



Universidad San Carlos de Guatemala

Escuela de Estudios de Postgrado de Ingeniería

Introducción a la minería de datos

Estudiante: Jorge David Ambrocio Ventura

Carnet: 999013655

INFORME DE PROCESAMIENTO PARA DATOS FIFA 2016-2020

Resumen

Se ejecuta el proceso Feature engineering sobre los conjuntos de datos de jugadores encontrados en FIFA archivos del año 2016 al año 2020. El proceso es realizado en lenguaje R mediante la herramienta RStudio, iniciando con la carga independiente de los archivos, explorando la distribución de los mismos para consolidar la información en un único dataset y así poder realizar limpieza de datos y estandarización de los mismos. Durante este proceso se eliminan valores NA, se normalizan datos y se crean nuevas características que serán de utilidad para la ejecución de algoritmos.

Librerías utilizadas

Para el desarrollo del proyecto fueron utilizadas las siguientes librerías:

- Dplyr: utilizada para la manipulación de los data set mediante tuberías de análisis
- TidyR: utilizada para la gestión de agrupaciones y traspuesta de características
- Arules: utilizada para ejecutar algoritmos de minería de datos
- Stringr: utilizada para facilitar la manipulación de formato características mediante expresiones regulares

```
# Obtención de librerías
```

```
```{r}  
library(dplyr)
library(ggplot2)
library(arules)
library(rpart)
library(rpart.plot)
library(tidyr)
library(stringr)
```
```

Carga de archivos

Los archivos son cargados al entorno de memoria de R mediante la instrucción `read.csv`, se carga cada archivo csv en una variable distinta con el formato `dPlayer_xx` donde `xx` representa el año al que pertenece el conjunto de datos.

```
# Obtención de conjuntos de datos
```

```
```{r}  

dPlayers_15 <- read.csv("players_15.csv")
dPlayers_16 <- read.csv("players_16.csv")
dPlayers_17 <- read.csv("players_17.csv")
dPlayers_18 <- read.csv("players_18.csv")
dPlayers_19 <- read.csv("players_19.csv")
dPlayers_20 <- read.csv("players_20.csv")

```
```

Análisis de columnas

Se realiza lectura e indagación del data set mediante la instrucción `summary`, de igual forma se evalúa el tamaño de los dataset y se compara que todos cuenten con las mismas columnas para poder consolidarlos sin previa manipulación. Se realiza esta comparación mediante la función siguiente: `all.equal(names(dPlayers_15), names(dPlayers_16))` que permite obtener la comparación del nombre de las columnas en orden de dos dataset, esta instrucción se repite para comparar todos los dataset.

```

5 {r}
7 # Cantidad de columnas
8 print("Cantidad de columnas dPlayers 15, 16, 17, 18, 19, 20")
9 print(ncol(dPlayers_15))
10 print(ncol(dPlayers_16))
11 print(ncol(dPlayers_17))
12 print(ncol(dPlayers_18))
13 print(ncol(dPlayers_19))
14 print(ncol(dPlayers_20))
15
16 # Compara que los nombres de las columnas sean iguales
17 print("Comparación de nombres de columnas entre datasets:")
18
19 print(all.equal(names(dPlayers_15), names(dPlayers_16)))
20 print(all.equal(names(dPlayers_16), names(dPlayers_17)))
21 print(all.equal(names(dPlayers_17), names(dPlayers_18)))
22 print(all.equal(names(dPlayers_18), names(dPlayers_19)))
23 print(all.equal(names(dPlayers_19), names(dPlayers_20)))
24
25 # Cantidad de filas
26 print("Cantidad de filas dPlayers 15, 16, 17, 18, 19 y 20")
27 print(nrow(dPlayers_15))
28 print(nrow(dPlayers_16))
29 print(nrow(dPlayers_17))
30 print(nrow(dPlayers_18))
31 print(nrow(dPlayers_19))
32 print(nrow(dPlayers_20))
33
34 }

```

Unificación de conjuntos

Al comprobar la idoneidad de los conjuntos de datos, se añade a cada conjunto de datos el año al que pertenecen y se unifican mediante la función `rbind` nativa de R. Esto con el objetivo de poderlos identificar más adelante mediante el año del que se desprende cada una de las observaciones en el conjunto.

```

# Data Cleaning
## Unificación de los conjuntos de datos
```{r}
Añadir la columna año a cada dataset
dPlayers_15 <- dPlayers_15 %>%
 mutate(año = 2015)

dPlayers_16 <- dPlayers_16 %>%
 mutate(año = 2016)

dPlayers_17 <- dPlayers_17 %>%
 mutate(año = 2017)

dPlayers_18 <- dPlayers_18 %>%
 mutate(año = 2018)

dPlayers_19 <- dPlayers_19 %>%
 mutate(año = 2019)

dPlayers_20 <- dPlayers_20 %>%
 mutate(año = 2020)

dPlayers <- rbind(dPlayers_15,dPlayers_16, dPlayers_17, dPlayers_18, dPlayers_19, dPlayers_20)
head(dPlayers)
```

