Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais

KOBE BRYANT: ARREMESSOS

Pré-processamento e análise da base de dados de arremessos do jogador de basquete Kobe Bryant

Jádyla Maria Cesário Firmino

Poços de Caldas - MG

2022

SUMÁRIO

Jádyla Maria Cesário Firmino
7º período de Engenharia de Computação

OBJETIVO

O aprendizado de máquina consiste em, a partir de uma base de dados, possibilitar que a máquina aprenda como cada atributo age sobre sua saída, e, com isso, de certa forma ser capaz de especular com certa precisão uma saída a partir de suas entradas.

Entretanto, antes de partir para a aplicação de algoritmos e o que de fato seria o processo de aprendizado da chamada Inteligência Artificial, é necessário passar pela etapa de pré-processamento de dados, que consiste em preparar os dados para leitura e implementação da IA.

A base de dados aqui em questão trata-se do registro de todas as cestas arremessadas em jogos pelo jogador da NBA Kobe Bryant, dentre as mais diversas circunstâncias e variáveis, os chamados atributos. No total a base contém 24 atributos de entrada mais o atributo de saída, e um total de 30,697 mil instâncias. O objetivo será realizar o pré-processamento desta base, aplicando técnicas de balanceamento, limpeza, etc.

A tecnologia utilizada foi a linguagem de programação *R*, aplicada no editor de texto *R Studio*, que gerou os gráficos aqui presentes.

IDENTIFICAÇÃO

1. Identificação do atributo alvo

A base de dados em questão apresenta diversos atributos que representam situações e condições diferentes para o arremesso do Kobe Bryant. Todos os atributos influenciam diretamente no quesito converter a cesta ou não, e, com isso, pode-se concluir que o atributo de saída, atributo alvo, trata-se do *shot_made_flag,* um valor binário no qual o 0 representa uma cesta não convertida e o 1, por sua vez, o arremesso bem sucedido.

Para padronizar a tabela foi utilizado o código apresentado pela figura 1.1, para que o atributo alvo ficasse localizado na última coluna da base de dados, como mostra a figura 1.2.

```
#TOPICO 1- COLOCANDO O ATRIBUTO ALVO NA ÚLTIMA COLUNA
shotMade <- select(data, shot_made_flag)

data$shot_made_flag <-NULL

shots <- data %>%
  mutate(shotMade, shot_made_flag = shot_made_flag)

View(shots)
```

Figura 1.1

game_date	matchup	opponent	shot_id	shot_made_flag
2000-10-31	LAL @ POR	POR	1	NA
2000-10-31	LAL @ POR	POR	2	0
2000-10-31	LAL @ POR	POR	3	1
2000-10-31	LAL @ POR	POR	4	0
2000-10-31	LAL @ POR	POR	5	1
2000-10-31	LAL @ POR	POR	6	0
2000-10-31	LAL @ POR	POR	7	1
2000-10-31	LAL @ POR	POR	8	NA
2000-10-31	LAL @ POR	POR	9	1
2000-10-31	LAL @ POR	POR	10	0

Figura 1.2

Como apresentado no código, após *data* receber a base de dados que irá ler, ela terá o atributo alvo excluído e posteriormente adicionado ao final da tabela.

2. Identificação dos tipos e escala de dados dos atributos de entrada

Os atributos podem ser classificados como quantitativos, que se dividem em intervalares ou racionais, ou como qualitativos, que por sua vez podem ser nominais ou ordinais. Neste tópico será abordado o que cada atributo indica e sua classificação. Posteriormente, ao abordar as medidas de localidade, serão expostos as classes pertencentes a cada atributo.

- *action_type*: **qualitativo nominal**. Indica o tipo de movimento realizado ao arremessar para a cesta. Trata-se de um atributo nominal por não apresentar relação de grandeza entre si, apenas indicar o nome de um movimento;
- *combined_shot_type*: **qualitativo nominal**. Representa o tipo de movimento combinado com o movimento principal, realizado ao arremessar para a cesta;

- *game_event_id*: **qualitativo nominal**. Identificador do evento. Apesar de numérico não trata-se de um valor quantitativo, apenas uma identificação;
- *game_id*: **qualitativo nominal**. Trata-se da identificação do jogo no qual a cesta foi feita. Apesar de se referir a um dado numérico, não se classifica como quantitativo por seus valores não possuírem significado, mas se tratarem apenas de uma identificação (como o CPF, por exemplo);
- *lat*: **quantitativo intervalar**. Esse atributo indica a latitude onde ocorreu o jogo, ou seja, a partir desta informação e da longitude é possível analisar se o jogo foi em casa ou na casa do adversário, tendo em vista que este atributo não é presente. Trata-se de um atributo intervalar por variar especificamente em um intervalo de latitude e longitude limitado aos EUA, onde acontece a NBA;
- *lon*: **quantitativo intervalar**. De semelhante modo ao atributo anterior, este tem como objetivo indicar a localização, mas para a longitude. Em combinação indica onde seria o jogo, e também tem seus valores em um intervalo específico;
- *loc_x*: **quantitativo intervalar**. Indica em que posição da quadra no eixo *x* o jogador Kobe Bryant estaria na hora do arremesso. Com a função indicada pela figura 2.1 foi possível gerar a plotagem indicada pela figura 2.2, que possibilitou determinar sobre o que seria o atributo. Foi classificado como um valor intervalar por ser limitado às dimensões da quadra;
- *loc_y*: **quantitativo intervalar**. Indica em que posição da quadra no eixo *y* o jogador Kobe Bryant estaria na hora do arremesso. Este atributo também foi determinado através da análise da imagem gerada, apresentada na figura 2.2;

```
#gráfico que indica como foi possível analisar sobre o que se referiam os atributos loc_x e loc_y
ggplot(data = shots) +
  geom_point(mapping = aes(x = loc_x, y = loc_y, color = shot_type))
```

Figura 2.1

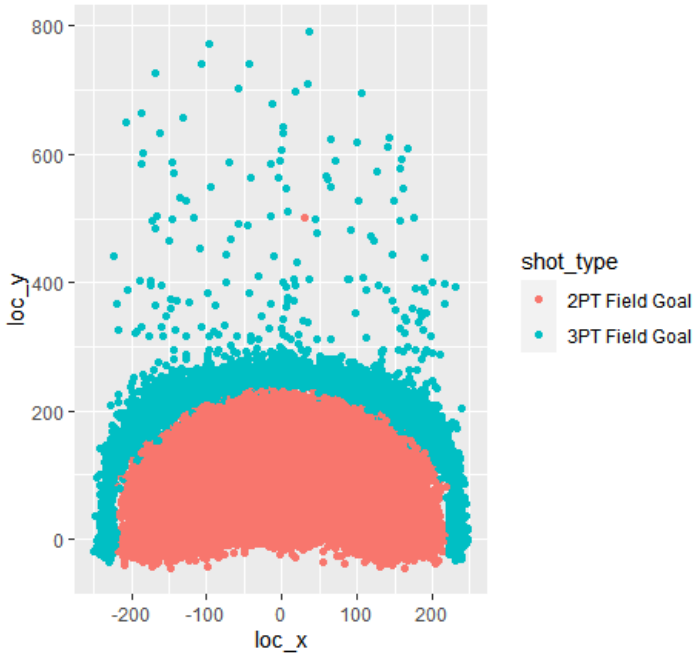


Figura 2.2

- *minutes_remaining*: **quantitativo racional**. Indica quantos minutos faltavam para que o jogo se encerrasse no momento em que o arremesso foi feito. Trata-se de um atributo racional principalmente pelo zero absoluto ter significado, ou seja, caso o tempo seja zero, o quarto se encerrou;
- *period*: **qualitativo ordinal**. Apesar de ser um valor numérico, apenas representa em ordem em qual quarto o jogo estava no momento do arremesso, e dessa forma pode ser lido como primeiro, segundo, terceiro e quarto. Está diretamente relacionado aos minutos restantes pois o valor indicado nestes minutos é referente ao tempo no quarto em questão;
- *playoffs*: **qualitativo nominal**. Atributo binário que indica se tratava-se de um jogo na fase de playoffs ou não;
- *season*: **qualitativo nominal**. Indica a temporada do jogo no qual o arremesso foi feito;
- *seconds_remaining*: **quantitativo racional**. Assim como os minutos indica quantos segundos faltavam para que o quarto se encerrasse no momento do arremesso;
- *shot_distance*: **quantitativo intervalar**. Indica a distância da cesta no momento do arremesso. De semelhante modo foi classificado como intervalar por se limitar às dimensões da quadra. Um fator interessante analisado é que

em 2010 as cestas de 3 pontos passaram de 6,25m para 6,75m, influenciando então no atributo *shot_type*;

- *shot_type*: **qualitativo nominal**. Indica se o arremesso foi de 2 pontos ou de 3 pontos;
- *shot_zone_area*: **qualitativo nominal**. Representa uma divisão da quadra em áreas distintas, sendo possível analisar na figura 2.4, gerada pelo código indicado pela figura 2.3;

```
#gráfico que indica os locais que são definidos pelo atributo 'shot_zone_area'
ggplot(data = shots) +
  geom_point(mapping = aes(x = loc_x, y = loc_y, color = shot_zone_area))
```

Figura 2.3

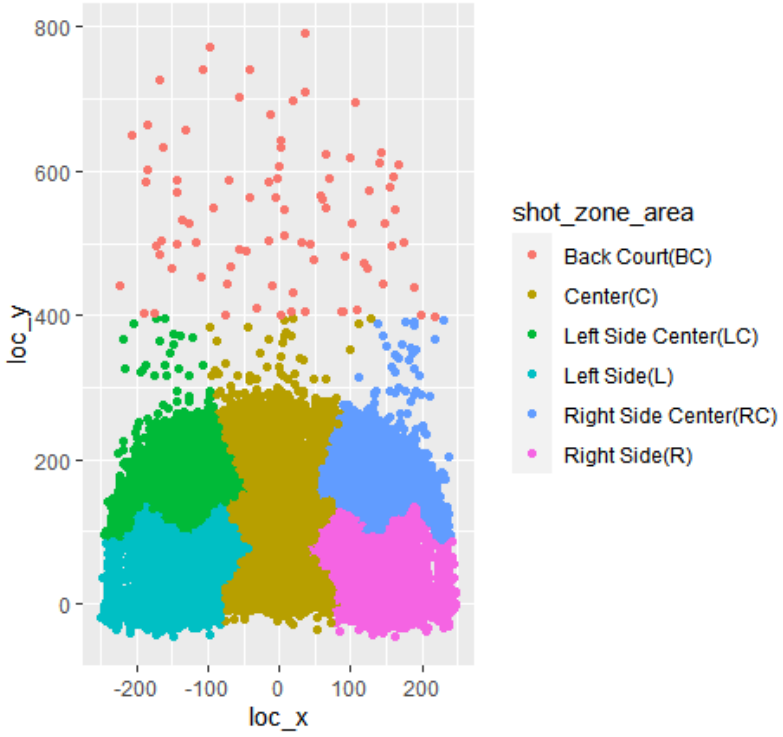


Figura 2.4

- *shot_zone_basic*: **qualitativo nominal**. Representa outro tipo de divisão de quadra, também dividindo a quadra e zonas, como mostra a figura 2.6 gerada através do código indicado na figura 2.5;

```
#gráfico que indica os locais que são definidos pelo atributo 'shot_zone_basic'
ggplot(data = shots) +
  geom_point(mapping = aes(x = loc_x, y = loc_y, color = shot_zone_basic))
```

