Avaliação

Fernando e Leonardo

01/10/2019

# Introdução

Esse trabalho analisa um dataset de mãos de Poker que consistem de 5 cartas. O objetivo é analisar as 5 cartas e estimar qual é a mão para aquele jogo.

# Naïve Bayes

No ramo de Machine Learning, Naïve Bayes é um grupo de classificadores probabilísticos simples (Wikipedia, 2019). Esses classificadores possuem a característica comum de se basear no teorema de Bayes, que calcula a probabilidade de um evento dado que outro evento já ocorreu (Gonçalves, 2019). Esses algoritmos não assumem dependência entre as variáveis e, por isso eles são considerados ingênuos (naive).

Devido à simplicidade do algoritmo ele é mais rápido que outros classificadores e costuma ser utilizado para classificar textos através da frequência das palavras utilizadas (Becker, 2019).

# Estrutura do Dataset

O dataset PokerHands possui as seguintes colunas:

1. S1 - Naipe da carta #1: Ordinal (1-4) representando o naipe (Copas, Espadas, Ouros, Paus)
2. C1 - Valor da carta #1: Numérico (1-13) representando o valor ou número da carta (Ás, 2, 3, …, 10, Valete, Rainha, Rei)
3. S2 - Naipe da carta #2: Ordinal (1-4) representando o naipe
4. C2 - Valor da carta #2: Numérico (1-13) representando o valor
5. S3 - Naipe da carta #3: Ordinal (1-4) representando o naipe
6. C3 - Valor da carta #3: Numérico (1-13) representando o valor
7. S4 - Naipe da carta #4: Ordinal (1-4) representando o naipe
8. C4 - Valor da carta #4: Numérico (1-13) representando o valor
9. S5 - Naipe da carta #5: Ordinal (1-4) representando o naipe
10. C5 - Valor da carta #5: Numérico (1-13) representando o valor
11. CLASS - Classificação: Ordinal (0-9) representando a classe que essa mão representa

# Tipos de mãos do Poker

As mãos do Poker são classificadas em 9 tipos, conforme a tabela a seguir. O jogador que possui a maior “mão” vencerá. A tabela está classificada conforme as variáveis do dataset.

|  |  |  |
| --- | --- | --- |
| Classe | Nome | Descrição |
| 0 | Carta mais alta | Nenhuma mão de Poker, vence quem possuir a carta mais alta |
| 1 | Um par | Duas cartas de mesmo valor |
| 2 | Dois pares | Dois valores se repetem entre as 5 cartas |
| 3 | Trinca | Três cartas de valores iguais |
| 4 | Sequência | 5 cartas em sequência, sem interrupção |
| 5 | Flush | 5 cartas de mesmo naipe |
| 6 | Full House | Uma trinca e um par na mesma mão |
| 7 | Quadra | 4 cartas de mesmo valor |
| 8 | Straight Flush | 5 cartas em sequência e do mesmo naipe, sem lacunas |
| 9 | Royal Flush | Sequência Dez, Valete, Dama, Rei e Ás, do mesmo naipe |

# Bibliotecas utilizadas

library(readr) # Para carregamento do arquivo  
library(sqldf) # Para executar SQLs sobre os DataSets

## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite

library(e1071) # Contém o algoritmo de Naïve Bayes  
library(tidyr) # Para transformar colunas em observações  
library(dplyr) # Para operações de seleção, agrupamento

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(ggplot2)  
library(RColorBrewer)  
library(data.table) # Para incluir sequencial em grupos

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

library(knitr)  
library(kableExtra)

##   
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':  
##   
## group\_rows

library(pander) # Para

# Carregando o arquivo

Leitura dos arquivos. Os arquivos não possuem cabeçalho.

pokerTreino <- read\_csv("datasets/poker-hand-training-true.data", col\_names = c ("S1","C1","S2","C2","S3","C3","S4","C4","S5","C5","CLASS"))

## Parsed with column specification:  
## cols(  
## S1 = col\_double(),  
## C1 = col\_double(),  
## S2 = col\_double(),  
## C2 = col\_double(),  
## S3 = col\_double(),  
## C3 = col\_double(),  
## S4 = col\_double(),  
## C4 = col\_double(),  
## S5 = col\_double(),  
## C5 = col\_double(),  
## CLASS = col\_double()  
## )

pokerTeste <- read\_csv("datasets/poker-hand-testing.data", col\_names = c ("S1","C1","S2","C2","S3","C3","S4","C4","S5","C5","CLASS"))

## Parsed with column specification:  
## cols(  
## S1 = col\_double(),  
## C1 = col\_double(),  
## S2 = col\_double(),  
## C2 = col\_double(),  
## S3 = col\_double(),  
## C3 = col\_double(),  
## S4 = col\_double(),  
## C4 = col\_double(),  
## S5 = col\_double(),  
## C5 = col\_double(),  
## CLASS = col\_double()  
## )

## Ajustes

A classe será transforamda em um fator ordenado.

pokerTreino$S1 <- factor(pokerTreino$S1, levels=c(1:4), ordered = TRUE)  
pokerTreino$C1 <- factor(pokerTreino$C1, levels=c(1:13), ordered = TRUE)  
pokerTreino$S2 <- factor(pokerTreino$S2, levels=c(1:4), ordered = TRUE)  
pokerTreino$C2 <- factor(pokerTreino$C2, levels=c(1:13), ordered = TRUE)  
pokerTreino$S3 <- factor(pokerTreino$S3, levels=c(1:4), ordered = TRUE)  
pokerTreino$C3 <- factor(pokerTreino$C3, levels=c(1:13), ordered = TRUE)  
pokerTreino$S4 <- factor(pokerTreino$S4, levels=c(1:4), ordered = TRUE)  
pokerTreino$C4 <- factor(pokerTreino$C4, levels=c(1:13), ordered = TRUE)  
pokerTreino$S5 <- factor(pokerTreino$S5, levels=c(1:4), ordered = TRUE)  
pokerTreino$C5 <- factor(pokerTreino$C5, levels=c(1:13), ordered = TRUE)  
pokerTreino$CLASS <- factor(pokerTreino$CLASS, levels=c(0:9), ordered =TRUE)  
  
pokerTeste$S1 <- factor(pokerTeste$S1, levels=c(1:4), ordered = TRUE)  
pokerTeste$C1 <- factor(pokerTeste$C1, levels=c(1:13), ordered = TRUE)  
pokerTeste$S2 <- factor(pokerTeste$S2, levels=c(1:4), ordered = TRUE)  
pokerTeste$C2 <- factor(pokerTeste$C2, levels=c(1:13), ordered = TRUE)  
pokerTeste$S3 <- factor(pokerTeste$S3, levels=c(1:4), ordered = TRUE)  
pokerTeste$C3 <- factor(pokerTeste$C3, levels=c(1:13), ordered = TRUE)  
pokerTeste$S4 <- factor(pokerTeste$S4, levels=c(1:4), ordered = TRUE)  
pokerTeste$C4 <- factor(pokerTeste$C4, levels=c(1:13), ordered = TRUE)  
pokerTeste$S5 <- factor(pokerTeste$S5, levels=c(1:4), ordered = TRUE)  
pokerTeste$C5 <- factor(pokerTeste$C5, levels=c(1:13), ordered = TRUE)  
pokerTeste$CLASS <- factor(pokerTeste$CLASS, levels=c(0:9), ordered =TRUE)

