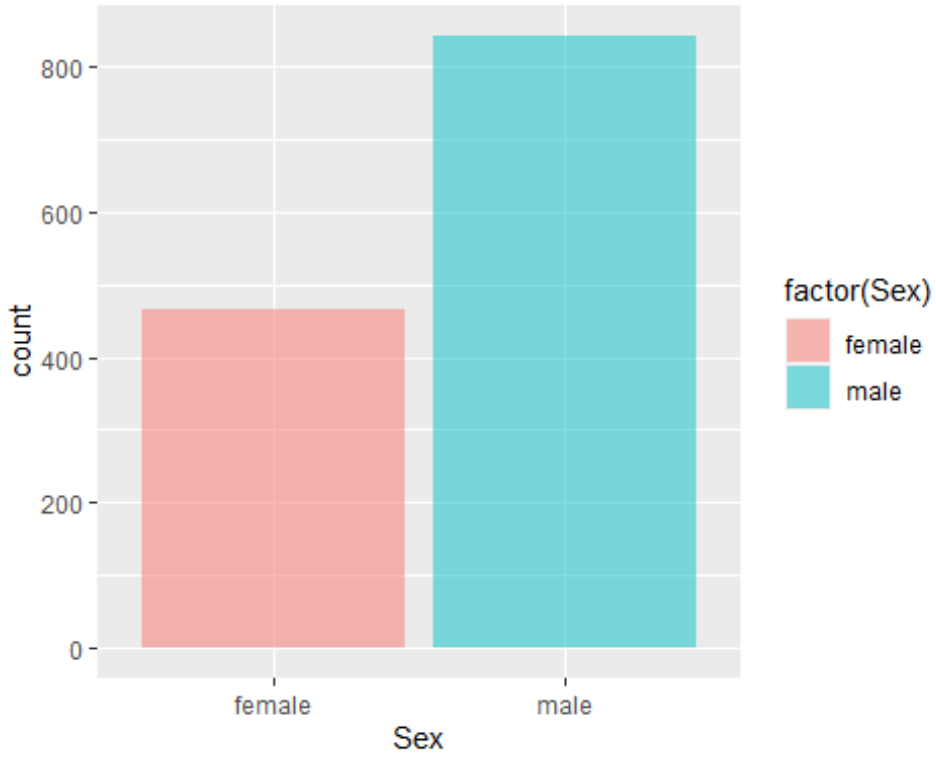
```
gender <- table(fulldata$Sex)
round(prop.table(gender), 4)

##
## female   male
##  0.356  0.644
```

Quando olhamos os dados de treino e teste separadamente vemos que a proporção é parecida.

```
gender_train <- table(train$Sex)
gender_test <- table(test$Sex)
round(prop.table(gender_train), 4)

##
## female   male
## 0.3524 0.6476

round(prop.table(gender_test), 4)

##
## female   male
## 0.3636 0.6364
```

Graficamente:

```
ggplot(fulldata,aes(Sex)) + geom_bar(aes(fill = factor(Sex)), alpha = 0.5)
```



Agrupando por gênero, e pensando na ideia de salvar mulheres e crianças primeiro, percebemos que a proporção de mulheres que morreu em comparação aos homens é bem menor.