Figura 2.5

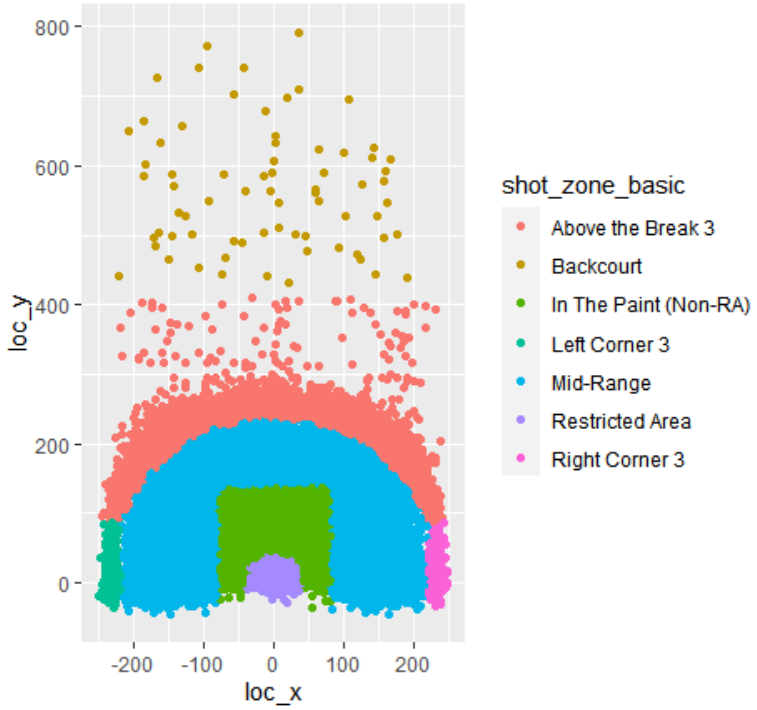


Figura 2.6

- *shot_zone_range*: **qualitativo ordinal**. Como os atributos anteriores, também representa uma divisão diferente na quadra, de acordo com a figura 2.8 gerada pelo código da figura 2.7. Entretanto, diferente das marcações anteriores, pode-se relacionar essa divisão com a proximidade da cesta, logo, representar de maneira ordinal;

```
#gráfico que indica os locais que são definidos pelo atributo 'shot_zone_range'
ggplot(data = shots) +
  geom_point(mapping = aes(x = loc_x, y = loc_y, color = shot_zone_range))
```

Figura 2.7

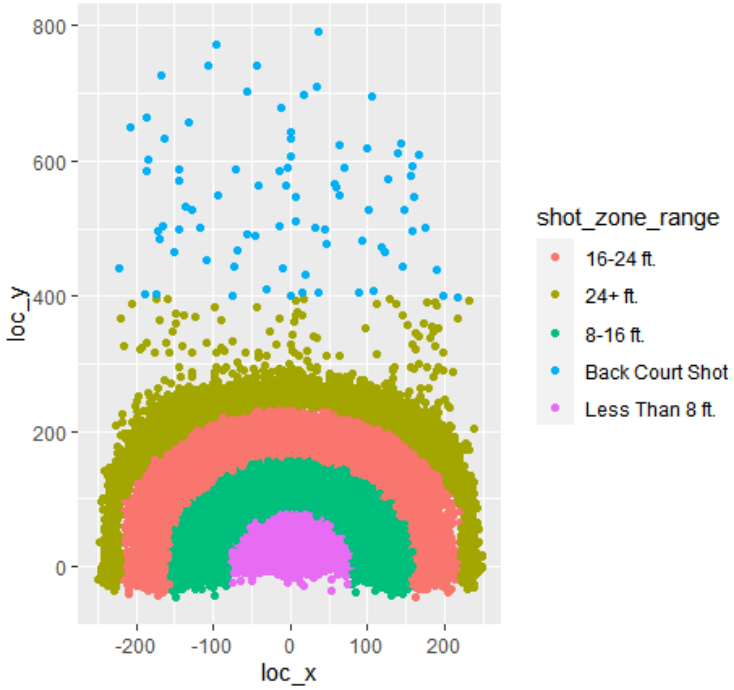


Figura 2.8

- *team_id*: **qualitativo nominal**. Trata-se de uma identificação do time do Kobe Bryant para cada instância, e, ao analisar individualmente, é possível ver que em todos os jogos o jogador integra o time dos Los Angeles Lakers, e por isso todos os arremessos possuem o mesmo valor nesse atributo;
- *team_name*: **qualitativo nominal**. Informa o nome do time, neste caso o Los Angeles Lakers para todas as instâncias, como explicitado no atributo anterior;
- *game_date*: **qualitativo ordinal**. Indica a data em que o jogo ocorreu. A ordem é importante pois pode indicar melhora ou piora no decorrer do tempo;
- *matchup*: **qualitativo nominal**. Expõe os dois times em quadra;
- *opponent*: **qualitativo nominal**. Revela qual é o oponente dos Los Angeles Lakers (único time apresentado na base de dados para o qual o jogador jogou);
- *shot_id*: **qualitativo nominal**. Identificação individual do jogo, que não está relacionada ou ordenada de acordo com o ano no qual o jogo foi realizado.

EXPLORAÇÃO

3. Exploração dos dados através de medidas de localidade

Observar e identificar o comportamento dos dados é de extrema importância ao realizarmos uma análise de dados. Diversas medidas de localidade podem ser aplicadas para entender como os valores de cada atributo irão se dividir, dentre elas estão: média, mediana, moda, quartis e percentil.

A aplicação de cada medida é feita por atributo, e a determinação de qual delas será aplicada depende da classificação deste, como feito anteriormente. Com isso, tem-se que, para esta base de dados que possui todos os tipos de dados (nominal, ordinal, intervalar e racional) será adotado o seguinte critério:

- Qualitativo nominal: moda;
- Qualitativo ordinal: moda;
- Quantitativo intervalar: média, mediana, moda e quartis;
- Quantitativo racional: média, mediana, moda e quartis.

Abaixo os atributos estão divididos segundo suas devidas classificações, e com isso apresentam as medidas como definido anteriormente. A figura 3.1 e 3.2 apresentam o código utilizado para a geração das informações.

```r
#funções das medidas de localidade
moda <- function(atributo){
  subset (table(atributo), table(atributo) == max (table (atributo)))
}

mediana <- function(atributo){
  median(atributo)
}

media <- function(atributo){
  mean(atributo)
}
```

Figura 3.1 - funções que geraram as medidas de localidade

```r
moda(shots$seconds_remaining)
media(shots$seconds_remaining)
mediana(shots$seconds_remaining)
boxplot(shots$seconds_remaining, main = "Segundos faltantes (atributo 'seconds_remaining')")
```

Figura 3.2 - exemplo de chamada de função para geração das informações

Atributo	Classif.	Média	Mediana	Moda | count	Boxplot
action_type	QN	-	-	*Jump Shot* | 18880	-
combined_shot_type	QN	-	-	*Jump Shot* | 23485	-
game_event_id	QN	-	-	*2* | 132	-
game_id	QN	-	-	*21501228* | 50	-
playoffs	QN	-	-	*0* | 26198	-
season	QN	-	-	*2005-06* | 2318	-
shot_type	QN	-	-	*2PT Field Goal* | 24271	-
shot_zone_area	QN	-	-	*Center(C)* | 13455	-
shot_zone_basic	QN	-	-	*Mid-Range* | 12625	-
team_id	QN	-	-	*1610612747* | 30697	-
team_name	QN	-	-	*Los Angeles Lakers* | 30697	-
matchup	QN	-	-	*LAL @ SAS* | 1020	-
opponent	QN	-	-	*SAS* | 1978	-
shot_id	QN	-	-	*id individual, logo, sem moda*	-
period	QO	-	-	*3* | 8296	-
shot_zone_range	QO	-	-	*Less Than 8 ft.* | 9398	-
game_date	QO	-	-	*2016-04-13* | 50	-
lat	QI	33.95319	33.9703	*34.0443* | 5599	3.3
lon	QI	-118.2627	-118.2698	*-118.2698* | 5475	3.4
loc_x	QI	7.110499	0	*0* | 5475	3.5
loc_y	QI	91.10753	74	*0* | 5599	3.6
shot_distance	QI	13.43744	15	*0* | 5542	3.7
minutes_remaining	QR	4.885624	5	*0* | 3866	3.8

Atributo	Classif.	Média	Mediana	Moda | count	Boxplot
seconds_remaining	QR	28.36508	28	*0* | 985	3.9

Legenda: QN - qualitativo nominal / QO - qualitativo ordinal / QI - quantitativo intervalar / QR - quantitativo racional