## Cabeçalho

pander(head(pokerTreino))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | C1 | S2 | C2 | S3 | C3 | S4 | C4 | S5 | C5 | CLASS |
| 1 | 10 | 1 | 11 | 1 | 13 | 1 | 12 | 1 | 1 | 9 |
| 2 | 11 | 2 | 13 | 2 | 10 | 2 | 12 | 2 | 1 | 9 |
| 3 | 12 | 3 | 11 | 3 | 13 | 3 | 10 | 3 | 1 | 9 |
| 4 | 10 | 4 | 11 | 4 | 1 | 4 | 13 | 4 | 12 | 9 |
| 4 | 1 | 4 | 13 | 4 | 12 | 4 | 11 | 4 | 10 | 9 |
| 1 | 2 | 1 | 4 | 1 | 5 | 1 | 3 | 1 | 6 | 8 |

pander(head(pokerTeste))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | C1 | S2 | C2 | S3 | C3 | S4 | C4 | S5 | C5 | CLASS |
| 1 | 1 | 1 | 13 | 2 | 4 | 2 | 3 | 1 | 12 | 0 |
| 3 | 12 | 3 | 2 | 3 | 11 | 4 | 5 | 2 | 5 | 1 |
| 1 | 9 | 4 | 6 | 1 | 4 | 3 | 2 | 3 | 9 | 1 |
| 1 | 4 | 3 | 13 | 2 | 13 | 2 | 1 | 3 | 6 | 1 |
| 3 | 10 | 2 | 7 | 1 | 2 | 2 | 11 | 4 | 9 | 0 |
| 1 | 3 | 4 | 5 | 3 | 4 | 1 | 12 | 4 | 6 | 0 |

# Análise do arquivo

Estrutura do arquivo:

str(pokerTreino)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 25010 obs. of 11 variables:  
## $ S1 : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 2 3 4 4 1 1 2 3 4 ...  
## $ C1 : Ord.factor w/ 13 levels "1"<"2"<"3"<"4"<..: 10 11 12 10 1 2 9 1 5 1 ...  
## $ S2 : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 2 3 4 4 1 1 2 3 4 ...  
## $ C2 : Ord.factor w/ 13 levels "1"<"2"<"3"<"4"<..: 11 13 11 11 13 4 12 2 6 4 ...  
## $ S3 : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 2 3 4 4 1 1 2 3 4 ...  
## $ C3 : Ord.factor w/ 13 levels "1"<"2"<"3"<"4"<..: 13 10 13 1 12 5 10 3 9 2 ...  
## $ S4 : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 2 3 4 4 1 1 2 3 4 ...  
## $ C4 : Ord.factor w/ 13 levels "1"<"2"<"3"<"4"<..: 12 12 10 13 11 3 11 4 7 3 ...  
## $ S5 : Ord.factor w/ 4 levels "1"<"2"<"3"<"4": 1 2 3 4 4 1 1 2 3 4 ...  
## $ C5 : Ord.factor w/ 13 levels "1"<"2"<"3"<"4"<..: 1 1 1 12 10 6 13 5 8 5 ...  
## $ CLASS: Ord.factor w/ 10 levels "0"<"1"<"2"<"3"<..: 10 10 10 10 10 9 9 9 9 9 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. S1 = col\_double(),  
## .. C1 = col\_double(),  
## .. S2 = col\_double(),  
## .. C2 = col\_double(),  
## .. S3 = col\_double(),  
## .. C3 = col\_double(),  
## .. S4 = col\_double(),  
## .. C4 = col\_double(),  
## .. S5 = col\_double(),  
## .. C5 = col\_double(),  
## .. CLASS = col\_double()  
## .. )

Sumário do arquivo

summary(pokerTreino)

## S1 C1 S2 C2 S3   
## 1:6150 1 : 1982 1:6309 13 : 2007 1:6230   
## 2:6298 9 : 1967 2:6244 1 : 1985 2:6208   
## 3:6250 7 : 1961 3:6157 6 : 1956 3:6153   
## 4:6312 8 : 1948 4:6300 12 : 1939 4:6419   
## 2 : 1941 11 : 1931   
## 12 : 1940 10 : 1922   
## (Other):13271 (Other):13270   
## C3 S4 C4 S5 C5   
## 10 : 2000 1:6269 3 : 1999 1:6308 8 : 1994   
## 12 : 1987 2:6248 2 : 1987 2:6172 1 : 1970   
## 3 : 1965 3:6314 1 : 1983 3:6314 5 : 1963   
## 4 : 1935 4:6179 11 : 1953 4:6216 2 : 1957   
## 7 : 1927 9 : 1946 10 : 1948   
## 5 : 1920 7 : 1943 4 : 1946   
## (Other):13276 (Other):13199 (Other):13232   
## CLASS   
## 0 :12493   
## 1 :10599   
## 2 : 1206   
## 3 : 513   
## 4 : 93   
## 5 : 54   
## (Other): 52

# Análise exploratória

Total de linhas nos datasets

paste("Linhas no dataset de treino:", nrow(pokerTreino))

## [1] "Linhas no dataset de treino: 25010"

paste("Linhas no dataset de teste:", nrow(pokerTeste))

## [1] "Linhas no dataset de teste: 1000000"

Visualização do início do DataSet

pander(head(pokerTreino))

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S1 | C1 | S2 | C2 | S3 | C3 | S4 | C4 | S5 | C5 | CLASS |
| 1 | 10 | 1 | 11 | 1 | 13 | 1 | 12 | 1 | 1 | 9 |
| 2 | 11 | 2 | 13 | 2 | 10 | 2 | 12 | 2 | 1 | 9 |
| 3 | 12 | 3 | 11 | 3 | 13 | 3 | 10 | 3 | 1 | 9 |
| 4 | 10 | 4 | 11 | 4 | 1 | 4 | 13 | 4 | 12 | 9 |
| 4 | 1 | 4 | 13 | 4 | 12 | 4 | 11 | 4 | 10 | 9 |
| 1 | 2 | 1 | 4 | 1 | 5 | 1 | 3 | 1 | 6 | 8 |