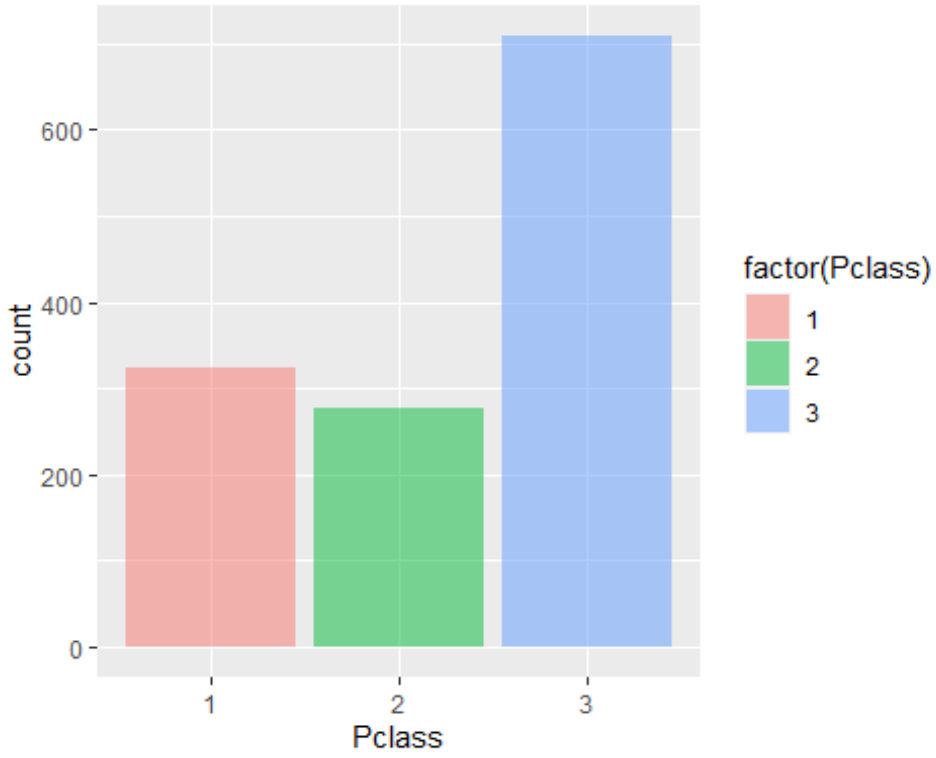
```
bySex <- with(train, table(Survived, Sex)) # Dados de treino porque não sabemos qu
antos sobreviveram nos dados de teste.
bySex

##          Sex
## Survived female male
##        0     81  468
##        1    233  109
```

Como a quantidade de passageiros da terceira classe era maior, é de se esperar também que essa classe deve concentrar o maior número de não sobreviventes.

```
SocEconClass <- table(fulldata$Pclass)
round(prop.table(SocEconClass), 4)

##
##      1      2      3
## 0.2468 0.2116 0.5416
```

Graficamente:

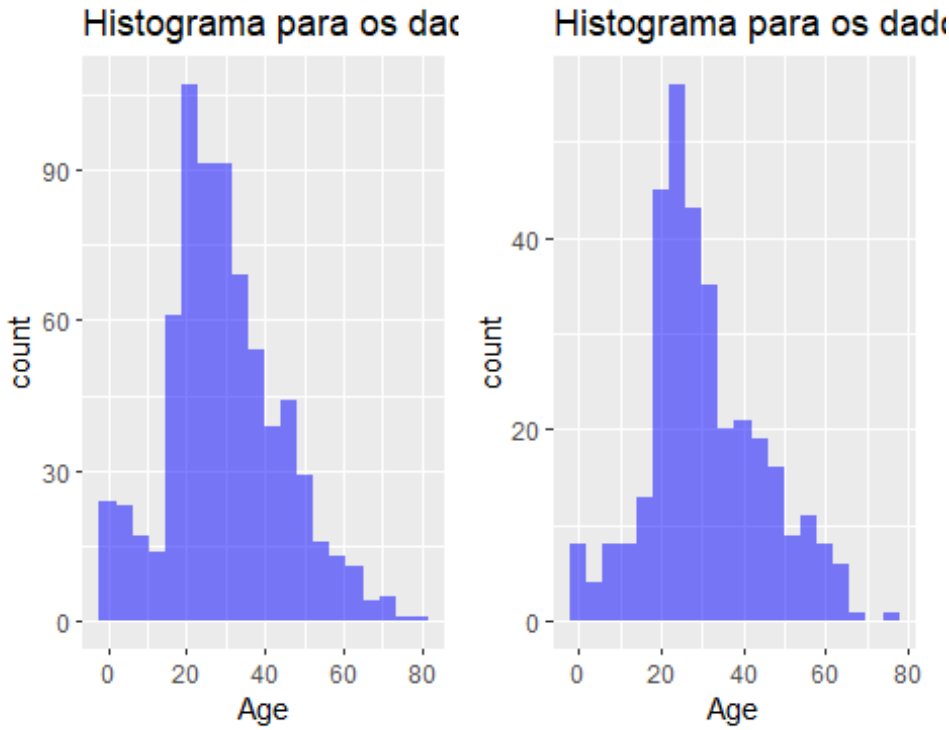
```
ggplot(fulldata,aes(Pclass)) + geom_bar(aes(fill = factor(Pclass)), alpha = 0.5)
```




```
byClass <- with(train, table(Survived, Pclass)) # Utilizo os dados de treino porqu
e não tenho a informação de quantos sobreviveram nos dados de teste.
byClass

##          Pclass
## Survived   1   2   3
##        0  80  97 372
##        1 136  87 119

round(prop.table(byClass), 4)

##          Pclass
## Survived      1      2      3
##        0 0.0898 0.1089 0.4175
##        1 0.1526 0.0976 0.1336
```