Tabela 3.1

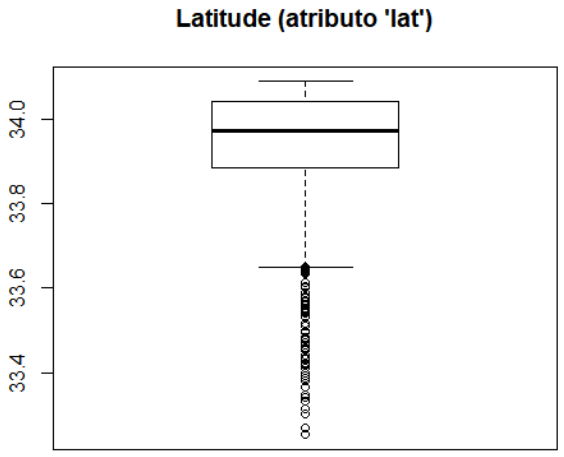


Figura 3.3 - Boxplot do atributo *lat*

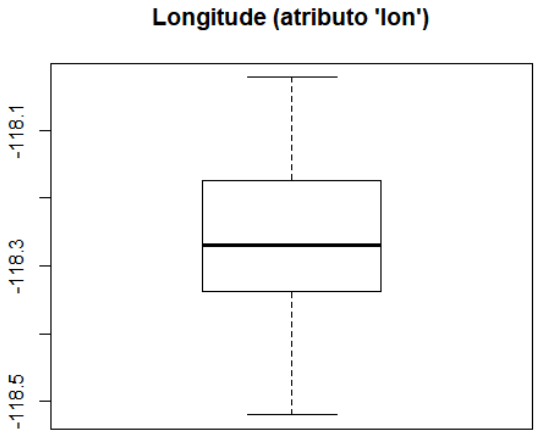


Figura 3.4 - Boxplot do atributo *lon*

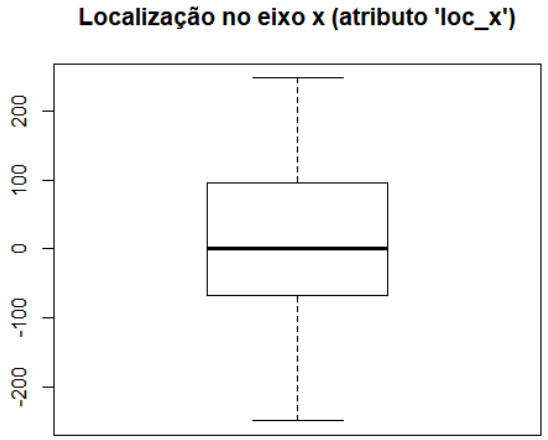


Figura 3.5 - Boxplot do atributo *loc_x*

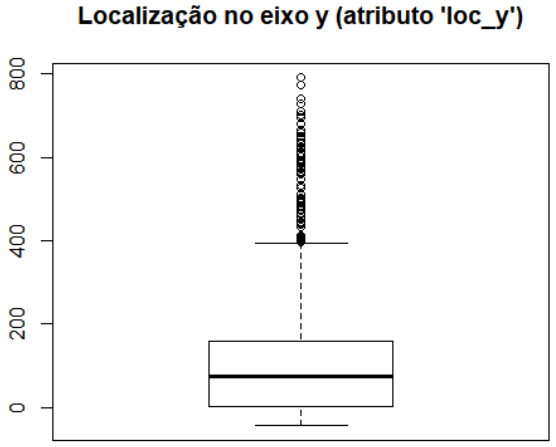


Figura 3.6 - Boxplot do atributo *loc_y*

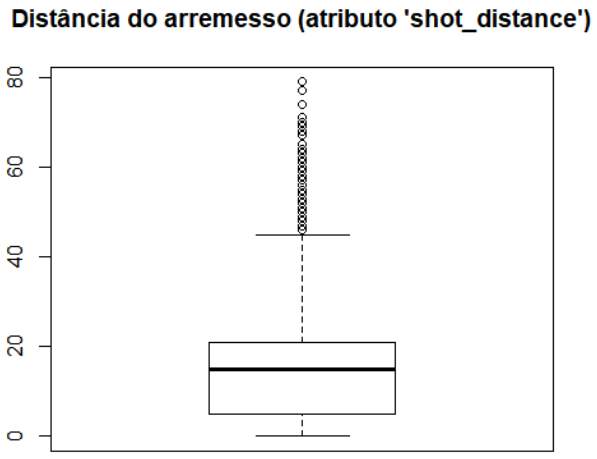


Figura 3.7 - Boxplot do atributo *shot_distance*

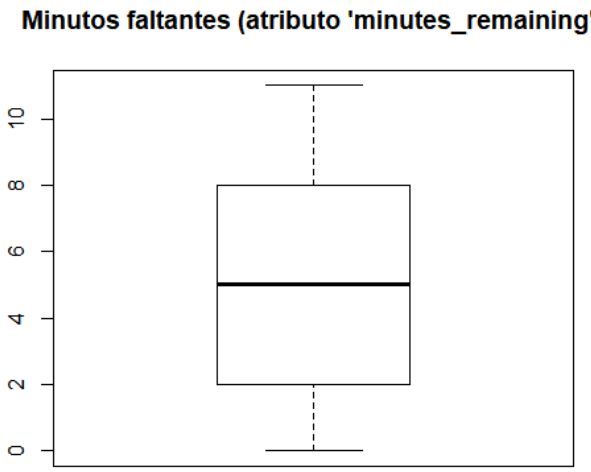


Figura 3.8 - Boxplot do atributo *minutes_remaining*

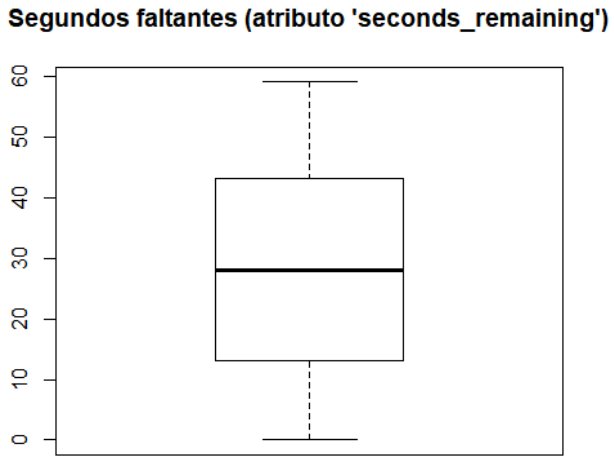


Figura 3.9 - Boxplot do atributo *seconds_remaining*

4. Exploração dos dados através de medidas de espalhamento

Uma forma de aprofundar as informações para além das fornecidas pela aplicação das medidas de localidade, que analisa como os dados estão divididos, é analisar a dispersão entre os dados de um atributo. Para isso utilizam-se alguns parâmetros que auxiliam em determinar o quão distante e variado está o conjunto. São estes: variância, desvio padrão e amplitude.

Atributo	Variância	Desvio Padrão	Amplitude	
			min	max
lat	0.007707323	0.08779136	33.2533	34.0883
lon	0.01212742	0.1101246	-118.5198	-118.0218
loc_x	12127.42	110.1246	-250	248
loc_y	7707.323	87.79136	-44	791
shot_distance	87.87543	9.374189	0	79
minutes_remaining	11.90179	3.449897	0	11
seconds_remaining	305.5137	17.47895	0	59

Tabela 4.1

5. Exploração dos dados através de medidas de distribuição

Aplicar medidas de distribuição são essenciais para entender o comportamento geral das classes dos atributos, podendo determinar se corresponde à uma distribuição normal, que implica na aplicação ou não de recursos futuros, como a padronização, por exemplo, que é aplicada na transformação de atributos.

As principais medidas, e que estão aqui presentes, são: curtose e obliquidade. Esta tem como objetivo determinar se há uma tendência para valores acima ou abaixo da média; já aquela, por sua vez, simboliza o achatamento da curva, ou seja, o quanto os valores estão distribuídos próximos à média ou não. Os resultados dos dois parâmetros indicados revelam:

- Obliquidade: {obliquidade = 0, simétrico;
  obliquidade < 0, valores à direita em maior quantidade;
  obliquidade > 0, valores à esquerda em maior quantidade.}
- Curtose: {curtose = 0, distribuição normal dita perfeita;
  curtose > 0, pico mais acentuado e cauda com mais valores;

curtose <0, pico mais achatado e caudas mais leves.}

Abaixo seguem os valores dos parâmetros mencionados, seguidos da plotagem dos histogramas e da curva normal, divididos pelas classes de cada atributo. Com a curva normal fica evidente o parâmetro obliquidade.

Atributo	Curtose	Obliquidade	Histograma	Curva de dist.
lat	1.220452	-0.8156174	5.3 (a)	5.3 (b)
lon	-0.678271	-0.08506946	5.4 (a)	5.4 (b)
loc_x	-0.678271	-0.08506946	5.5 (a)	5.5 (b)
loc_y	1.220452	0.8156174	5.6 (a)	5.6 (b)
shot_distance	0.0441573	0.1054527	5.7 (a)	5.7 (b)
minutes_remaining	-1.163079	0.1985361	5.8 (a)	5.8 (b)
seconds_remaining	-1.185399	0.03100214	5.9 (a)	5.9 (b)
action_type	-	-	5.10	-
combined_shot_type	-	-	5.11	-
game_event_id	-	-	5.12	-
game_id	-	-	5.13	-
playoffs	-	-	5.14	-
season	-	-	5.15	-
shot_type	-	-	5.16	-
shot_zone_area	-	-	5.17	-
shot_zone_basic	-	-	5.18	-
team_id	-	-	5.19	-
team_name	-	-	5.20	-

Atributo	Curtose	Obliquidade	Histograma	Curva de dist.
matchup	-	-	5.21	-
opponent	-	-	5.22	-
shot_id	-	-	5.23	-
period	-	-	5.24	-
shot_zone_range	-	-	5.25	-
game_date	-	-	5.26	-

Tabela 5.1

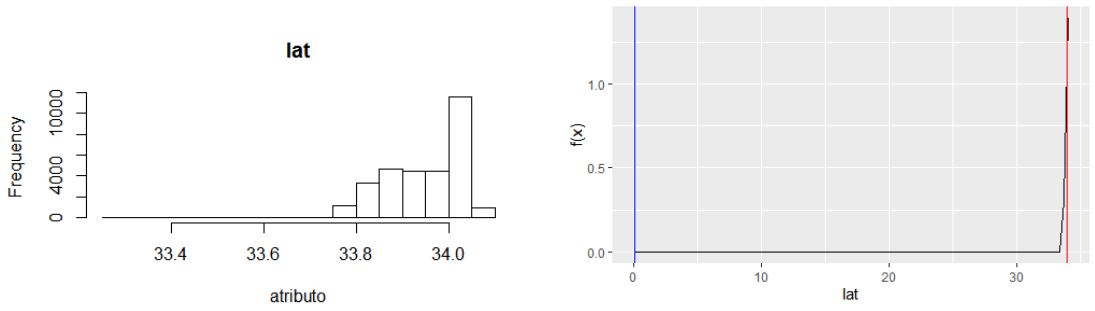
As figuras 5.1 e 5.2 apresentam os códigos das funções utilizadas para a plotagem das imagens seguintes. Nos gráficos que apresentam as medidas de distribuição também estão presentes a média (vermelho) e desvio padrão (azul).

```
curtose <- function(atributo){
  kurtosis(atributo)
}

obliq <- function(atributo){
  skewness(atributo)
}
```