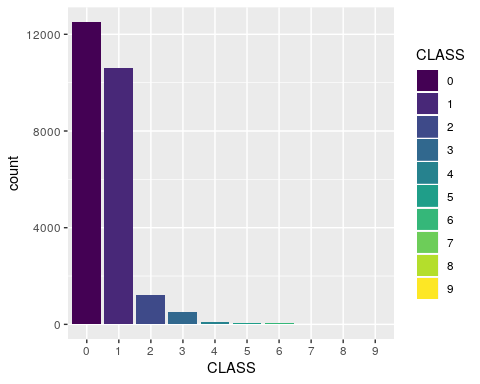
Divisão dos registros no dataset

pander(sqldf('SELECT COUNT(1) AS [Quantidade],   
 class,   
 (CASE WHEN Class = 0 THEN "Nothing"  
 WHEN Class = 1 THEN "One pair"  
 WHEN Class = 2 THEN "Two pairs"  
 WHEN Class = 3 THEN "Three of a kind"  
 WHEN Class = 4 THEN "Straight"  
 WHEN Class = 5 THEN "Flush"  
 WHEN Class = 6 THEN "Full house"  
 WHEN Class = 7 THEN "Four of a kind"  
 WHEN Class = 8 THEN "Straight flush"  
 WHEN Class = 9 THEN "Royal flush"   
 END) AS [TipoENG],  
 ROUND(((CAST (COUNT(1) AS real)/(SELECT COUNT(1) FROM pokerTreino)) \* 100),2) AS [%]  
 FROM pokerTreino   
 GROUP BY class   
 ORDER BY [Quantidade] DESC'))

|  |  |  |  |
| --- | --- | --- | --- |
| Quantidade | CLASS | TipoENG | % |
| 12493 | 0 | Nothing | 49.95 |
| 10599 | 1 | One pair | 42.38 |
| 1206 | 2 | Two pairs | 4.82 |
| 513 | 3 | Three of a kind | 2.05 |
| 93 | 4 | Straight | 0.37 |
| 54 | 5 | Flush | 0.22 |
| 36 | 6 | Full house | 0.14 |
| 6 | 7 | Four of a kind | 0.02 |
| 5 | 8 | Straight flush | 0.02 |
| 5 | 9 | Royal flush | 0.02 |

Distribuição das classes

ggplot(data = pokerTreino, aes(CLASS)) + geom\_bar(aes(fill=CLASS))



Unica combinação que se repetiu

pander(sqldf('SELECT DISTINCT   
 COUNT(1) AS [Quantidade],   
 class,   
 (CASE WHEN Class = 0 THEN "Nothing"  
 WHEN Class = 1 THEN "One pair"  
 WHEN Class = 2 THEN "Two pairs"  
 WHEN Class = 3 THEN "Three of a kind"  
 WHEN Class = 4 THEN "Straight"  
 WHEN Class = 5 THEN "Flush"  
 WHEN Class = 6 THEN "Full house"  
 WHEN Class = 7 THEN "Four of a kind"  
 WHEN Class = 8 THEN "Straight flush"  
 WHEN Class = 9 THEN "Royal flush"   
 END) AS [TipoENG],  
 S1,  
 C1,  
 S2,  
 C2,  
 S3,  
 C3,  
 S4,  
 C4,  
 S5,  
 C5  
 FROM pokerTreino   
 GROUP BY class, [TipoENG], S1, C1, S2, C2, S3, C3, S4, C4, S5, C5   
 HAVING COUNT(1) > 1  
 ORDER BY S1, C1, S2, C2, S3, C3, S4, C4, S5, C5 DESC'))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Quantidade | CLASS | TipoENG | S1 | C1 | S2 | C2 | S3 | C3 | S4 | C4 | S5 | C5 |
| 2 | 1 | One pair | 3 | 12 | 3 | 2 | 2 | 8 | 1 | 2 | 2 | 3 |
| 2 | 1 | One pair | 4 | 2 | 2 | 5 | 4 | 9 | 2 | 7 | 2 | 2 |

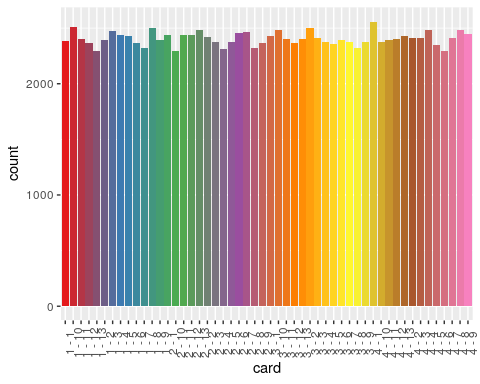
Quantidade de linhas onde as combinações são unicas: 25006 de 25010

pander(head(sqldf('SELECT DISTINCT   
 COUNT(1) AS [Quantidade],   
 class,   
 (CASE WHEN Class = 0 THEN "Nothing"  
 WHEN Class = 1 THEN "One pair"  
 WHEN Class = 2 THEN "Two pairs"  
 WHEN Class = 3 THEN "Three of a kind"  
 WHEN Class = 4 THEN "Straight"  
 WHEN Class = 5 THEN "Flush"  
 WHEN Class = 6 THEN "Full house"  
 WHEN Class = 7 THEN "Four of a kind"  
 WHEN Class = 8 THEN "Straight flush"  
 WHEN Class = 9 THEN "Royal flush"   
 END) AS [TipoENG],  
 S1,  
 C1,  
 S2,  
 C2,  
 S3,  
 C3,  
 S4,  
 C4,  
 S5,  
 C5  
 FROM pokerTreino   
 GROUP BY class, [TipoENG], S1, C1, S2, C2, S3, C3, S4, C4, S5, C5   
 HAVING COUNT(1) = 1  
 ORDER BY S1, C1, S2, C2, S3, C3, S4, C4, S5, C5 DESC')))

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Quantidade | CLASS | TipoENG | S1 | C1 | S2 | C2 | S3 | C3 | S4 | C4 | S5 | C5 |
| 1 | 5 | Flush | 1 | 1 | 1 | 10 | 1 | 12 | 1 | 2 | 1 | 4 |
| 1 | 0 | Nothing | 1 | 1 | 1 | 10 | 1 | 4 | 2 | 2 | 1 | 9 |
| 1 | 0 | Nothing | 1 | 1 | 1 | 10 | 1 | 4 | 3 | 9 | 3 | 6 |
| 1 | 1 | One pair | 1 | 1 | 1 | 10 | 1 | 7 | 4 | 6 | 4 | 10 |
| 1 | 0 | Nothing | 1 | 1 | 1 | 10 | 1 | 9 | 2 | 3 | 1 | 11 |
| 1 | 3 | Three of a kind | 1 | 1 | 1 | 10 | 2 | 1 | 2 | 2 | 4 | 1 |

Distribuição das cartas

dataset <- data.frame(pokerTreino)  
dataset$id = seq.int(nrow(dataset))  
ds1 <- select(dataset,id,S1,S2,S3,S4,S5) %>% gather(key,suit,S1:S5)  
ds2 <- select(dataset,id,C1,C2,C3,C4,C5) %>% gather(key,rank,C1:C5)  
ds1$key <- sub("S","",ds1$key)  
ds2$key <- sub("C","",ds2$key)  
cartasTreino <- merge(ds1, ds2, by.x = c("id","key"), by.y = c("id","key"))  
rm(dataset)  
rm(ds1)  
rm(ds2)  
cartas <- data.frame(card = paste(cartasTreino$suit,"-",cartasTreino$rank))  
color.pallete <- colorRampPalette(brewer.pal(8, "Set1"))(52)  
ggplot(data = cartas, aes(x = card)) + geom\_bar(aes(fill=card)) + theme(axis.text.x = element\_text(angle = 90), legend.position = "none") + scale\_fill\_manual(values=color.pallete)



rm(cartas)  
rm(color.pallete)