A distribuição por idade fica melhor quando olhamos um histograma, onde podemos ver que a tripulação era bem jovem.

```
age_train <- ggplot(train, aes(Age)) + geom_histogram(fill = 'blue', bins = 20, al
pha = 0.5) + ggtitle("Histograma para os dados de treino")
age_test <- ggplot(test,aes(Age)) + geom_histogram(fill = 'blue', bins = 20, alpha
= 0.5) + ggtitle("Histograma para os dados de teste")
age_train + age_test
```



Podemos perceber que existe uma diferença na dispersão das idades entre as amostras de treino e teste em relação à idade.

```
summary(train$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   20.12   28.00   29.70   38.00   80.00     177

summary(test$Age)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.17   21.00   27.00   30.27   39.00   76.00      86

quantile(train$Age, na.rm = TRUE)

##     0%    25%    50%    75%   100%
##  0.420 20.125 28.000 38.000 80.000

quantile(test$Age, na.rm = TRUE)

##    0%   25%   50%   75%  100%
##  0.17 21.00 27.00 39.00 76.00

IQR(train$Age, na.rm = TRUE)

## [1] 17.875

IQR(test$Age, na.rm = TRUE)
```

```
## [1] 18
```

6. Escolhendo o algoritmo de machine learning e treinando o modelo com os dados

Vamos começar a implementação de diversos algoritmos de machine learning, visando encontrar o mais parcimonioso para o presente estudo. Vou selecionar as variáveis que serão utilizadas no treino e no teste.

Todas as alterações feitas no "fulldata" serão replicados para os dados de treino e teste. Porém, o dataset "fulldata" contém 20 colunas enquanto os datasets "train" e "test" tem apenas 13. Isso ocorre por causa da criação das variáveis dummy no "fulldata".

```
ncol(fulldata)

## [1] 20

ncol(train)

## [1] 13

ncol(test)

## [1] 13
```

Para solucionar isso eu crio essas mesmas variáveis nos outros dois datasets, recebendo valor NA

```
train$Female <- NA
test$Female <- NA

train$FClasse <- NA
test$FClasse <- NA
train$SClasse <- NA
test$SClasse <- NA
train$TClasse <- NA
test$TClasse <- NA

train$CEmbarked <- NA
test$CEmbarked <- NA
train$QEmbarked <- NA
test$QEmbarked <- NA
train$SEmbarked <- NA
test$SEmbarked <- NA

ncol(fulldata)

## [1] 20

ncol(train)

## [1] 20

ncol(test)

## [1] 20
```