Figura 5.1 - Código para geração da curtose e obliquidade

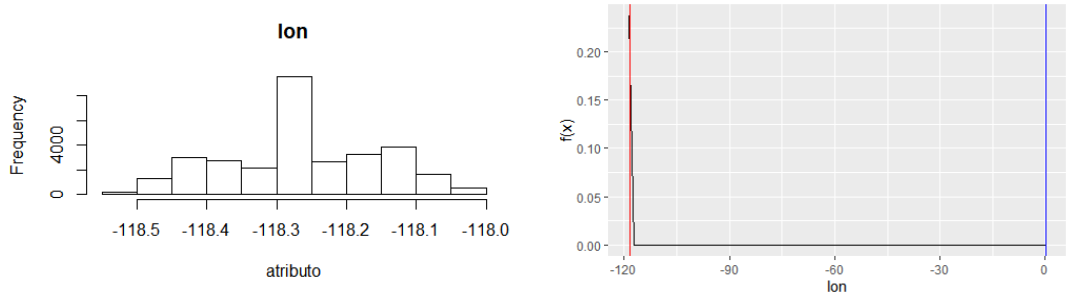
```
#funcao para plotar a distribuição das classes dos atributos com barras
distrib <- function(atributo, eixoXName){
  hist(atributo, main = eixoXName)
}

#função para plotar a curva normal
normal <- function(atributo, eixoXName){
  media <- mean(atributo)
  desvio <- sd(atributo)
  maxim <- max(atributo)
  minim <- min(atributo)
  ggplot(data = data.frame(x = c(minim,
                                 maxim)), aes(x)) +
    stat_function(fun = dnorm,
                  args = list(mean = media,
                              sd = desvio)) +
    labs(y = "f(x)", x = eixoXName) +
    geom_vline(xintercept = media, color = "red") +
    labs(x = eixoXName) +
    geom_vline(xintercept = desvio, color = "blue")
}
```

Figura 5.2 - Código com as funções para gerar respectivamente o histograma e a curva normal demarcada pela média e desvio



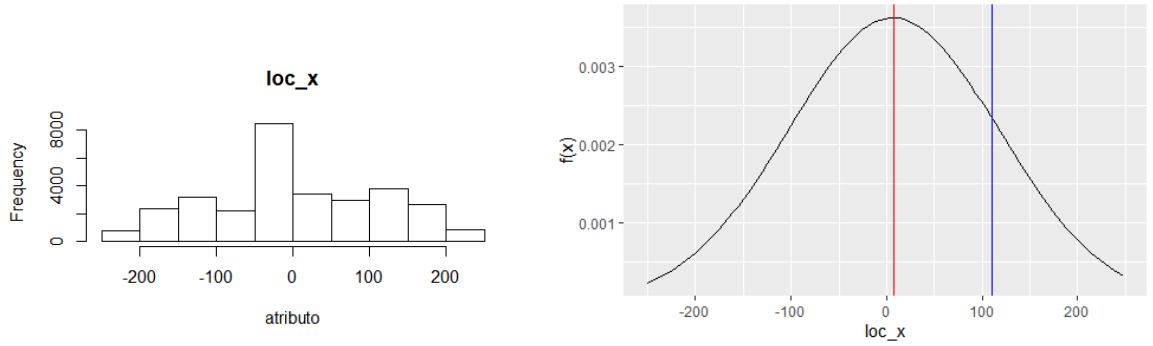
(a) (b)

Figura 5.3 - Atributo *lat*



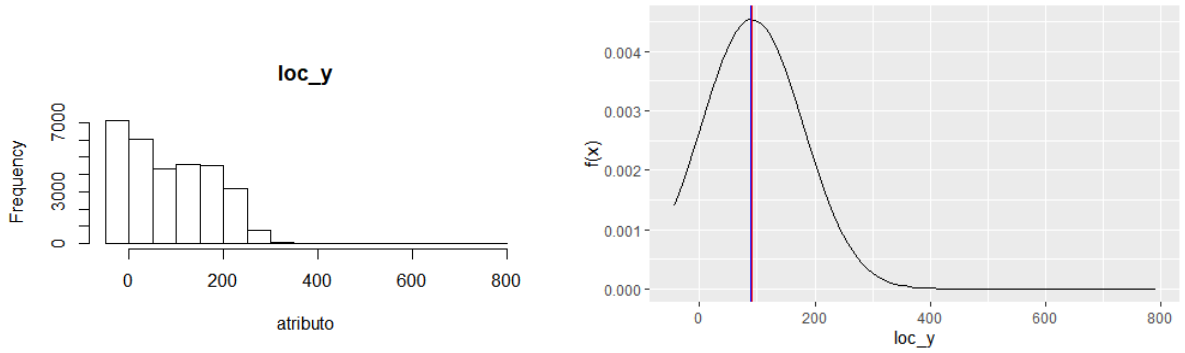
(a) (b)

Figura 5.4 - Atributo *lon*



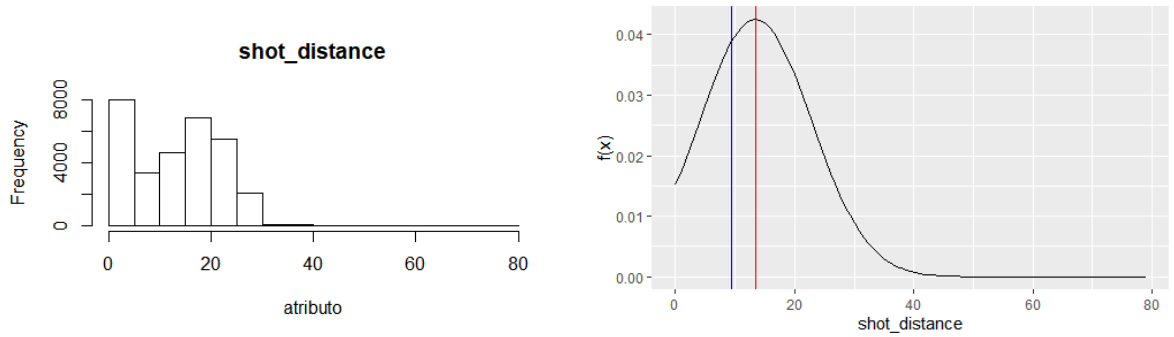
(a) (b)

Figura 5.5 - Atributo *loc_x*



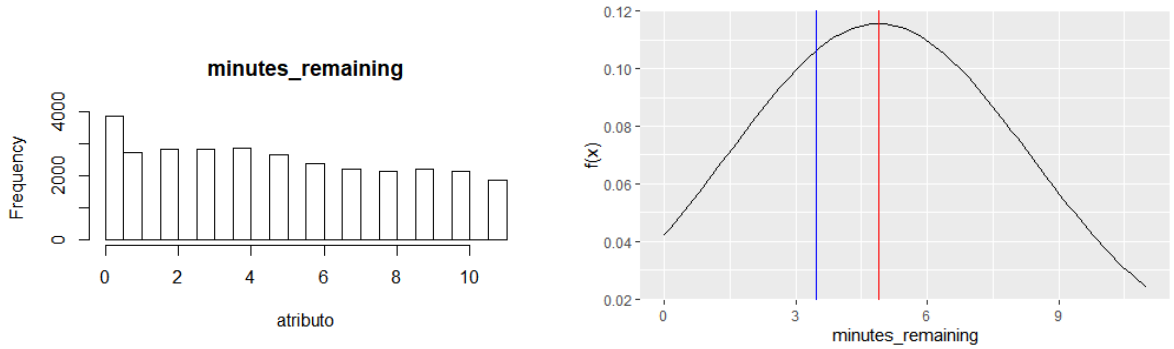
(a) **(b)**

Figura 5.6 - Atributo *loc_y*



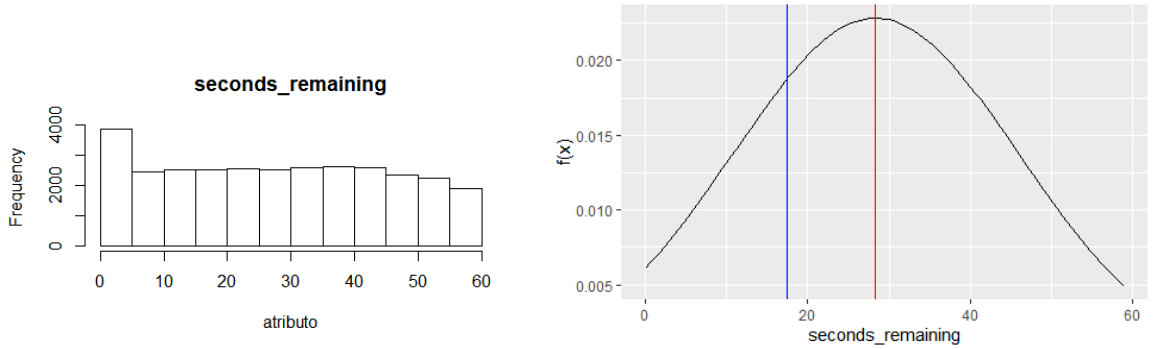
(a) **(b)**

Figura 5.7 - Atributo *shot_distance*



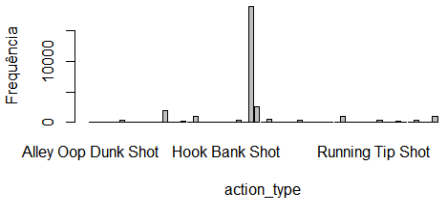
(a) **(b)**

Figura 5.8 - Atributo *minutes_remaning*



(a) **(b)**

Figura 5.9 - Atributo *seconds_remaning*




```
                                  Alley oop dunk shot                                       Alley Oop Layup shot
                                                  122                                                         80
                    Cutting Finger Roll Layup Shot                                            Cutting Layup Shot
                                                    1                                                          6
                                  Driving Bank Shot                                         Driving Dunk Shot
                                                    5                                                        310
                    Driving Finger Roll Layup Shot                                      Driving Finger Roll Shot
                                                   69                                                         82
                 Driving Floating Bank Jump Shot                                     Driving Floating Jump Shot
                                                    1                                                          5
                                  Driving Hook Shot                                         Driving Jump Shot
                                                   14                                                         28
                                 Driving Layup Shot                                  Driving Reverse Layup Shot
                                                 1978                                                         97
                           Driving Slam Dunk Shot                                                    Dunk Shot
                                                   48                                                        262
                              Fadeaway Bank Shot                                          Fadeaway Jump Shot
                                                   31                                                       1048
                             Finger Roll Layup Shot                                               Finger Roll Shot
                                                   33                                                         28
                             Floating Jump Shot                                          Follow Up Dunk Shot
                                                  114                                                         15
                                  Hook Bank Shot                                                    Hook Shot
                                                    5                                                         84
                                  Jump Bank Shot                                                Jump Hook Shot
                                                  333                                                         24
                                       Jump Shot                                                     Layup Shot
                                                18880                                                       2567
                                 Pullup Bank Shot                                              Pullup Jump Shot
                                                   12                                                        476
                                 Putback Dunk Shot                                           Putback Layup Shot
                                                    5                                                         15
                            Putback Slam Dunk Shot                                             Reverse Dunk Shot
                                                    2                                                         75
                                 Reverse Layup Shot                                        Reverse Slam Dunk Shot
                                                  395                                                         16
                                 Running Bank Shot                                           Running Dunk Shot
                                                   48                                                         19
                    Running Finger Roll Layup Shot                                      Running Finger Roll Shot
                                                    6                                                          4
                                 Running Hook Shot                                           Running Jump Shot
                                                   41                                                        926
                                Running Layup Shot                                  Running Pull-up Jump Shot
                                                   72                                                          4
                         Running Reverse Layup Shot                                        Running Slam Dunk Shot
                                                   11                                                          1
                                  Running Tip Shot                                                  Slam Dunk Shot
                                                    2                                                        411
                             Step Back Jump Shot                                                 Tip Layup Shot
                                                  118                                                          2
                                         Tip Shot                                                Turnaround Bank Shot
                                                  182                                                         71
                Turnaround Fadeaway Bank Jump Shot                                  Turnaround Fadeaway Shot
                                                    1                                                        439
                           Turnaround Finger Roll Shot                                         Turnaround Hook Shot
                                                    2                                                         14
                                Turnaround Jump Shot
                                                 1057
```