Quantidade de cartas distintas:

pander(sqldf('SELECT DISTINCT  
 suit,  
 rank,  
 COUNT(1) AS [Quantidade],  
 ROUND(((CAST (COUNT(1) AS real)/(SELECT COUNT(1) FROM cartasTreino)) \* 100),2) AS [%]  
 FROM cartasTreino   
 GROUP BY suit, rank  
 ORDER BY [Quantidade] DESC'))

|  |  |  |  |
| --- | --- | --- | --- |
| suit | rank | Quantidade | % |
| 4 | 1 | 2559 | 2.05 |
| 1 | 10 | 2508 | 2.01 |
| 3 | 2 | 2501 | 2 |
| 1 | 8 | 2499 | 2 |
| 3 | 10 | 2487 | 1.99 |
| 2 | 13 | 2482 | 1.98 |
| 4 | 8 | 2480 | 1.98 |
| 4 | 4 | 2479 | 1.98 |
| 1 | 3 | 2477 | 1.98 |
| 2 | 7 | 2468 | 1.97 |
| 2 | 6 | 2459 | 1.97 |
| 4 | 9 | 2443 | 1.95 |
| 1 | 4 | 2437 | 1.95 |
| 2 | 1 | 2435 | 1.95 |
| 2 | 12 | 2435 | 1.95 |
| 2 | 11 | 2434 | 1.95 |
| 3 | 1 | 2433 | 1.95 |
| 1 | 5 | 2429 | 1.94 |
| 4 | 13 | 2425 | 1.94 |
| 2 | 2 | 2418 | 1.93 |
| 4 | 3 | 2414 | 1.93 |
| 4 | 7 | 2413 | 1.93 |
| 3 | 3 | 2410 | 1.93 |
| 4 | 2 | 2407 | 1.92 |
| 1 | 11 | 2406 | 1.92 |
| 4 | 12 | 2401 | 1.92 |
| 3 | 13 | 2400 | 1.92 |
| 3 | 11 | 2398 | 1.92 |
| 1 | 9 | 2394 | 1.91 |
| 3 | 6 | 2392 | 1.91 |
| 1 | 2 | 2390 | 1.91 |
| 4 | 11 | 2390 | 1.91 |
| 1 | 1 | 2388 | 1.91 |
| 4 | 10 | 2379 | 1.9 |
| 3 | 4 | 2375 | 1.9 |
| 3 | 7 | 2375 | 1.9 |
| 2 | 5 | 2373 | 1.9 |
| 3 | 9 | 2372 | 1.9 |
| 2 | 3 | 2371 | 1.9 |
| 3 | 12 | 2369 | 1.89 |
| 1 | 6 | 2365 | 1.89 |
| 2 | 9 | 2364 | 1.89 |
| 1 | 12 | 2362 | 1.89 |
| 3 | 5 | 2356 | 1.88 |
| 4 | 5 | 2345 | 1.88 |
| 1 | 7 | 2320 | 1.86 |
| 3 | 8 | 2320 | 1.86 |
| 2 | 8 | 2318 | 1.85 |
| 2 | 4 | 2316 | 1.85 |
| 2 | 10 | 2297 | 1.84 |
| 1 | 13 | 2291 | 1.83 |
| 4 | 6 | 2291 | 1.83 |

Quantidade de números de carta distintos:

pander(sqldf('SELECT DISTINCT  
 rank,  
 suit,  
 COUNT(1) AS [Quantidade],  
 ROUND(((CAST (COUNT(1) AS real)/(SELECT COUNT(1) FROM cartasTreino)) \* 100),2) AS [%]  
 FROM cartasTreino   
 GROUP BY rank  
 ORDER BY [Quantidade] DESC'))

|  |  |  |  |
| --- | --- | --- | --- |
| rank | suit | Quantidade | % |
| 1 | 4 | 9815 | 7.85 |
| 2 | 2 | 9716 | 7.77 |
| 3 | 1 | 9672 | 7.73 |
| 10 | 3 | 9671 | 7.73 |
| 11 | 2 | 9628 | 7.7 |
| 8 | 1 | 9617 | 7.69 |
| 4 | 2 | 9607 | 7.68 |
| 13 | 1 | 9598 | 7.68 |
| 7 | 4 | 9576 | 7.66 |
| 9 | 4 | 9573 | 7.66 |
| 12 | 3 | 9567 | 7.65 |
| 6 | 3 | 9507 | 7.6 |
| 5 | 1 | 9503 | 7.6 |

rm(cartasTreino)

# Treinamento

Treinar com o Naïve Bayes

pokerTreino <- data.frame(pokerTreino)  
nv <- naiveBayes(pokerTreino[,c("S1","C1","S2","C2","S3","C3","S4","C4","S5","C5")],pokerTreino[,c("CLASS")])

Como o algoritmo estruturou seus parâmetros:

str(nv)

## List of 5  
## $ apriori : 'table' int [1:10(1d)] 12493 10599 1206 513 93 54 36 6 5 5  
## ..- attr(\*, "dimnames")=List of 1  
## .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## $ tables :List of 10  
## ..$ S1: 'table' num [1:10, 1:4] 0.247 0.248 0.217 0.234 0.301 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ S1 : chr [1:4] "1" "2" "3" "4"  
## ..$ C1: 'table' num [1:10, 1:13] 0.0793 0.0793 0.0688 0.1014 0.0538 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ C1 : chr [1:13] "1" "2" "3" "4" ...  
## ..$ S2: 'table' num [1:10, 1:4] 0.253 0.252 0.256 0.25 0.237 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ S2 : chr [1:4] "1" "2" "3" "4"  
## ..$ C2: 'table' num [1:10, 1:13] 0.0778 0.0814 0.0779 0.0936 0.0215 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ C2 : chr [1:13] "1" "2" "3" "4" ...  
## ..$ S3: 'table' num [1:10, 1:4] 0.249 0.25 0.248 0.244 0.237 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ S3 : chr [1:4] "1" "2" "3" "4"  
## ..$ C3: 'table' num [1:10, 1:13] 0.073 0.0784 0.0821 0.0877 0.043 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ C3 : chr [1:13] "1" "2" "3" "4" ...  
## ..$ S4: 'table' num [1:10, 1:4] 0.253 0.249 0.235 0.248 0.323 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ S4 : chr [1:4] "1" "2" "3" "4"  
## ..$ C4: 'table' num [1:10, 1:13] 0.0792 0.0791 0.0813 0.0838 0.0645 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ C4 : chr [1:13] "1" "2" "3" "4" ...  
## ..$ S5: 'table' num [1:10, 1:4] 0.249 0.254 0.265 0.259 0.226 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ S5 : chr [1:4] "1" "2" "3" "4"  
## ..$ C5: 'table' num [1:10, 1:13] 0.0788 0.0772 0.0862 0.0994 0.0215 ...  
## .. ..- attr(\*, "dimnames")=List of 2  
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2" "3" ...  
## .. .. ..$ C5 : chr [1:13] "1" "2" "3" "4" ...  
## $ levels : chr [1:10] "0" "1" "2" "3" ...  
## $ isnumeric: Named logi [1:10] FALSE FALSE FALSE FALSE FALSE FALSE ...  
## ..- attr(\*, "names")= chr [1:10] "S1" "C1" "S2" "C2" ...  
## $ call : language naiveBayes.default(x = pokerTreino[, c("S1", "C1", "S2", "C2", "S3", "C3", "S4", "C4", "S5", "C5")], y = pok| \_\_truncated\_\_  
## - attr(\*, "class")= chr "naiveBayes"