Agora vamos separar os dados de treino e teste, para rodar o KNN.

```r
train <- fulldata[fulldata$isTrainSet == TRUE,]
test <- fulldata[fulldata$isTrainSet == FALSE,]

str(train)
```

```
## 'data.frame':    891 obs. of  20 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Flo
rence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily
May Peel)" ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
##  $ Age        : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 3 1 3 3 3 2 3 3 3 1 ...
##  $ isTrainSet : logi  TRUE TRUE TRUE TRUE TRUE TRUE ...
##  $ Female     : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ FClasse    : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...
##  $ SClasse    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ TClasse    : Factor w/ 2 levels "0","1": 2 1 2 1 2 2 1 2 2 1 ...
##  $ CEmbarked  : Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 2 ...
##  $ QEmbarked  : Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
##  $ SEmbarked  : Factor w/ 2 levels "0","1": 2 1 2 2 2 1 2 2 2 1 ...
```

```r
str(test)
```

```
## 'data.frame':    418 obs. of  20 variables:
##  $ PassengerId: int  892 893 894 895 896 897 898 899 900 901 ...
##  $ Survived   : Factor w/ 2 levels "0","1": NA NA NA NA NA NA NA NA NA NA ...
##  $ Pclass     : Factor w/ 3 levels "1","2","3": 3 3 2 3 3 3 3 2 3 3 ...
##  $ Name       : chr  "Kelly, Mr. James" "Wilkes, Mrs. James (Ellen Needs)" "Myl
es, Mr. Thomas Francis" "Wirz, Mr. Albert" ...
##  $ Sex        : Factor w/ 2 levels "female","male": 2 1 2 2 1 2 1 2 1 2 ...
##  $ Age        : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp      : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch      : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Ticket     : chr  "330911" "363272" "240276" "315154" ...
##  $ Fare       : num  7.83 7 9.69 8.66 12.29 ...
##  $ Cabin      : chr  "" "" "" "" ...
##  $ Embarked   : Factor w/ 3 levels "C","Q","S": 2 3 2 3 3 3 2 3 1 3 ...
##  $ isTrainSet : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Female     : Factor w/ 2 levels "0","1": 1 2 1 1 2 1 2 1 2 1 ...
##  $ FClasse    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SClasse    : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 2 1 1 ...
##  $ TClasse    : Factor w/ 2 levels "0","1": 2 2 1 2 2 2 2 1 2 2 ...
##  $ CEmbarked  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...
##  $ QEmbarked  : Factor w/ 2 levels "0","1": 2 1 2 1 1 1 2 1 1 1 ...
##  $ SEmbarked  : Factor w/ 2 levels "0","1": 1 2 1 2 2 2 1 2 1 2 ...
```

```r
names(train)
```

```
##  [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
##  [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
## [11] "Cabin"       "Embarked"    "isTrainSet"  "Female"      "FClasse"
## [16] "SClasse"     "TClasse"     "CEmbarked"   "QEmbarked"   "SEmbarked"
```

```
names(test)
```

```
##  [1] "PassengerId" "Survived"    "Pclass"      "Name"        "Sex"
##  [6] "Age"         "SibSp"       "Parch"       "Ticket"      "Fare"
## [11] "Cabin"       "Embarked"    "isTrainSet"  "Female"      "FClasse"
## [16] "SClasse"     "TClasse"     "CEmbarked"   "QEmbarked"   "SEmbarked"
```

Agora eu vou deixar no dataset de treino e de teste apenas as variáveis que serão usadas no algoritimo de machine learning. Ou seja, as variáveis "Sex", "Pclass", "Embarked", "train" "PassengerId", "Name", "Ticket", "Cabin" e "isTrainSet" não fazem sentido em um modelo que pretende predizer se um passageiro sobreviveu ou não ao desastre.

Também tenho que retirar da base de dados pelo menos uma variável dummy por categoria criada, para evitar a "armadilha das variáveis dummy", que consiste em criar uma situação de colinearidade perfeita entre as variáveis.