Figura 5.10 - Atributo *action_tipe*

Bank Shot	Dunk	Hook Shot	Jump Shot	Layup	Tip Shot
141	1286	153	23485	5448	184

Figura 5.11 - Atributo *combined_shot_type*

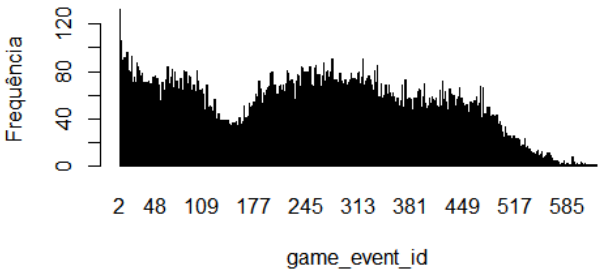


Figura 5.12 - Atributo *game_event_id*

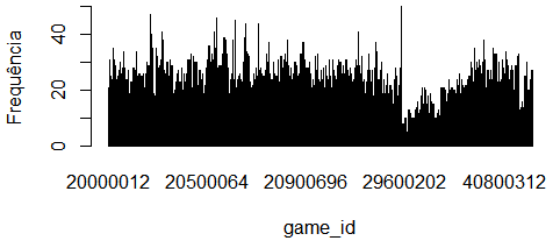


Figura 5.13 - Atributo *game_id*

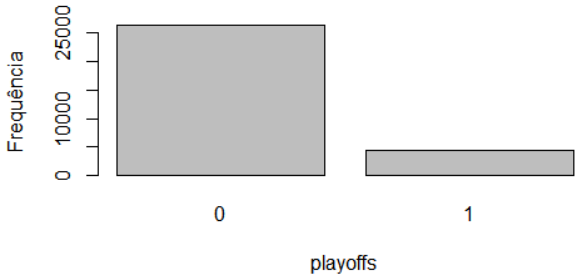
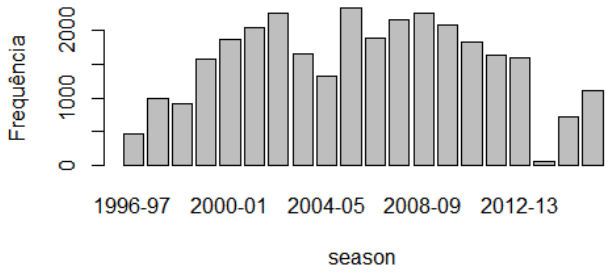


Figura 5.14 - Atributo *playoffs*




```
1996-97 1997-98 1998-99 1999-00 2000-01 2001-02 2002-03 2003-04 2004-05 2005-06 2006-07 2007-08 2008-09
    477     989     921    1577    1868    2028    2241    1638    1324    2318    1887    2153    2242
2009-10 2010-11 2011-12 2012-13 2013-14 2014-15 2015-16
   2080    1825    1637    1593      73     713    1113
```

Figura 5.15 - Atributo *season*

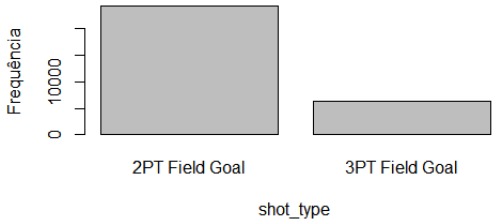


Figura 5.16 - Atributo *shot_type*


```
      Back Court(BC)                Center(C)  Left Side Center(LC)               Left Side(L)
                  83                    13455                  4044                       3751
Right Side Center(RC)          Right Side(R)
                4776                    4588
```

Figura 5.17 - Atributo *shot_zone_area*

Above the Break 3	Backcourt In The Paint (Non-RA)	Left Corner 3
5620	71	4578	280
Mid-Range	Restricted Area	Right Corner 3	
12625	7136	387	

Figura 5.18 - Atributo *shot_zone_basic*

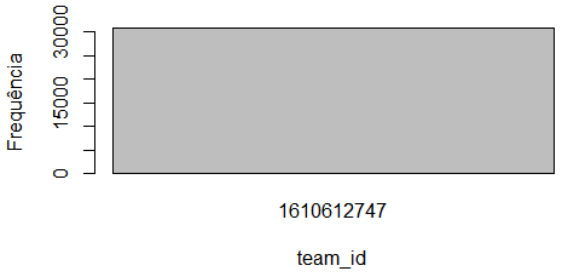


Figura 5.19 - Atributo *time_id*

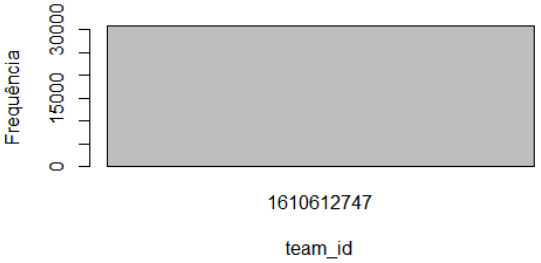
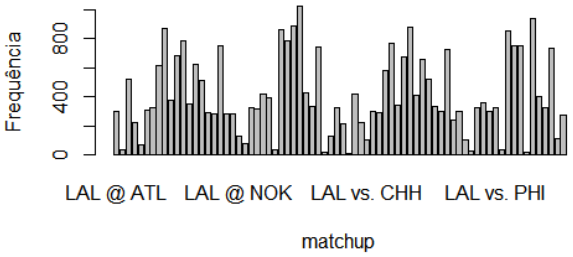
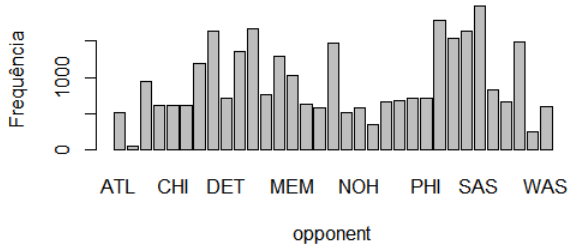


Figura 5.20 - Atributo *team_id*




```
 LAL @ ATL   LAL @ BKN   LAL @ BOS   LAL @ CHA   LAL @ CHH   LAL @ CHI   LAL @ CLE   LAL @ DAL   LAL @ DEN
       301          40         525         223          71         311         323         618         873
 LAL @ DET   LAL @ GSW   LAL @ HOU   LAL @ IND   LAL @ LAC   LAL @ MEM   LAL @ MIA   LAL @ MIL   LAL @ MIN
       373         683         788         352         626         513         294         282         748
 LAL @ NJN   LAL @ NOH   LAL @ NOK   LAL @ NOP   LAL @ NYK   LAL @ OKC   LAL @ ORL   LAL @ PHI   LAL @ PHO
       280         283         128          81         330         319         415         393          33
 LAL @ PHX   LAL @ POR   LAL @ SAC   LAL @ SAS   LAL @ SEA   LAL @ TOR   LAL @ UTA   LAL @ UTH   LAL @ VAN
       859         786         889        1020         427         338         738          21         131
 LAL @ WAS LAL vs. ATL LAL vs. BKN LAL vs. BOS LAL vs. CHA LAL vs. CHH LAL vs. CHI LAL vs. CLE LAL vs. DAL
       326         218          15         421         224         102         299         296         581
LAL vs. DEN LAL vs. DET LAL vs. GSW LAL vs. HOU LAL vs. IND LAL vs. LAC LAL vs. MEM LAL vs. MIA LAL vs. MIL
       769         342         673         878         409         659         517         333         304
LAL vs. MIN LAL vs. NJN LAL vs. NOH LAL vs. NOK LAL vs. NOP LAL vs. NYK LAL vs. OKC LAL vs. ORL LAL vs. PHI
       726         240         298         108          27         327         358         304         327
LAL vs. PHO LAL vs. PHX LAL vs. POR LAL vs. SAC LAL vs. SAN LAL vs. SAS LAL vs. SEA LAL vs. TOR LAL vs. UTA
        40         849         753         754          22         936         401         326         731
LAL vs. VAN LAL vs. WAS
       115         274
```

Figura 5.21 - Atributo *matchup*




```
 ATL  BKN  BOS  CHA  CHI  CLE  DAL  DEN  DET  GSW  HOU  IND  LAC  MEM  MIA  MIL  MIN  NJN  NOH  NOP  NYK
 519   55  946  620  610  619 1199 1642  715 1356 1666  761 1285 1030  627  586 1474  520  581  344  657
 OKC  ORL  PHI  PHX  POR  SAC  SAS  SEA  TOR  UTA  VAN  WAS
 677  719  720 1781 1539 1643 1978  828  664 1490  246  600
```