Níveis e distribuição dos valores de treino:

nv$levels

## [1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"

nv$apriori

## pokerTreino[, c("CLASS")]  
## 0 1 2 3 4 5 6 7 8 9   
## 12493 10599 1206 513 93 54 36 6 5 5

Probabilidades de cada parâmetro:

for (i in 0:9) {  
 print(nv$tables[i])  
}

## named list()  
## $S1  
## S1  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.2473385 0.2523813 0.2459777 0.2543024  
## 1 0.2477592 0.2511558 0.2524766 0.2486084  
## 2 0.2172471 0.2678275 0.2553897 0.2595357  
## 3 0.2339181 0.2397661 0.2612086 0.2651072  
## 4 0.3010753 0.1827957 0.3333333 0.1827957  
## 5 0.2222222 0.2037037 0.2222222 0.3518519  
## 6 0.2500000 0.1666667 0.3055556 0.2777778  
## 7 0.0000000 0.1666667 0.5000000 0.3333333  
## 8 0.4000000 0.2000000 0.2000000 0.2000000  
## 9 0.2000000 0.2000000 0.2000000 0.4000000  
##   
## $C1  
## C1  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.07932442 0.07700312 0.07572240 0.07844393  
## 1 0.07934711 0.07783753 0.07613926 0.07557317  
## 2 0.06882255 0.07711443 0.06882255 0.06716418  
## 3 0.10136452 0.08966862 0.07212476 0.08187135  
## 4 0.05376344 0.04301075 0.07526882 0.07526882  
## 5 0.09259259 0.12962963 0.00000000 0.05555556  
## 6 0.05555556 0.05555556 0.02777778 0.13888889  
## 7 0.00000000 0.16666667 0.16666667 0.00000000  
## 8 0.40000000 0.20000000 0.00000000 0.00000000  
## 9 0.20000000 0.00000000 0.00000000 0.00000000  
## C1  
## pokerTreino[, c("CLASS")] 5 6 7 8  
## 0 0.07396142 0.07580245 0.07788361 0.07812375  
## 1 0.07613926 0.07368620 0.07972450 0.07661100  
## 2 0.08540630 0.06882255 0.07462687 0.08374793  
## 3 0.08576998 0.05847953 0.08771930 0.07212476  
## 4 0.07526882 0.09677419 0.04301075 0.12903226  
## 5 0.05555556 0.11111111 0.03703704 0.03703704  
## 6 0.08333333 0.11111111 0.05555556 0.13888889  
## 7 0.00000000 0.00000000 0.00000000 0.50000000  
## 8 0.20000000 0.00000000 0.00000000 0.00000000  
## 9 0.00000000 0.00000000 0.00000000 0.00000000  
## C1  
## pokerTreino[, c("CLASS")] 9 10 11 12  
## 0 0.08084527 0.07428160 0.07436164 0.07740335  
## 1 0.07689405 0.07538447 0.07944146 0.07642230  
## 2 0.07711443 0.08126036 0.08291874 0.08706468  
## 3 0.06042885 0.06432749 0.07992203 0.08187135  
## 4 0.06451613 0.08602151 0.06451613 0.10752688  
## 5 0.14814815 0.09259259 0.07407407 0.09259259  
## 6 0.08333333 0.11111111 0.05555556 0.00000000  
## 7 0.00000000 0.00000000 0.16666667 0.00000000  
## 8 0.20000000 0.00000000 0.00000000 0.00000000  
## 9 0.00000000 0.40000000 0.20000000 0.20000000  
## C1  
## pokerTreino[, c("CLASS")] 13  
## 0 0.07684303  
## 1 0.07679970  
## 2 0.07711443  
## 3 0.06432749  
## 4 0.08602151  
## 5 0.07407407  
## 6 0.08333333  
## 7 0.00000000  
## 8 0.00000000  
## 9 0.00000000  
##   
## $S2  
## S2  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.2529416 0.2452573 0.2494997 0.2523013  
## 1 0.2516275 0.2529484 0.2451175 0.2503066  
## 2 0.2562189 0.2537313 0.2288557 0.2611940  
## 3 0.2495127 0.2690058 0.2319688 0.2495127  
## 4 0.2365591 0.3118280 0.2258065 0.2258065  
## 5 0.2222222 0.2037037 0.2222222 0.3518519  
## 6 0.2222222 0.2777778 0.3055556 0.1944444  
## 7 0.0000000 0.5000000 0.1666667 0.3333333  
## 8 0.4000000 0.2000000 0.2000000 0.2000000  
## 9 0.2000000 0.2000000 0.2000000 0.4000000  
##   
## $C2  
## C2  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.07780357 0.07428160 0.07692308 0.07700312  
## 1 0.08142278 0.07746014 0.07378055 0.07529012  
## 2 0.07794362 0.09701493 0.09121061 0.07213930  
## 3 0.09356725 0.07407407 0.08576998 0.07992203  
## 4 0.02150538 0.05376344 0.06451613 0.05376344  
## 5 0.07407407 0.07407407 0.11111111 0.11111111  
## 6 0.05555556 0.16666667 0.05555556 0.11111111  
## 7 0.00000000 0.16666667 0.16666667 0.00000000  
## 8 0.00000000 0.20000000 0.00000000 0.40000000  
## 9 0.00000000 0.00000000 0.00000000 0.00000000  
## C2  
## pokerTreino[, c("CLASS")] 5 6 7 8  
## 0 0.07540223 0.07852397 0.07948451 0.07284079  
## 1 0.07274271 0.07934711 0.07378055 0.08029059  
## 2 0.07296849 0.06384743 0.07131012 0.06965174  
## 3 0.05847953 0.07797271 0.07212476 0.07602339  
## 4 0.06451613 0.09677419 0.06451613 0.07526882  
## 5 0.07407407 0.05555556 0.11111111 0.05555556  
## 6 0.11111111 0.11111111 0.02777778 0.05555556  
## 7 0.00000000 0.00000000 0.16666667 0.33333333  
## 8 0.00000000 0.20000000 0.00000000 0.00000000  
## 9 0.00000000 0.00000000 0.00000000 0.00000000  
## C2  
## pokerTreino[, c("CLASS")] 9 10 11 12  
## 0 0.07204034 0.07684303 0.07772353 0.08116545  
## 1 0.07878102 0.07679970 0.07717709 0.07368620  
## 2 0.07213930 0.07628524 0.07462687 0.07048093  
## 3 0.08576998 0.05847953 0.06822612 0.08382066  
## 4 0.08602151 0.20430108 0.06451613 0.08602151  
## 5 0.03703704 0.07407407 0.05555556 0.12962963  
## 6 0.02777778 0.08333333 0.11111111 0.00000000  
## 7 0.00000000 0.00000000 0.16666667 0.00000000  
## 8 0.00000000 0.00000000 0.00000000 0.20000000  
## 9 0.00000000 0.00000000 0.60000000 0.00000000  
## C2  
## pokerTreino[, c("CLASS")] 13  
## 0 0.07996478  
## 1 0.07944146  
## 2 0.09038143  
## 3 0.08576998  
## 4 0.06451613  
## 5 0.03703704  
## 6 0.08333333  
## 7 0.00000000  
## 8 0.00000000  
## 9 0.40000000  
##   
## $S3  
## S3  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.2486993 0.2454975 0.2501401 0.2556632  
## 1 0.2499292 