Na verdade eu poderia ter evitado isso apenas não criando essas variáveis a mais. Decidi fazer mesmo assim, apenas para fins de aprendizado.

```
train <- select(train, -Sex, -Pclass, -TClasse, -Embarked, -SEmbarked, -PassengerI
d, -Name, -Ticket, -Cabin, -isTrainSet)
test <- select(test, -Sex, -Pclass, -TClasse, -Embarked, -SEmbarked, -PassengerId,
-Name, -Ticket, -Cabin, -isTrainSet)

str(train)
```

```
## 'data.frame':    891 obs. of  10 variables:
##  $ Survived : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ Age      : num  22 38 26 35 35 24 54 2 27 14 ...
##  $ SibSp    : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch    : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Fare     : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Female   : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
##  $ FClasse  : Factor w/ 2 levels "0","1": 1 2 1 2 1 1 2 1 1 1 ...
##  $ SClasse  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...
##  $ CEmbarked: Factor w/ 2 levels "0","1": 1 2 1 1 1 1 1 1 1 2 ...
##  $ QEmbarked: Factor w/ 2 levels "0","1": 1 1 1 1 1 2 1 1 1 1 ...
```

```
str(test)
```

```
## 'data.frame':    418 obs. of  10 variables:
##  $ Survived : Factor w/ 2 levels "0","1": NA NA NA NA NA NA NA NA NA NA ...
##  $ Age      : num  34.5 47 62 27 22 14 30 26 18 21 ...
##  $ SibSp    : int  0 1 0 0 1 0 0 1 0 2 ...
##  $ Parch    : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ Fare     : num  7.83 7 9.69 8.66 12.29 ...
##  $ Female   : Factor w/ 2 levels "0","1": 1 2 1 1 2 1 2 1 2 1 ...
##  $ FClasse  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ SClasse  : Factor w/ 2 levels "0","1": 1 1 2 1 1 1 1 2 1 1 ...
##  $ CEmbarked: Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...
##  $ QEmbarked: Factor w/ 2 levels "0","1": 2 1 2 1 1 1 2 1 1 1 ...
```

```
head(train, 10)
```

```
##    Survived Age SibSp Parch    Fare Female FClasse SClasse CEmbarked QEmbarked
## 1         0  22     1     0  7.2500      0       0       0         0         0
## 2         1  38     1     0 71.2833      1       1       0         1         0
## 3         1  26     0     0  7.9250      1       0       0         0         0
## 4         1  35     1     0 53.1000      1       1       0         0         0
```

```
## 5         0  35     0     0  8.0500      0       0       0         0         0
## 6         0  24     0     0  8.4583      0       0       0         0         1
## 7         0  54     0     0 51.8625      0       1       0         0         0
## 8         0   2     3     1 21.0750      0       0       0         0         0
## 9         1  27     0     2 11.1333      1       0       0         0         0
## 10        1  14     1     0 30.0708      1       0       1         1         0
```

```
head(test, 10)
```

```
##     Survived  Age SibSp Parch    Fare Female FClasse SClasse CEmbarked
## 892     <NA> 34.5     0     0  7.8292      0       0       0         0
## 893     <NA> 47.0     1     0  7.0000      1       0       0         0
## 894     <NA> 62.0     0     0  9.6875      0       0       1         0
## 895     <NA> 27.0     0     0  8.6625      0       0       0         0
## 896     <NA> 22.0     1     1 12.2875      1       0       0         0
## 897     <NA> 14.0     0     0  9.2250      0       0       0         0
## 898     <NA> 30.0     0     0  7.6292      1       0       0         0
## 899     <NA> 26.0     1     1 29.0000      0       0       1         0
## 900     <NA> 18.0     0     0  7.2292      1       0       0         1
## 901     <NA> 21.0     2     0 24.1500      0       0       0         0
##     QEmbarked
## 892         1
## 893         0
## 894         1
## 895         0
## 896         0
## 897         0
## 898         1
## 899         0
## 900         0
## 901         0
```

```
names(train)
```

```
##  [1] "Survived"  "Age"       "SibSp"     "Parch"     "Fare"      "Female"
##  [7] "FClasse"   "SClasse"   "CEmbarked" "QEmbarked"
```

```
names(test)
```

```
##  [1] "Survived"  "Age"       "SibSp"     "Parch"     "Fare"      "Female"
##  [7] "FClasse"   "SClasse"   "CEmbarked" "QEmbarked"
```

Feito esses ajustes iniciais, agora podemos escolher o algoritmo a ser empregado. Vou utilizar vários e selecionar para envio ao Kaggle o que tiver o melhor desempenho. O primeiro a ser utilizado é o KNN (K-Nearest Neighbors).

