

PokerHand

Fernando e Leonardo

01/10/2019

Avaliação

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.

Tipos de mãos do Poker

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

A ordem dessa tabela é da mão mais baixa à mão mais alta. A que possui número de classe maior vence sobre as mãos de classe menor.

Estrutura do Dataset

O dataset PokerHands possui as seguintes colunas: 1) S1 - Naipes 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 - Naipes da carta #2: Ordinal (1-4) representando o naipe 4) C2 - Valor da carta #2: Numérico (1-13) representando o valor 5) S3 - Naipes da carta #3: Ordinal (1-4) representando o naipe 6) C3 - Valor da carta #3: Numérico (1-13) representando o valor 7) S4 - Naipes da carta #4: Ordinal (1-4) representando o naipe 8) C4 - Valor da carta #4: Numérico (1-13) representando o valor 9) S5 - Naipes 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

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':
```

Carregando o arquivo

Leitura do arquivo. Os dados serão lidos como “factors” porque são variáveis ordinais discretas.

```
pokerTreino <- read_csv("datasets/poker-hand-training-true")  
head(pokerTreino)
```

```
## # A tibble: 6 x 11
```

```
##   S1      C1      S2      C2      S3      C3      S4      C4      S5  
##   <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct> <fct>  
## 1 1      10      1      11      1      13      1      12      1  
## 2 2      11      2      13      2      10      2      12      2  
## 3 3      12      3      11      3      13      3      10      3  
## 4 4      10      4      11      4      1       4      13      4  
## 5 4      1       4      13      4      12      4      11      4  
## 6 1      2       1      4       1      5       1      3       1
```

Leitura do arquivo de teste

```
pokerTeste <- read_csv("datasets/poker-hand-testing.data",  
head(pokerTeste))
```

Ajustes

```
pokerTreino$CLASS <- factor(pokerTreino$CLASS, levels=c(0,1,2))  
pokerTeste$CLASS <- factor(pokerTeste$CLASS, levels=c(0,1,2))
```


Análise do arquivo

```
str(pokerTreino)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame'
## $ S1 : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 4
## $ C1 : Factor w/ 13 levels "10","11","12",...: 1 2 3 1
## $ S2 : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 4
## $ C2 : Factor w/ 13 levels "11","13","4",...: 1 2 1 1
## $ S3 : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 4
## $ C3 : Factor w/ 13 levels "13","10","1",...: 1 2 1 3
## $ S4 : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 4
## $ C4 : Factor w/ 13 levels "12","10","13",...: 1 1 2 3
## $ S5 : Factor w/ 4 levels "1","2","3","4": 1 2 3 4 4
## $ C5 : Factor w/ 13 levels "1","12","10",...: 1 1 1 2
## $ CLASS: Ord.factor w/ 10 levels "0"<"1"<"2"<"3"<...: 10
## - attr(*, "spec")=
## .. cols(
## .. .default = col_factor(),
## .. S1 = col_factor(levels = NULL, ordered = FALSE,
## .. C1 = col_factor(levels = NULL, ordered = FALSE,
```

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

```
head(pokerTreino)
```

```
## # A tibble: 6 x 11
```

	S1	C1	S2	C2	S3	C3	S4	C4	S5
	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>
## 1	1	10	1	11	1	13	1	12	1
## 2	2	11	2	13	2	10	2	12	2
## 3	3	12	3	11	3	13	3	10	3
## 4	4	10	4	11	4	1	4	13	4

Treinamento

Treinar com o Naïve Bayes

```
pokerTreino <- data.frame(pokerTreino)
resultado <- naiveBayes(pokerTreino[,c("S1", "C1", "S2", "C2")])
```

Como o algoritmo estruturou seus parâmetros:

```
str(resultado)

## List of 5
## $ apriori : 'table' int [1:10(1d)] 12493 10599 1206 51
## ..- attr(*, "dimnames")=List of 1
## .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2"
## $ tables :List of 10
## ..$ S1: 'table' num [1:10, 1:4] 0.247 0.248 0.217 0.23
## .. ..- attr(*, "dimnames")=List of 2
## .. .. ..$ pokerTreino[, c("CLASS")]: chr [1:10] "0" "1" "2"
## .. .. ..$ S1 : chr [1:4] "1" "2"
## ..$ C1: 'table' num [1:10, 1:13] 0.0743 0.0754 0.0813
## .. ..- attr(*, "dimnames")=List of 2
```

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()  
  #para cada item, agrupar e pegar o maior  
  dsGroup1 <- group_by(select(dsGather,-key),id,suit) %>%  
  dsGroup2 <- group_by(dsGroup1,id) %>% summarise(suit = max(suit))  
  dataset <- merge(dataset, dsGroup2, by.x = "id", by.y = "id", all = TRUE)  
  
  #Contagem de valores  
  dsGather <- select(dataset,id,C1,C2,C3,C4,C5) %>% gather()  
  dsGroup1 <- group_by(select(dsGather,-key),id,rank) %>%  
  dsGroup2 <- group_by(dsGroup1,id) %>% summarise(rank = max(rank))  
  dataset <- merge(dataset, dsGroup2, by.x = "id", by.y = "id", all = TRUE)
```

Fontes

[1] UCI Poker Hand Dataset

<https://archive.ics.uci.edu/ml/datasets/Poker+Hand>

[2] [https:](https://www.pokerstars.com/br/poker/games/rules/?no_redirect=1)

[//www.pokerstars.com/br/poker/games/rules/?no_redirect=1](https://www.pokerstars.com/br/poker/games/rules/?no_redirect=1)

[3] <https://br.pokernews.com/regras-poker/>

[] https://en.wikipedia.org/wiki/Naive_Bayes_classifier []

https://en.wikipedia.org/wiki/Bayesian_inference []

https://en.wikipedia.org/wiki/Bayes%27_theorem []

https://en.wikipedia.org/wiki/Maximum_likelihood_estimation