0.2533258 0.2405887 0.2561562  
## 2 0.2479270 0.2504146 0.2454395 0.2562189  
## 3 0.2436647 0.2241715 0.2534113 0.2787524  
## 4 0.2365591 0.2043011 0.3118280 0.2473118  
## 5 0.2222222 0.2037037 0.2222222 0.3518519  
## 6 0.3055556 0.1666667 0.1944444 0.3333333  
## 7 0.3333333 0.1666667 0.3333333 0.1666667  
## 8 0.4000000 0.2000000 0.2000000 0.2000000  
## 9 0.2000000 0.2000000 0.2000000 0.4000000  
##   
## $C3  
## C3  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.07300088 0.07836388 0.07676299 0.07428160  
## 1 0.07840362 0.07425229 0.07925276 0.07953581  
## 2 0.08208955 0.07794362 0.09121061 0.08457711  
## 3 0.08771930 0.07602339 0.08382066 0.08966862  
## 4 0.04301075 0.00000000 0.01075269 0.10752688  
## 5 0.03703704 0.09259259 0.14814815 0.03703704  
## 6 0.02777778 0.11111111 0.05555556 0.11111111  
## 7 0.00000000 0.16666667 0.16666667 0.00000000  
## 8 0.00000000 0.20000000 0.20000000 0.00000000  
## 9 0.20000000 0.00000000 0.00000000 0.00000000  
## C3  
## pokerTreino[, c("CLASS")] 5 6 7 8  
## 0 0.07916433 0.07668294 0.07572240 0.07572240  
## 1 0.07444098 0.07500708 0.07934711 0.07764883  
## 2 0.06965174 0.06882255 0.07131012 0.06799337  
## 3 0.07017544 0.06237817 0.07602339 0.08382066  
## 4 0.12903226 0.09677419 0.08602151 0.07526882  
## 5 0.14814815 0.05555556 0.07407407 0.07407407  
## 6 0.02777778 0.11111111 0.05555556 0.13888889  
## 7 0.00000000 0.00000000 0.16666667 0.33333333  
## 8 0.20000000 0.00000000 0.00000000 0.00000000  
## 9 0.00000000 0.00000000 0.00000000 0.00000000  
## C3  
## pokerTreino[, c("CLASS")] 9 10 11 12  
## 0 0.07476187 0.08172577 0.07516209 0.08148563  
## 1 0.07293141 0.07887537 0.07717709 0.07887537  
## 2 0.07048093 0.08126036 0.07628524 0.07213930  
## 3 0.07992203 0.06432749 0.08187135 0.07212476  
## 4 0.13978495 0.06451613 0.10752688 0.05376344  
## 5 0.03703704 0.01851852 0.09259259 0.03703704  
## 6 0.08333333 0.08333333 0.16666667 0.02777778  
## 7 0.00000000 0.00000000 0.16666667 0.00000000  
## 8 0.20000000 0.20000000 0.00000000 0.00000000  
## 9 0.00000000 0.20000000 0.00000000 0.20000000  
## C3  
## pokerTreino[, c("CLASS")] 13  
## 0 0.07716321  
## 1 0.07425229  
## 2 0.08623549  
## 3 0.07212476  
## 4 0.08602151  
## 5 0.14814815  
## 6 0.00000000  
## 7 0.00000000  
## 8 0.00000000  
## 9 0.40000000  
##   
## $S4  
## S4  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.25318178 0.24925959 0.24845914 0.24909950  
## 1 0.24851401 0.25332579 0.25313709 0.24502312  
## 2 0.23548922 0.21724710 0.29436153 0.25290216  
## 3 0.24756335 0.26315789 0.25341131 0.23586745  
## 4 0.32258065 0.26881720 0.19354839 0.21505376  
## 5 0.22222222 0.20370370 0.22222222 0.35185185  
## 6 0.30555556 0.38888889 0.25000000 0.05555556  
## 7 0.83333333 0.00000000 0.16666667 0.00000000  
## 8 0.40000000 0.20000000 0.20000000 0.20000000  
## 9 0.20000000 0.20000000 0.20000000 0.40000000  
##   
## $C4  
## C4  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.07924438 0.08108541 0.08116545 0.07660290  
## 1 0.07906406 0.07840362 0.07849797 0.07434664  
## 2 0.08126036 0.07877280 0.07379768 0.08126036  
## 3 0.08382066 0.08382066 0.09356725 0.07407407  
## 4 0.06451613 0.00000000 0.05376344 0.10752688  
## 5 0.09259259 0.07407407 0.07407407 0.11111111  
## 6 0.08333333 0.00000000 0.11111111 0.11111111  
## 7 0.00000000 0.16666667 0.16666667 0.00000000  
## 8 0.00000000 0.00000000 0.40000000 0.20000000  
## 9 0.00000000 0.00000000 0.00000000 0.00000000  
## C4  
## pokerTreino[, c("CLASS")] 5 6 7 8  
## 0 0.07556231 0.07500200 0.07948451 0.07628272  
## 1 0.07585621 0.08047929 0.07576186 0.07151618  
## 2 0.07462687 0.06799337 0.06716418 0.07711443  
## 3 0.07407407 0.05847953 0.07602339 0.09356725  
## 4 0.04301075 0.06451613 0.18279570 0.06451613  
## 5 0.03703704 0.01851852 0.11111111 0.07407407  
## 6 0.02777778 0.11111111 0.05555556 0.05555556  
## 7 0.00000000 0.00000000 0.16666667 0.16666667  
## 8 0.00000000 0.00000000 0.20000000 0.00000000  
## 9 0.00000000 0.00000000 0.00000000 0.00000000  
## C4  
## pokerTreino[, c("CLASS")] 9 10 11 12  
## 0 0.07772353 0.07676299 0.07460178 0.07212039  
## 1 0.07736579 0.07632796 0.08066799 0.07538447  
## 2 0.07379768 0.08457711 0.09369818 0.07296849  
## 3 0.09356725 0.06822612 0.05847953 0.06237817  
## 4 0.09677419 0.09677419 0.12903226 0.08602151  
## 5 0.11111111 0.09259259 0.05555556 0.12962963  
## 6 0.08333333 0.11111111 0.13888889 0.00000000  
## 7 0.00000000 0.00000000 0.16666667 0.16666667  
## 8 0.00000000 0.00000000 0.20000000 0.00000000  
## 9 0.00000000 0.20000000 0.20000000 0.40000000  
## C4  
## pokerTreino[, c("CLASS")] 13  
## 0 0.07436164  
## 1 0.07632796  
## 2 0.07296849  
## 3 0.07992203  
## 4 0.01075269  
## 5 0.01851852  
## 6 0.11111111  
## 7 0.00000000  
## 8 0.00000000  
## 9 0.20000000  
##   
## $S5  
## S5  
## pokerTreino[, c("CLASS")] 1 2 3 4  
## 0 0.2490995 0.2436564 0.2533419 0.2539022  
## 1 0.2544580 0.2533258 0.2502123 0.2420040  
## 2 0.2645108 0.2280265 0.2686567 0.2388060  
## 3 0.2592593 0.2397661 0.2456140 0.2553606  
## 4 0.2258065 0.2258065 0.2580645 0.2903226  
## 5 0.2222222 0.2037037 0.2222222 0.3518519  
## 6 0.3055556 0.2500000 0.2500000 0.1944444  
## 7 0.0000000 0.3333333 0.0000000 0.6666667  
## 8 0.4000000 0.2000000 0.2000000 0.2000000  
## 9 0.2000000 0.2000000 0.2000000 0.4000000

rm(i)