6.1 KNN (K-NEAREST NEIGHBOR)

O KNN é um algoritmo de aprendizagem supervisionada (classificador). O aprendizado se dá por similaridade (proximidade) entre vetores. Tudo começa com dados não classificados, e a partir daí medimos a distância ou a similaridade entre duas instâncias.

Definimos então o valor para o parâmetro "k" a ser utilizado (menores distâncias). A classificação do passageiro entre sobrevivente ou não sobrevivente virá justamente dessa distância, que nesse estudo será calculado como a distância Euclidiana. O valor de "k" não pode ser nem muito alto e nem muito baixo (bias-variance tradeoff), a fim de evitar overfitting e underfitting.

Temos um problema de escala no nosso dataset! Isso ocorre porque temos uma variável "Age", que representa o número de anos (idade), as variáveis "SibSp" e "Parch" que são indicadores de parentesco e a variável "Fare" medido em $. Podemos ver esse problema de escala observando a média e mediana dessas variáveis.

```
summary(train[c("Age", "SibSp", "Parch", "Fare")])
```

```
##       Age            SibSp           Parch             Fare
##  Min.   : 0.42   Min.   :0.000   Min.   :0.0000   Min.   :  0.00
##  1st Qu.:22.00   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:  7.91
##  Median :26.00   Median :0.000   Median :0.0000   Median : 14.45
##  Mean   :29.13   Mean   :0.523   Mean   :0.3816   Mean   : 32.20
##  3rd Qu.:37.00   3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.: 31.00
##  Max.   :80.00   Max.   :8.000   Max.   :6.0000   Max.   :512.33
```

Para resolver esse problema nós vamos criar uma função de normalização:

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

Vamos testar a função nos dados abaixo, para verificar se eles ficam na mesma escala.

```
normalize(c(1,2,3,4,5))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
normalize(c(10,20,30,40,50))
```

```
## [1] 0.00 0.25 0.50 0.75 1.00
```

```
train$Age <- normalize(train$Age)
train$SibSp <- normalize(train$SibSp)
train$Parch <- normalize(train$Parch)
train$Fare <- normalize(train$Fare)
```

Faço o mesmo procedimento para os dados de teste

```
test$Age <- normalize(test$Age)
test$SibSp <- normalize(test$SibSp)
test$Parch <- normalize(test$Parch)
test$Fare <- normalize(test$Fare)
```

Chegou a hora de rodar o KNN, do pacote "class".

Primeiro uma nova olhada nos dados pra ver se está tudo certo.

```
View(train)
View(test)
```

O k receberá valor inicial de 30, que é aproximadamente a raiz quadrada de 891, que é o número de observações da amostra de treino.

```
nrow(train)
```

```
## [1] 891
```

```
sqrt(nrow(train))
```

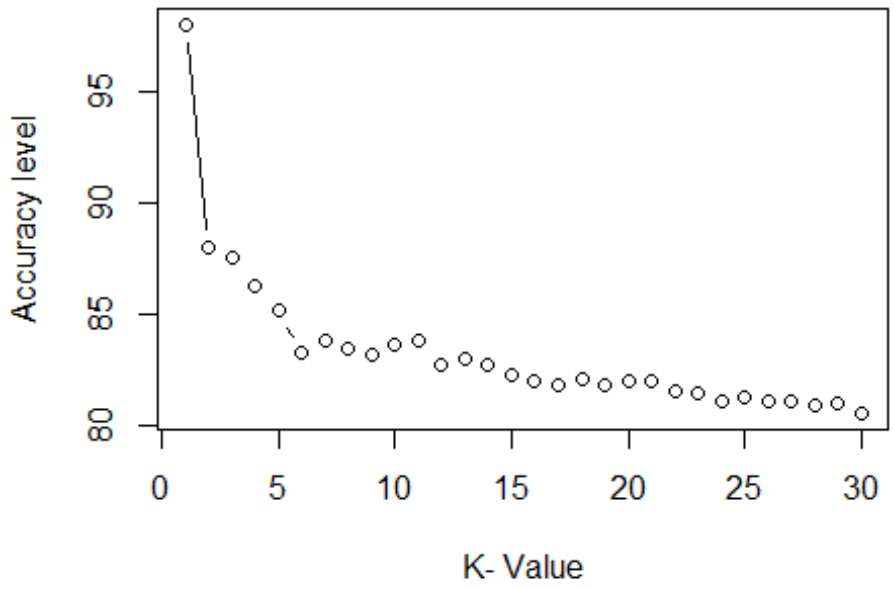
```
## [1] 29.84962

modelo <- knn(train = train[,-1], test = train[,-1], cl = train[,1], k = 30)
```