Figura 5.22 - Atributo *opponent*

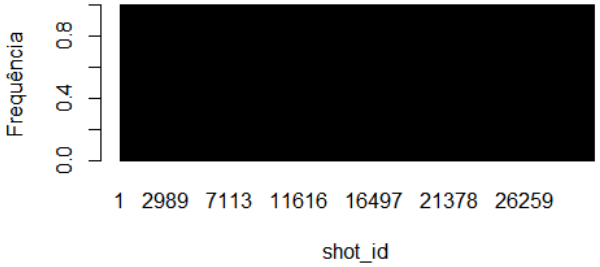


Figura 5.23 - Atributo *shot_id*

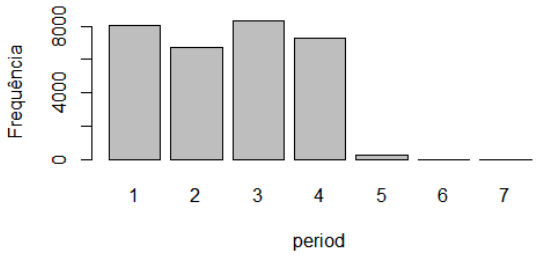
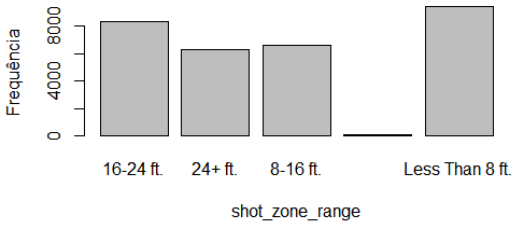


Figura 5.24 - Atributo *period*




```
16-24 ft.      24+ ft.      8-16 ft. Back Court Shot Less Than 8 ft.
      8315         6275         6626              83             9398
```

Figura 5.25 - Atributo *shot_zone_range*

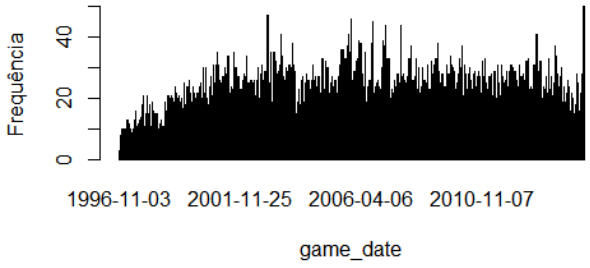


Figura 5.26 - Atributo *game_date*

SEPARAÇÃO

6. Identificação e separação do conjunto de teste

O aprendizado da IA pode ser dividido em três partes essenciais, são elas: treinamento, validação e teste. A separação do conjunto teste do conjunto total de dados se faz durante a etapa aqui descrita, de pré-processamento.

A divisão aqui abordada foi de estabelecer aproximadamente 80% de instâncias para treinamento e validação, e as outras 20% para o teste. A fim de facilitar a divisão, foram separados 6 mil instâncias para o conjunto teste, que representa 19,55% do total, resultando em 24,647 mil instâncias para treinamento.

A figura 6.1 revela o código utilizado para a separação em questão. Primeiramente foi utilizada a função *sample* para selecionar aleatoriamente 6 mil instâncias, que foram armazenadas em *teste*. Após isso, a função *setdiff* foi utilizada para armazenar em *shotsTraining* a diferença entre as instâncias presentes em *shots* e *teste*. A figura 3.2 revela a quantidade final de linhas e colunas em para cada data frame.

```
#selecionando aleatoriamente os casos teste
teste <- shots[sample(nrow(shots), 6000), replace = FALSE]
#separando os casos de treinamento a partir da diferença entre o teste e 'shots'
shotsTraining <- setdiff(shots, teste)
```

Figura 6.1

shots	30697 obs. of 25 variables
shotsTraining	24697 obs. of 25 variables
teste	6000 obs. of 25 variables

Figura 6.2

ELIMINAÇÃO

7. Identificação e eliminação de atributos não necessários

A eliminação de atributos não necessários consiste em retirar aqueles que não contribuem para o aprendizado da IA. Um exemplo bem nítido seria eliminar aqueles que se referem à identificação individual de instâncias, pois para essa

situação, por exemplo, a identificação do arremesso apenas auxilia na organização do data frame, e não possui uma influência direta ou indireta sobre a possibilidade do jogador acertar ou não a cesta.

Vale ressaltar que esse e os demais processos serão aplicados tanto para o conjunto teste quanto para o conjunto de treinamento. De fato, o teste deve sempre ser isolado e não modificado, mas é necessário recordar que o processo aqui descrito é o de pré-processamento, o qual sim é aplicado para ambos os conjuntos.

Com isso, o primeiro passo foi eliminar os atributos de identificação: *game_event_id, game_id, team_id, team_name* e *shot_id*. Após essa alteração os atributos de reduziram a 20.

Outro atributo que após análise revelou-se irrelevante foi o *matchup*. Este expõe um texto que indica o confronto do jogo, entretanto, há também um atributo próprio que indica o oponente, ou seja, um deles pode ser eliminado. Como a descrição está mais clara no atributo *opponent*, o *matchup* será retirado.

A eliminação dos atributos foi realizada atribuindo o valor NULL para a coluna selecionada, como evidenciado no código da figura 7.1. Após esse processo, o conjunto teste e treinamento passaram a possuir 19 colunas.

```
shotsTraining$game_event_id <- NULL
shotsTraining$game_id <- NULL
shotsTraining$team_id <- NULL
shotsTraining$team_name <- NULL
shotsTraining$shot_id <- NULL
shotsTraining$matchup <- NULL

teste$game_event_id <- NULL
teste$game_id <- NULL
teste$team_id <- NULL
teste$team_name <- NULL
teste$shot_id <- NULL
teste$matchup <- NULL
```

Figura 7.1

AMOSTRAGEM

8. Análise e aplicação de técnicas de amostragem de dados

Um conjunto de dados pode possuir as mais diversas informações sobre as mais diversas temáticas, e, com isso, a quantidade de atributos e instâncias varia significativamente, ou não, de uma base de dados para outra. Por isso, técnicas

como a de amostragem são aplicadas principalmente com a finalidade de reduzir a quantidade de instâncias de uma base considerada grande para aquela situação. A base aqui estudada possui 30,697 mil observações, que serão reduzidas após a aplicação da técnica de amostragem.

A amostragem pode ser feita de maneira aleatória, estratificada ou progressiva. Para a base do jogador de basquete Kobe Bryant, objeto de estudo desta análise, será feita uma amostragem simples visando apenas reduzir as instâncias, pois posteriormente será feito o balanceamento que visará uma boa divisão entre as classes do atributo de saída.

O objetivo foi reduzir a quantidade de observações do conjunto de treinamento para 10 mil, que representa 40, 49% do valor atual. Já para o conjunto teste, se aplicarmos a mesma porcentagem, será obtido um valor final de 2,43 mil instâncias.

A figura 8.1 expõe o código que resultou na diminuição do número de linhas do grupo de treinamento e de teste, com resultado final das quantidades revelado na figura 8.2.

```
shotsTraining <- shotsTraining[sample(nrow(shotsTraining), 10000), replace = FALSE]
teste <- teste[sample(nrow(teste), 2430), replace = FALSE]
```

Figura 8.1

shotsTraining	10000 obs. of 19 variables
teste	2430 obs. of 19 variables

Figura 8.2

BALANCEAMENTO

9. Identificação e resolução de problemas de desbalanceamento

Dizer que uma base de dados está desbalanceada é afirmar que a divisão entre as aparições de cada classe do atributo de saída não é equilibrada, e por isso necessita de aplicações de técnicas que irão minimizar este cenário.

Para visualizar como estão distribuídos os dados foi utilizado o código apresentado pela figura 9.1, que gerou as informações exibidas na figura 9.2.

```
table(shotsTraining$shot_made_flag)
table(teste$shot_made_flag)
```

Figura 9.1

```
> table(shotsTraining$shot_made_flag)

   0    1
4657 3725
> table(teste$shot_made_flag)

   0    1
1146  892
```

Figura 9.2

Ao observamos os valores de saída exibidos na figura 9.2, é possível averiguar que:

- O conjunto treinamento possui a classe ‘0’ representando 46,57% e a classe ‘1’ representando aproximadamente 37,25%;
- O conjunto teste tem-se a classe ‘0’ representando aproximadamente 47,16% e a classe ‘1’ representando aproximadamente 36,71%.

Ao analisarmos os valores, à primeira vista parece necessária a aplicação de alguma técnica de balanceamento. Entretanto, se notarmos que o atributo em questão, atributo de saída, possui dados faltantes, é possível calcular qual seria a representatividade das quantidades obtidas em relação à quantidade após a eliminação dos dados faltantes. Por exemplo, o valor 46,57% foi obtido pela divisão da quantidade de aparições pela quantidade total de atributos, mas se dividirmos pelo novo total após a retirada dos valores faltantes será obtido um valor aproximado de 55,56%, como exemplificado pela figura 9.3.

$$\frac{4657}{4657+\ 3725} = 0{,}5556$$

Figura 9.3

O mesmo critério foi aplicado para os demais resultados, obtendo:

- O conjunto treinamento com a classe ‘0’ representando 55,56% e a classe ‘1’ representando aproximadamente 44,44%;

- O conjunto teste com a classe ‘0’ representando aproximadamente 56,23% e a classe ‘1’ representando aproximadamente 43,77%.

Dessa forma, com os valores obtidos e visando a limpeza da base, ambos conjuntos apresentam-se balanceados, com uma divisão aproximada de 56% para 44% entre as classes, o que é considerado aceitável.

LIMPEZA

10. Identificação e eliminação de ruídos ou outliers

Ao analisarmos os gráficos boxplot gerados no tópico 3, é notória a presença de pontos externos ao gráfico, os conhecidos outliers. Estes pontos representam valores que se comportam de maneira discrepante dos demais valores do atributo analisado. Para este caso tem-se a presença de três atributos que possuem dados com essa característica: *lat*, *loc_y* e *shot_distance*.

Como a base possui uma elevada quantidade de dados considerando o contexto aqui exposto, após verificar com a função *boxplot.stats(x)$out* que os pontos em questão representavam em cada um desses atributos cerca de 0,26% do total, foi concluído que retirá-los de cada um dos atributos não seria prejudicial para a base final como um todo.