Tentar predizer e verificar com o resultado

pokerTeste <- data.frame(pokerTeste)  
pokerTeste$resultado\_previsto <- predict(nv, newdata = pokerTeste[1:10], type = "class")

Matriz de confusão

matrizConfusao <- table(pokerTeste$resultado\_previsto,pokerTeste$CLASS)  
matrizConfusao

##   
## 0 1 2 3 4 5 6 7 8 9  
## 0 465870 394355 44744 19960 3525 1818 1369 223 11 3  
## 1 35316 28042 2855 1146 352 177 55 7 1 0  
## 2 0 0 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0 0 0  
## 4 0 0 0 0 0 0 0 0 0 0  
## 5 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0  
## 7 18 52 10 12 0 0 0 0 0 0  
## 8 5 0 0 0 0 1 0 0 0 0  
## 9 0 49 13 3 8 0 0 0 0 0

Porcentagem de erros

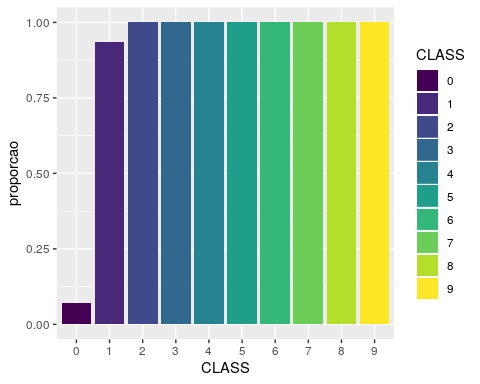
paste("Erro na estimativa durante treino:", sum(as.numeric(pokerTeste$resultado\_previsto) != as.numeric(pokerTeste$CLASS)) / nrow(pokerTeste))

## [1] "Erro na estimativa durante treino: 0.506088"

50% de erro é um valor muito alto. Precisaremos fornecer mais informações para o algoritmo.

Distribuição do erro por classe:

res <- transmute(pokerTeste, CLASS, resultado\_previsto, errado = (as.numeric(resultado\_previsto) != as.numeric(CLASS)))  
res <- group\_by(res,CLASS) %>% summarise(count = n(), errors = sum(errado), proporcao = sum(errado)/n())  
ggplot(data=res, aes(x = CLASS, y = proporcao)) + geom\_bar(aes(fill=CLASS), stat = "identity")



rm(res)

Limpeza

rm(matrizConfusao)  
rm(nv)

# Novas colunas

Incluir colunas para: - Contagem de cartas iguais - Contagem de cartas do mesmo naipe - É sequencial?

criarColunas <- function(dataset) {  
 dataset$id = seq.int(nrow(dataset))  
 #Contagem de naipes  
 #transformar os 5 valores da coluna em linhas  
 dsGather <- select(dataset,id,S1,S2,S3,S4,S5) %>% gather(key,suit,S1:S5)  
 #para cada item, agrupar e pegar o maior  
 dsGroup1 <- group\_by(select(dsGather,-key),id,suit) %>% summarise(soma = n())  
 dsGroup2 <- group\_by(dsGroup1,id) %>% summarise(suit = max(soma))  
 dataset <- merge(dataset, dsGroup2, by.x = "id", by.y = "id")  
   
 #Contagem de valores  
 dsGather <- select(dataset,id,C1,C2,C3,C4,C5) %>% gather(key,rank,C1:C5)  
 dsGroup1 <- group\_by(select(dsGather,-key),id,rank) %>% summarise(soma = n())  
 dsGroup2 <- group\_by(dsGroup1,id) %>% summarise(rank = max(soma))  
 dataset <- merge(dataset, dsGroup2, by.x = "id", by.y = "id")  
   
 #Teste de sequencial  
 dsGather <- select(dataset,id,C1,C2,C3,C4,C5) %>% gather(key,rank,C1:C5)  
 #inserir um sequencial para manter ordenação nos groupBy  
 dsGather$rank <- as.numeric(dsGather$rank)  
 dsGather <- arrange(dsGather,id,rank) %>% select(id,rank)  
 dsGather$pos <- rowid(dsGather$id)  
 dsGather$previous <- c(0,dsGather$rank[1:nrow(dsGather)-1])  
 #se é o primeiro da lista, não pode ser sequencial  
 dsGather$sequential <- ifelse(dsGather$pos == 1,0,dsGather$rank-1 == dsGather$previous)  
 #Sequencial especial: 1 e 13. Se existe 1 e 13 para um ID em específico, iremos dizer que o 1 é sequencial  
 idsContem1e13 <- intersect(unique(dsGather$id[dsGather$rank==1]),unique(dsGather$id[dsGather$rank==13]))  
 dsGather$sequential[dsGather$id %in% idsContem1e13 & dsGather$pos == 1 & dsGather$rank == 1] <- 1  
 dsGroup1 <- group\_by(dsGather,id) %>% summarise(is\_sequential = sum(sequential) == 4)  
 dataset <- merge(dataset, dsGroup1, by.x = "id", by.y = "id")  
 rm(idsContem1e13)  
 rm(dsGather)  
 rm(dsGroup1)  
 rm(dsGroup2)  
 dataset <- data.frame(select(dataset,-c(id)))  
 return(dataset)  
}  
  
pokerTreino <- criarColunas(pokerTreino)  
pokerTeste <- criarColunas(pokerTeste)  
rm(criarColunas)

Vamos treinar novamente:

#dadosTreino <- data.frame(lapply(pokerTreino, function(x) as.factor(x)))  
pokerTreino <- data.frame(pokerTreino)  
nv <- naiveBayes(pokerTreino[,c("S1","C1","S2","C2","S3","C3","S4","C4","S5","C5","suit","rank","is\_sequential")],pokerTreino[,c("CLASS")])  
nv$levels

## [1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"

nv$apriori

## pokerTreino[, c("CLASS")]  
## 0 1 2 3 4 5 6 7 8 9   
## 12493 10599 1206 513 93 54 36 6 5 5

Novos testes (com a base de treino e a base de teste):

pokerTreino$resultado <- predict(nv, newdata = pokerTreino, type = "class")  
pokerTeste <- data.frame(pokerTeste)  
pokerTeste$resultado <- predict(nv, newdata = pokerTeste, type = "class")  
  
dadosTreino <- data.frame(lapply(pokerTreino, function(x) as.factor(x)))  
pokerTreino$resultado <- predict(nv, newdata = dadosTreino, type = "class")

## Warning in predict.naiveBayes(nv, newdata = dadosTreino, type = "class"):  
## Type mismatch between training and new data for variable 'suit'. Did you  
## use factors with numeric labels for training, and numeric values for new  
## data?