Vamos tentar identificar o k ótimo para o knn, usando a função abaixo. Quanto menor o "o.optm", melhor a acurácia do modelo.

```
i=1
k.optm=1
for (i in 1:30){
  modelo <- knn(train = train[,-1], test = train[,-1], cl = train[,1], k = i)
  k.optm[i] <- 100 * sum(train[,1] == modelo)/NROW(train[,1])
  k=i
  cat(k,'=',k.optm[i],'')
}
```

```
## 1 = 97.9798 2 = 87.99102 3 = 87.54209 4 = 86.30752 5 = 85.18519 6 = 83.27722 7
= 83.83838 8 = 83.50168 9 = 83.16498 10 = 83.61392 11 = 83.83838 12 = 82.71605 13
= 83.05275 14 = 82.71605 15 = 82.26712 16 = 82.04265 17 = 81.81818 18 = 82.15488 1
9 = 81.81818 20 = 82.04265 21 = 82.04265 22 = 81.59371 23 = 81.48148 24 = 81.14478
25 = 81.25701 26 = 81.14478 27 = 81.14478 28 = 80.92031 29 = 81.03255 30 = 80.5836
1
```

```
plot(k.optm, type="b", xlab="K- Value",ylab="Accuracy level")
```



Tem pelo menos 4 opções de valores para k que parecem melhorar a acurácia, k = 24, 29, 28 e 27, que produzem resultados diferentes em termos de acurácia, medido pelos erros de previsão abaixo.

```
modelo_30 <- knn(train = train[,-1], test = train[,-1], cl = train[,1], k = 30)
modelo_27 <- knn(train = train[,-1], test = train[,-1], cl = train[,1], k = 27)
modelo_28 <- knn(train = train[,-1], test = train[,-1], cl = train[,1], k = 28)
modelo_29 <- knn(train = train[,-1], test = train[,-1], cl = train[,1], k = 29)
modelo_24 <- knn(train = train[,-1], test = train[,-1], cl = train[,1], k = 24)
```

```r
round(prop.table(table(modelo_30))*100, 2)
```

```
## modelo_30
##     0     1
## 73.51 26.49
```

```r
mean(train[,1] != modelo_30)
```

```
## [1] 0.1907969
```

```r
round(prop.table(table(modelo_27))*100, 2)
```

```
## modelo_27
##     0     1
## 73.51 26.49
```

```r
mean(train[,1] != modelo_27)
```

```
## [1] 0.1885522
```

```r
round(prop.table(table(modelo_28))*100, 2)
```

```
## modelo_28
##     0     1
## 73.63 26.37
```

```r
mean(train[,1] != modelo_28)
```

```
## [1] 0.1919192
```

```r
round(prop.table(table(modelo_29))*100, 2)
```

```
## modelo_29
##     0     1
## 73.18 26.82
```

```r
mean(train[,1] != modelo_29)
```

```
## [1] 0.1896745
```

```r
round(prop.table(table(modelo_24))*100, 2)
```

```
## modelo_24
##     0     1
## 73.18 26.82
```

```r
mean(train[,1] != modelo_24)
```

```
## [1] 0.1851852
```