Para colocar em prática os fatos expostos acima foi desenvolvido o código exposto pela figura 10.1, que exemplifica com o atributo *shot_distance*, mas que também foi aplicado para os demais. Foram seguidos os seguintes passos:

- Inicialmente o gráfico boxplot foi plotado novamente, para verificar o estado após passar pela amostragem;
- Após verificar que de fato os outliers continuam presentes foram utilizadas duas variáveis: *aux* e *aux2*. Esta teve como objetivo armazenar os valores dos outliers e aquela recebeu a base de dados de treinamento, para que as alterações fossem feitas por completo, e depois repassadas à base teste;
- Depois de identificar os valores, o loop *for* foi utilizado para que fossem percorridas todas as posições do auxiliar *aux2* comparando-as com todas as posições da base auxiliar. Ao localizar os valores iguais, o atributo naquela posição de aux recebe o valor ‘-1’;

- Por fim, com o auxílio da função *intersect()*, uma nova base recebe as instâncias que são a intersecção das bases *aux* e a nova base (que possui os valores '-1'). Todas as observações que possuem o valor '-1' serão eliminadas;

Essa aplicação foi feita para todos os três atributos que possuíam esse tipo de dado. Após isso, uma nova base de treinamento e teste foi gerada, eliminando os dados ruidosos.

```
boxplot(shotsTraining$shot_distance)

aux <- shotsTraining
aux2 <- boxplot.stats(aux$shot_distance)$out
?seq_along
#for para atribuir -1 para todos os valores que são outiliers no atributo shot_distance
for (i in seq_along(aux$shot_distance)) {
  for (j in seq_along(aux2)) {
    if (aux$shot_distance[i] == aux2[j]){
      aux$shot_distance[i] <- -1
    }
  }
}

#vai retirar as linhas que possuem valor -1 no atributo shot_distance (outliers), atribuir e
#depois provar plotando o bloxpot
retiraOutShotDistance <- intersect(shotsTraining, aux)
boxplot(retiraOutShotDistance$shot_distance)
```

Figura 10.1

As bases após a limpeza foram nomeadas como *shotsLimpeza* para o treinamento, com dimensões 9972 x 19, e *testeLimpeza* para o teste, com dimensões 2425 x 19.

11. Identificação e eliminação de dados inconsistentes

Os diferentes atributos em uma tabela permitem que diferentes saídas sejam percebidas a partir de diferentes combinações dos valores dos atributos. Uma instância *a* que possui em seus atributos os mesmos valores que uma instância *b*, mas resulta em uma saída diferente, é classificada como inconsistente. Retirar dados inconsistentes faz-se necessário pois interfere no aprendizado da IA.

Com o intuito de definir se a base de dados possui ou não esse tipo de dados foi aplicado o código exposto na figura 11.1. A saída, exibida pelas figuras 11.2 (treinamento) e 11.3 (teste), revela que o número de linhas manteve-se o mesmo,

indicando que não houve repetição para os atributos passados (todos exceto o atributo alvo). A função *count()* foi utilizada, e será detalhada no próximo tópico.

```
View(shotsTraining %>% count(action_type, combined_shot_type, lat, loc_x, loc_y,
                             lon, minutes_remaining, period, playoffs, season, seconds_remaining,
                             shot_distance, shot_type, shot_zone_area, shot_zone_basic,
                             shot_zone_range, game_date, opponent, sort = TRUE))

View(teste %>% count(action_type, combined_shot_type, lat, loc_x, loc_y,
                     lon, minutes_remaining, period, playoffs, season, seconds_remaining,
                     shot_distance, shot_type, shot_zone_area, shot_zone_basic,
                     shot_zone_range, game_date, opponent, sort = TRUE))
```

Figura 11.1

on	seconds_remaining	shot_distance	shot_type	shot_zone_area	shot_zone_basic	shot_zone_range	game_date	opponent	n
1-12	9	1	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2012-01-10	PHX	1
3-04	58	1	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2003-11-06	SAS	1
3-09	59	1	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2009-02-18	GSW	1
0-11	5	3	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2010-12-08	LAC	1
7-08	7	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2008-06-05	BOS	1
5-16	36	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2016-01-07	SAC	1
7-08	30	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2008-03-21	SEA	1
1-02	36	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2001-10-30	POR	1
2-03	1	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2003-04-06	PHX	1
3-09	7	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2009-01-11	MIA	1
2-03	10	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2002-12-03	MEM	1
0-01	21	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2001-03-09	SAS	1

Showing 1 to 12 of 10,000 entries, 19 total columns

Figura 11.2

n	seconds_remaining	shot_distance	shot_type	shot_zone_area	shot_zone_basic	shot_zone_range	game_date	opponent	n
11	27	2	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2011-01-09	NYK	1
12	33	2	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2011-12-26	SAC	1
12	46	1	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2012-04-22	OKC	1
11	5	2	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2010-11-03	SAC	1
12	37	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2012-01-20	ORL	1
05	6	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2005-03-17	MIA	1
06	16	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2006-03-12	SEA	1
01	44	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2001-02-07	PHX	1
02	41	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2002-04-15	SEA	1
03	13	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2002-12-20	PHI	1
08	34	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2008-01-29	NYK	1
04	30	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2003-12-25	HOU	1

Showing 1 to 12 of 2,430 entries, 19 total columns

Figura 11.3

12. Identificação e eliminação de dados redundantes

Os dados redundantes representam aquelas instâncias que se repetem, ou seja, observações que possuem todos os valores de todos os atributos iguais, não representando uma nova possibilidade de aprendizado para a IA.

Para visualizar a presença destes dados redundantes foi utilizada a função *count()*, que recebe como parâmetro os atributos e gera, na última coluna, a quantidade de repetições com os atributos selecionados combinados. A figura 12.2 apresenta um exemplo de saída produzido pelo código apresentado pela figura 12.1, após a combinação dos atributos *action_type* e *combined_shot_tipe*, que resultou em 50 linhas (que indicam 50 combinações diferentes) e com a última coluna indicando a quantidade de cada combinação.

```
View(shotsTraining %>%
  count(action_type, combined_shot_type, sort = TRUE))
```

Figura 12.1

```
# A tibble: 52 x 3
   action_type              combined_shot_type     n
   <chr>                    <chr>              <int>
 1 Jump Shot                Jump Shot           6191
 2 Layup Shot               Layup                842
 3 Driving Layup Shot       Layup                617
 4 Turnaround Jump Shot     Jump Shot            339
 5 Fadeaway Jump Shot       Jump Shot            323
 6 Running Jump Shot        Jump Shot            323
 7 Turnaround Fadeaway shot Jump Shot            151
 8 Pullup Jump shot         Jump Shot            143
 9 Jump Bank Shot           Jump Shot            118
10 Slam Dunk Shot           Dunk                 117
# ... with 42 more rows
```

Figura 12.2

O código da figura 12.3 foi aplicado para todo o conjunto de dados de treinamento e teste. Ao analisar-se as saídas apresentadas pelas figuras 12.4 e 12.5, respectivamente treinamento e teste, foi possível ver que o número de linhas é o mesmo que o do conjunto original, e que na coluna *n* todos os valores são iguais a ‘1’, revelando então não haver dados redundantes nestes.

```
View(shotsTraining %>% count(action_type, combined_shot_type, lat, loc_x, loc_y,
                      lon, minutes_remaining, period, playoffs, season, seconds_remaining,
                      shot_distance, shot_type, shot_zone_area, shot_zone_basic,
                      shot_zone_range, game_date, opponent, shot_made_flag, sort = TRUE))

View(teste %>% count(action_type, combined_shot_type, lat, loc_x, loc_y,
                            lon, minutes_remaining, period, playoffs, season, seconds_remaining,
                            shot_distance, shot_type, shot_zone_area, shot_zone_basic,
                            shot_zone_range, game_date, opponent, shot_made_flag, sort = TRUE))
```

Figura 12.3

shot_distance	shot_type	shot_zone_area	shot_zone_basic	shot_zone_range	game_date	opponent	shot_made_flag	n
16	2PT Field Goal	Left Side(L)	Mid-Range	16-24 ft.	2002-11-15	GSW	1	1
12	2PT Field Goal	Right Side(R)	Mid-Range	8-16 ft.	2002-12-29	TOR	0	1
15	2PT Field Goal	Right Side(R)	Mid-Range	8-16 ft.	2002-12-19	NJN	0	1
16	2PT Field Goal	Right Side(R)	Mid-Range	16-24 ft.	2007-04-12	LAC	NA	1
16	2PT Field Goal	Right Side(R)	Mid-Range	16-24 ft.	2005-02-15	UTA	1	1
15	2PT Field Goal	Right Side(R)	Mid-Range	8-16 ft.	2008-03-24	GSW	0	1
16	2PT Field Goal	Right Side(R)	Mid-Range	16-24 ft.	2010-05-04	UTA	0	1
15	2PT Field Goal	Right Side(R)	Mid-Range	8-16 ft.	2002-03-29	POR	0	1
14	2PT Field Goal	Left Side(L)	Mid-Range	8-16 ft.	2015-12-22	DEN	NA	1
18	2PT Field Goal	Left Side(L)	Mid-Range	16-24 ft.	2002-01-30	ORL	1	1
15	2PT Field Goal	Left Side(L)	Mid-Range	8-16 ft.	2009-02-02	NYK	0	1
16	2PT Field Goal	Right Side(R)	Mid-Range	16-24 ft.	2001-06-10	PHI	1	1
19	2PT Field Goal	Right Side(R)	Mid-Range	16-24 ft.	2002-01-30	ORL	0	1
9	2PT Field Goal	Left Side(L)	Mid-Range	8-16 ft.	2003-11-19	NYK	1	1
13	2PT Field Goal	Right Side(R)	Mid-Range	8-16 ft.	2005-02-28	NYK	0	1
16	2PT Field Goal	Right Side(R)	Mid-Range	16-24 ft.	2002-11-08	WAS	NA	1
14	2PT Field Goal	Right Side(R)	Mid-Range	8-16 ft.	2003-02-06	NYK	1	1

Showing 9,983 to 10,000 of 10,000 entries, 20 total columns

Figura 12.4

ining	shot_distance	shot_type	shot_zone_area	shot_zone_basic	shot_zone_range	game_date	opponent	shot_made_flag	n
27	2	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2011-01-09	NYK	1	1
33	2	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2011-12-26	SAC	1	1
46	1	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2012-04-22	OKC	1	1
5	2	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2010-11-03	SAC	1	1
37	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2012-01-20	ORL	0	1
6	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2005-03-17	MIA	1	1
16	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2006-03-12	SEA	1	1
44	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2001-02-07	PHX	1	1
41	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2002-04-15	SEA	1	1
13	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2002-12-20	PHI	1	1
34	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2008-01-29	NYK	1	1
30	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2003-12-25	HOU	NA	1
10	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2004-02-22	PHX	1	1
9	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2008-11-23	SAC	0	1
29	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2001-10-30	POR	NA	1
49	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2009-02-10	OKC	1	1
21	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2003-12-09	NYK	1	1
41	0	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2005-11-09	MIN	1	1
56	1	2PT Field Goal	Center(C)	Restricted Area	Less Than 8 ft.	2013-01-15	MIL	NA	1

Showing 1 to 19 of 2,430 entries, 20 total columns

Figura 12.5

13. Identificação e resolução de dados incompletos

Após aplicar as técnicas de limpeza apresentadas anteriormente, restou apenas retirar da base os dados faltantes para então concluir a etapa de limpeza.