## Warning in predict.naiveBayes(nv, newdata = dadosTreino, type = "class"):  
## Type mismatch between training and new data for variable 'rank'. Did you  
## use factors with numeric labels for training, and numeric values for new  
## data?

rm(dadosTreino)  
  
dadosTeste <- data.frame(lapply(pokerTeste, function(x) as.factor(x)))  
pokerTeste$resultado <- predict(nv, newdata = dadosTeste, type = "class")

## Warning in predict.naiveBayes(nv, newdata = dadosTeste, type = "class"):  
## Type mismatch between training and new data for variable 'suit'. Did you  
## use factors with numeric labels for training, and numeric values for new  
## data?

## Warning in predict.naiveBayes(nv, newdata = dadosTeste, type = "class"):  
## Type mismatch between training and new data for variable 'rank'. Did you  
## use factors with numeric labels for training, and numeric values for new  
## data?

rm(dadosTeste)

Matriz de confusão

matrizConfusao <- table(pokerTreino$resultado,pokerTreino$CLASS)  
matrizConfusao

##   
## 0 1 2 3 4 5 6 7 8 9  
## 0 11711 9771 1129 478 2 51 35 3 0 0  
## 1 778 828 77 35 0 3 1 0 0 0  
## 2 0 0 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0 0 0  
## 4 2 0 0 0 89 0 0 0 1 0  
## 5 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0  
## 7 1 0 0 0 0 0 0 3 0 0  
## 8 1 0 0 0 0 0 0 0 4 0  
## 9 0 0 0 0 2 0 0 0 0 5

matrizConfusao <- table(pokerTeste$resultado,pokerTeste$CLASS)  
matrizConfusao

##   
## 0 1 2 3 4 5 6 7 8 9  
## 0 464331 393337 44642 19928 752 1812 1365 221 1 0  
## 1 36582 29108 2970 1181 0 181 59 9 0 0  
## 2 0 0 0 0 0 0 0 0 0 0  
## 3 0 0 0 0 0 0 0 0 0 0  
## 4 238 0 0 0 3017 3 0 0 11 1  
## 5 0 0 0 0 0 0 0 0 0 0  
## 6 0 0 0 0 0 0 0 0 0 0  
## 7 18 53 10 12 0 0 0 0 0 0  
## 8 25 0 0 0 39 0 0 0 0 0  
## 9 15 0 0 0 77 0 0 0 0 2

Porcentagem de erros.

paste("Erro de predição no dataset de treino:", sum(as.numeric(pokerTreino$resultado) != as.numeric(pokerTreino$CLASS)) / nrow(pokerTreino))

## [1] "Erro de predição no dataset de treino: 0.494602159136345"

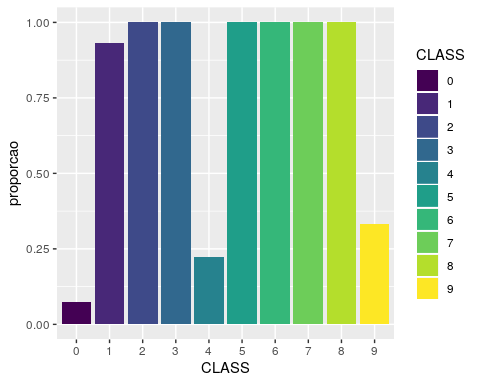
paste("Erro de predição no dataset de teste:", sum(as.numeric(pokerTeste$resultado) != as.numeric(pokerTeste$CLASS)) / nrow(pokerTeste) )

## [1] "Erro de predição no dataset de teste: 0.503542"

Aproximadamente 6%.

Distribuição do erro por classe:

res <- transmute(pokerTeste, CLASS, resultado, errado = (as.numeric(resultado) != as.numeric(CLASS)))  
res <- group\_by(res,CLASS) %>% summarise(count = n(), errors = sum(errado), proporcao = sum(errado)/n())  
ggplot(data=res, aes(x = CLASS, y = proporcao)) + geom\_bar(aes(fill=CLASS), stat = "identity")



# Fontes

Becker, Lauro. **Algoritmo de Classificação Naive Bayes**. 2019. Disponível em: <https://www.organicadigital.com/blog/algoritmo-de-classificacao-naive-bayes/>. Acesso em 12 de outubro de 2019.

Gonçalves, Thiago. **Teorema de Bayes: o que é e qual sua aplicação?**. 2019. Disponível em: <https://www.voitto.com.br/blog/artigo/teorema-de-bayes>. Acesso em 12 de outubro de 2019.

Oppermann, Artem. **Bayes’ Theorem: The Holy Grail of Data Science**. 2018. Disponível em: <https://towardsdatascience.com/bayes-theorem-the-holy-grail-of-data-science-55d93315defb>. Acesso em 06 de Outubro de 2019.

Oracle Corporation, 2017. **Naive Bayes, Oracle Database Online Documentation Library, 12c Release 1 (12.1.0.2)**. Disponível em: <https://docs.oracle.com/database/121/DMCON/GUID-BB77D68D-3E07-4522-ACB6-FD6723BDA92A.htm#DMCON343>. Acesso em 06 de Outubro de 2019.

Robert Cattral, Franz Oppacher. **UCI Poker Hand Dataset**. 2007. Disponível em <https://archive.ics.uci.edu/ml/datasets/Poker+Hand>. Acesso em 06 de outubro de 2019

S.L. Ting, W.H. Ip, Albert H.C. Tsang. \*\*Is Naïve Bayes a Good Classifier for Document Classification?\*"\*\* International Journal of Software Engineering and Its Applications, University, Hung Hum, Kowloon, Hong Kong, 2011

**As Regras de Poker Online**. In: Pokerstars. Disponível em: <https://www.pokerstars.com/br/poker/games/rules/?no_redirect=1>. Acesso em 12 de outubro de 2019.

**Regras de Poker: como jogar poker passo a passo**. In: PokerNews. Disponível em: <https://br.pokernews.com/regras-poker/>. Acesso em 12 de Outubro de 2019.

**Bayesian inference**. In: Wikipedia, a enciclopédia livre. Wikimedia Foundation, 2019. Disponível em: <https://en.wikipedia.org/wiki/Bayesian_inference>. Acesso em 12 de Outubro de 2019.

**Bayes’ theorem**. In: Wikipedia, a enciclopédia livre. Wikimedia Foundation, 2019. Disponível em: <https://en.wikipedia.org/wiki/Bayes%27_theorem>. Acesso em 06 de Outubro de 2019.

**Naive Bayes classifier**. In: Wikipedia, a enciclopédia livre. Wikimedia Foundation, 2019. Disponível em: <https://en.wikipedia.org/w/index.php?title=Naive_Bayes_classifier&oldid=920481611>. Acesso em 06 de Outubro de 2019.

**Bayesian probability**. In: Wikipedia, a enciclopédia livre. Wikimedia Foundation, 2019. Disponível em: <https://en.wikipedia.org/w/index.php?title=Bayesian_probability&oldid=920310586>. Acesso em 06 de Outubro de 2019.