Ussando o CrossTable, podemos ver que temos 94,2% de True Negative, 59,6% de True Positive.

```r
CrossTable(train[,1], modelo_30, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
```

```
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  891
##
##
##              | modelo_30
##   train[, 1] |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       517 |        32 |       549 |
##              |     0.942 |     0.058 |     0.616 |
##              |     0.789 |     0.136 |           |
##              |     0.580 |     0.036 |           |
## -------------|-----------|-----------|-----------|
##            1 |       138 |       204 |       342 |
##              |     0.404 |     0.596 |     0.384 |
##              |     0.211 |     0.864 |           |
##              |     0.155 |     0.229 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       655 |       236 |       891 |
##              |     0.735 |     0.265 |           |
## -------------|-----------|-----------|-----------|
##
##
```

No modelo 27 temos 94,4% de True Negative e 59,9% de True Positive

```
CrossTable(train[,1], modelo_27, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  891
##
##
##              | modelo_27
##   train[, 1] |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       518 |        31 |       549 |
##              |     0.944 |     0.056 |     0.616 |
##              |     0.791 |     0.131 |           |
##              |     0.581 |     0.035 |           |
## -------------|-----------|-----------|-----------|
##            1 |       137 |       205 |       342 |
##              |     0.401 |     0.599 |     0.384 |
##              |     0.209 |     0.869 |           |
##              |     0.154 |     0.230 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       655 |       236 |       891 |
##              |     0.735 |     0.265 |           |
```

```
## -------------|-----------|-----------|-----------|
##
##
```

No modelo 28 temos 94,2% de True Negative e 59,4% de True Positive

```
CrossTable(train[,1], modelo_28, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  891
##
##
##              | modelo_28
##   train[, 1] |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       517 |        32 |       549 |
##              |     0.942 |     0.058 |     0.616 |
##              |     0.788 |     0.136 |           |
##              |     0.580 |     0.036 |           |
## -------------|-----------|-----------|-----------|
##            1 |       139 |       203 |       342 |
##              |     0.406 |     0.594 |     0.384 |
##              |     0.212 |     0.864 |           |
##              |     0.156 |     0.228 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       656 |       235 |       891 |
##              |     0.736 |     0.264 |           |
## -------------|-----------|-----------|-----------|
##
##
```

No modelo 29 temos 94% de True Negative e 60,2% de True Positive

```
CrossTable(train[,1], modelo_29, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  891
##
##
##              | modelo_29
```

```
##   train[, 1] |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       516 |        33 |       549 |
##              |     0.940 |     0.060 |     0.616 |
##              |     0.791 |     0.138 |           |
##              |     0.579 |     0.037 |           |
## -------------|-----------|-----------|-----------|
##            1 |       136 |       206 |       342 |
##              |     0.398 |     0.602 |     0.384 |
##              |     0.209 |     0.862 |           |
##              |     0.153 |     0.231 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       652 |       239 |       891 |
##              |     0.732 |     0.268 |           |
## -------------|-----------|-----------|-----------|
##
##
```

No modelo 24 temos 94,4% de True Negative e 60,8% de True Positive

```
CrossTable(train[,1], modelo_24, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  891
##
##
##              | modelo_24
##   train[, 1] |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       518 |        31 |       549 |
##              |     0.944 |     0.056 |     0.616 |
##              |     0.794 |     0.130 |           |
##              |     0.581 |     0.035 |           |
## -------------|-----------|-----------|-----------|
##            1 |       134 |       208 |       342 |
##              |     0.392 |     0.608 |     0.384 |
##              |     0.206 |     0.870 |           |
##              |     0.150 |     0.233 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       652 |       239 |       891 |
##              |     0.732 |     0.268 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Em resumo, o modelo que apresenta melhores resultados tem k = 24.

```
modelo <- knn(train = train[,-1], test = test[,-1], cl = train[,1], k = 24)
```

Agora é só gravar em um arquivo csv e enviar para o Kaggle.

```
submission <- data.frame(PassengerId = passengers, Survived = modelo)
write.csv(submission,'titanic_knn.csv', row.names = FALSE)
```

Após a postagem consegui obter 77,75% de acertos, ou seja, um resultado razoável mas que me deixa na posição 5.683. Vou continuar usando o R com outros algoritmos e dessa forma tentar melhorar a minha performance.