Como as instâncias que apresentam os valores faltantes representam um número pouco significativo, optou-se por retirar as observações sinalizadas com *NA* (presente somente no atributo de saída), através da função *na.omit(x)*, como mostrado na figura 13.1. O resultado foram duas bases com dimensões:

- *shotsLimpeza* (treinamento): 8358 x 19
- *testeLimpeza* (teste): 2034 x 19

```
shotsLimpeza <- na.omit(shotsLimpeza)
testeLimpeza <- na.omit(testeLimpeza)
```

Figura 13.1

CONVERSÃO

14. Conversão de tipos

A conversão dos dados foi feita seguindo o padrão sugerido de:

- Simbólico para numérico;
- Ordinal para numérico;
- Nominal para binário;
- Numérico para ordinal.

Durante o desenvolvimento e aplicação, houve uma certa limitação em encontrar funções específicas, como nos outros tópicos, para realizar as conversões necessárias. Por isso, para essa etapa a conversão foi realizada de maneira “bruta”, ou seja, com o auxílio de *ifelse* e *for*.

Além disso, notou-se também uma redundância em manter o atributo *game_date*, pois o ano também estava registrado no atributo *season*, e por isso ele foi retirado.

15. Normalização dos dados

Aplicar técnicas de normalização como a reescala e padronização envolve conhecer a curva normal produzida pelos valores dos atributos. Como esses gráficos foram gerados no tópico 5, apenas foi feita a análise para determinar a aplicação.

Ao observar as curvas é possível ver que somente o atributo *loc_x* poderia receber a padronização, por apresentar uma curva normal. Como este está diretamente relacionado ao atributo *loc_y*, optou-se por aplicar somente a reescala para todos os atributos numéricos. A figura 15.1 apresenta o código das funções utilizadas para reescala e padronização.

```
reescala <- function(maximo, minimo, dmax, dmin, atributo){
  return(dmin + ((dmax - dmin)/(maximo - minimo)) * (atributo - minimo))
}

#funcao de padronização
padronizacao <- function(atributo){
  media <- mean(atributo)
  desvioP <- sd(atributo)

  return((atributo - media)/desvioP)
}
```

Figura 15.1

REDUÇÃO

16. Redução de dimensionalidade

Ao mencionar-se a redução da dimensionalidade o fator abordado é da influência dos atributos sobre o atributo alvo, ou seja, qual seria o peso de cada coluna sobre a saída que será produzida.

Para compreender esta situação foi utilizada a Análise de Componentes Principais (PCA)[7], que gera gráficos e informações pertinentes para a determinação necessária. O requisito para aplicação é a limitação dos atributos para os numéricos, e com isso foi utilizado o código da figura 16.1 para essa separação. As colunas selecionadas são: *lat, loc_x, loc_y, lon, minutes_remaining, period, season, seconds_remaining, shot_distance* e *shot_zone_range*, respectivamente.

```
dados_pca <- testeDimensao[,c(3:8, 10:12, 16)]
sapply(dados_pca, sd)
```

Figura 16.1

Após a aplicação de geradores de gráficos e códigos foram geradas as figuras 16.2, 16.3 e 16.4, que apresentam uma análise da influência, representação e contribuição de cada um dos atributos.

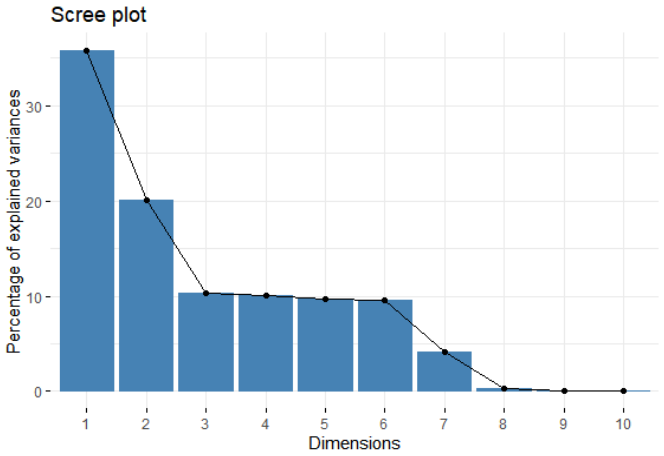


Figura 16.2

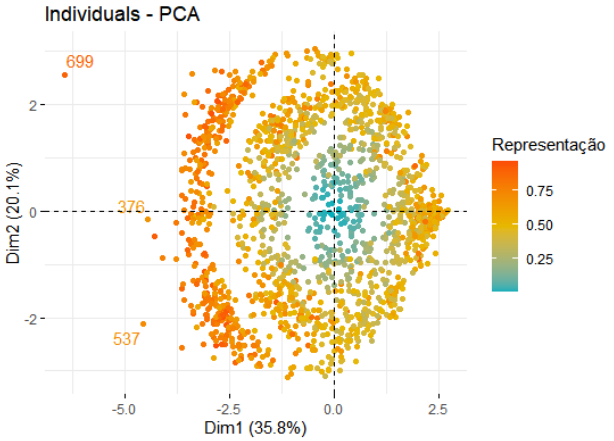


Figura 16.3

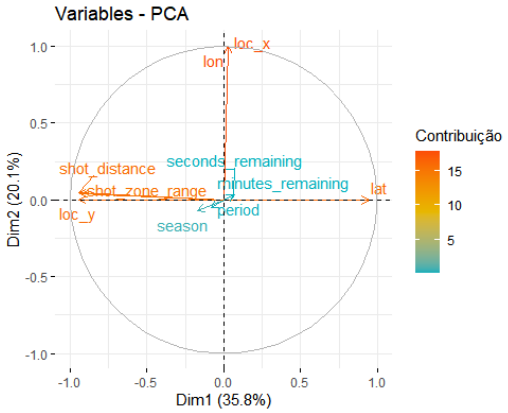


Figura 16.4

Todas as figuras aqui geradas pelo PCA têm como objetivo permitir visualizar os resultados dos cálculos feitos internamente, entretanto o que será utilizado e incrementado na base de dados são os valores gerados pelo PCA. A figura 16.5 expõe o código utilizado, e nele as principais linhas foram sinalizadas.

```
dados_pca <- testeDimensao[,c(3:8, 10:12, 16)]
sapply(dados_pca, sd)

pca_cov <- prcomp(dados_pca)
summary(pca_cov)

pca_corr <- prcomp(dados_pca, center = TRUE, scale = TRUE, rank. = 2)
summary(pca_corr)

fviz_eig(pca_corr)

summary(pca_corr)$rotation
summary(pca_corr)$x

fviz_pca_ind(pca_corr,
             col.ind = "cos2", #Cor pela qualidade de representação
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE, # Texto não sobreposto
             legend.title = "Representação"
)

fviz_pca_var(pca_corr,
             col.var = "contrib", # Cor por contribuições para o PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE,
             legend.title = "Contribuição"
)

testeDimensao[,c(3:8, 10:12, 16)] <- NULL
testeDimensao <- testeDimensao %>%
  mutate(testeDimensao, summary(pca_corr)$x[,1])
testeDimensao <- testeDimensao %>%
  mutate(testeDimensao, summary(pca_corr)$x[,2])
```

Figura 16.5

Nas linhas destacadas estão, respectivamente:

- A aplicação do PCA passando como parâmetro um *rank* igual a 2, que indica o número de colunas que serão geradas;
- O resultado da atribuição;
- Retirada dos atributos numéricos que serão agora substituídos pelos novos valores de PCA. Essa etapa reduzirá a base de 18 para 8 atributos;
- Acrescentar na base o primeiro PCA, que elevará para 9 o número de colunas;
- Acrescentar na base o segundo PCA, que fechará a base em um total de 10 atributos.

Vale lembrar que os passos aqui apresentados para a base teste também foram aplicados para o conjunto de treinamento, e ambas finalizaram com as 10 colunas.

AMBIENTE

Para aplicação dos códigos aqui apresentados é necessária a instalação das seguintes bibliotecas:

- *tidyverse*, para visualização dos gráficos com o comando *ggplot*;
- readr, para leitura da base de dados .csv;
- *dplyr*, para utilizar ferramentas de seleção;
- *tidyselect*, para utilizar ferramentas de incremento e reorganização do data frame;
- *e1071*, para utilizar as funções de cálculo da obliquidade e curtose;
- *base*, para funções básicas;
- *factoextra*, para gerar os gráficos criados pela PCA.

CONCLUSÃO

A base de dados final passou por diversas transformações com o intuito de prepará-la para posteriormente receber um algoritmo que será a base para o

aprendizado da IA. Como resultado do processo de pré-processamento foram obtidas duas bases: *dataTreino* (8358 x 10) para treinamento e *dataTeste* (2034 x 10) para teste.

Pode-se dizer que após toda aplicação os dados estão prontos para receberem os próximos passos, que envolvem o treinamento e de fato o aprendizado de máquina.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] WICKHAM, Hadley; GROLEMUND, Garrett. R para Data Science: Importe, Arrume, Transforme, Visualize e Modele Dados. Alta Books Editora, 2019.

[2] Estatísticas Básicas: Medidas de Precisão. Disponível em: <https://rodriguesloures.com/files/html/estat%C3%ADsticas-b%C3%A1sicas.html>. Acesso em 4 jul. 2022.

[3] Manipulando Dados com dplyr e tidyr. Disponível em: <https://www.ufrgs.br/wiki-r/index.php?title=Manipulando_Dados_com_dplyr_e_tidyr>. Acesso em 5 jul. 2022.

[4] Set operations. Disponível em: <https://dplyr.tidyverse.org/reference/setops.html>. Acesso em 5 jul. 2022.

[5] Count observations by group. Disponível em: <https://dplyr.tidyverse.org/reference/count.html> Acesso em 6 jul. 2022.

[6] Interpretar todas as estatísticas e gráficos para Sumário gráfico. Disponível em: <https://support.minitab.com/pt-br/minitab/18/help-and-how-to/statistics/basic-statistics/how-to/graphical-summary/interpret-the-results/all-statistics-and-graphs/>. Acesso em 5 jul. 2022.

[7] ANÁLISE DE COMPONENTES PRINCIPAIS (PCA): CÁLCULO E APLICAÇÃO NO R. Disponível em: <https://operdata.com.br/blog/analise-de-componentes-principais-pca-calculo-e-aplic

acao-no-r/#:~:text=An%C3%A1lise%20de%20Componentes%20Principais%20(PCA)%3A%20c%C3%A1lculo%20e%20aplica%C3%A7%C3%A3o%20no%20R,-Blog%2C%20Data%20Science&text=A%20PCA%20%C3%A9%20um%20m%C3%A9todo,dados%20com%20vari%C3%A1veis%20possivelmente%20correlacionadas.>. Acesso em 6 jul. 2022.