```

Eliminación de NA

En este apartado se inicia con el conteo de observaciones con valor NA por cada una de las características del conjunto, luego esta cantidad es dividida por la cantidad total de observaciones para obtener un porcentaje de faltantes. Se ordena este porcentaje para identificar los peores casos en las características, todas las características que superan el 80% de valores faltantes, son eliminadas del conjunto y todas aquellas que se encuentran en el 11% de valores faltantes son rellenados con el valor mediana de la característica a la que pertenecen.

```

9 ## Obtener cantidad de celdas nulas por columna
10 ```{r}
11 cnt_filas <- nrow(dPlayers)
12
13 # conteo de celdas con valores NA por cada columna
14 na_counts <- dPlayers %>%
15   summarise(across(everything(), ~ sum(is.na(.)))) %>%
16   pivot_longer(cols = everything(), names_to = "Column", values_to = "NA_Count") %>%
17   mutate(PERCENT = NA_Count / cnt_filas ) %>%
18   arrange(desc(PERCENT))
19
20 na_counts
21
22
23
24
25

```

A tibble: 105 x 3

| Column
<chr> | NA_Count
<int> | PERCENT
<dbl> |
|----------------------|-------------------|------------------|
| shooting | 11247 | 0.11136195 |
| passing | 11247 | 0.11136195 |
| dribbling | 11247 | 0.11136195 |
| defending | 11247 | 0.11136195 |
| physic | 11247 | 0.11136195 |
| contract_valid_until | 1333 | 0.01319867 |
| team_jersey_number | 1326 | 0.01312936 |
| sofifa_id | 0 | 0.00000000 |
| player_url | 0 | 0.00000000 |

```

### Omitir columnas con más del 80% de valores faltantes
```{r}
Ignorar las columnas que tienen más de 80% de valores faltantes
dPlayers <- dPlayers %>%
 select(-nation_jersey_number, - gk_diving, -gk_handling, -gk_kicking, - gk_reflexes, -gk_speed,
 -gk_positioning)

Para las columnas con 11% de faltantes, se rellenan con el valor mediana de la característica
```{r}
# obtener valor mediana de la columna
m_shooting <- median(dPlayers$shooting, na.rm = TRUE)
m_passing <- median(dPlayers$passing, na.rm = TRUE)
m_dribbling <- median(dPlayers$dribbling, na.rm = TRUE)
m_defending <- median(dPlayers$defending, na.rm = TRUE)
m_physic <- median(dPlayers$physic, na.rm = TRUE)

# Llenar los valores na con el valor mediana de la característica
dPlayers <- dPlayers %>%
  mutate(shooting = ifelse(is.na(shooting),m_shooting, shooting)) %>%
  mutate(passing = ifelse(is.na(passing),m_passing, passing)) %>%
  mutate(dribbling = ifelse(is.na(dribbling),m_dribbling, dribbling)) %>%
  mutate(defending = ifelse(is.na(defending),m_defending, defending)) %>%
  mutate(physic = ifelse(is.na(physic),m_physic, physic))

```

Transformación de tipo de datos de columnas

Se detecta en el conjunto de dato múltiples columnas, todas aquellas dedicadas a la medición de capacidades específicas que se encuentran en valor texto con la forma del siguiente ejemplo “10 + 4” a manera de suma, estos valores se consideran trabajar de mejor

manera como números enteros, por lo que estos valores son separados por medio de expresiones regulares por el signo de adición o el signo de sustracción, los valores obtenidos son casteados a valores numéricos y finalmente son operados mediante adición para obtener el valor numérico final de cada una de estas características.

Para la ejecución de esta transformación se construye una función encargada de realizarla y se construye un array con el nombre de todas las columnas que se requiere que pasen por dicha transformación. Al ejecutarse sobre cada característica se obtiene como resultado que se añaden todas las columnas con valores numéricos y se eliminan las columnas con los valores de tipo texto que contienen el problema inicialmente descrito.

```
## Columnas texto que deberían ser numéricas
```{r}
Construir una función que separa por isngo más o menos, caste ambas partes de la operación
aritmética y luego la ejecuta
transform_numeric <- function(column) {
 base_value <- str_extract(column, "^-?\\d+") %>% as.numeric()
 modifier <- str_extract(column, "[+-]\\d+$") %>% as.numeric()
 result <- base_value + ifelse(is.na(modifier), 0, modifier) # Maneja casos sin modificador
 return(result)
}
...

8
9 dPlayers <- dPlayers %>%
0 mutate(across(all_of(cols_to_transform), transform_numeric, .names = "numeric_{col}"))
1
2 # Mostrar los primeros resultados
3 head(dPlayers)
4
5

Eliminar las columnas que están en formato texto cuando ya se añadieron como número
```{r}
# Eliminar las columnas que fueron transformadas
dPlayers <- dPlayers %>%
  select(-all_of(cols_to_transform))
...

```

Agrupaciones

Se crean características basadas en la agrupación de valores numéricos de tipo entero, específicamente esto se realiza con la edad de los jugadores, clasificando las edades en tres grupos: Debajo de veinte años, entre veinte y 30 y finalmente arriba de treinta años.

Se realiza la misma labor con la característica value_aur, clasificando en bajo para los valores menores a 1000000, medio a los valores comprendidos entre este valor hasta 10000000 y todo lo que está por arriba de este valor se clasifica como alto.

Para poder realizar agrupaciones en futuros análisis se crea la característica nacionalidad & club, permitiendo trabajar en relación de clubes asociados a sus países, favoreciendo el análisis a nivel país.

Se crea la característica para describir la cantidad de años desde que el jugador se unió al club.

```

## Agrupaciones
```{r}

Agrupación de observaciones por rangos de edad
dPlayers <- dPlayers %>%
 mutate(age_group = case_when(
 age < 20 ~ "Under 20",
 age >= 20 & age <= 30 ~ "20-30",
 age > 30 ~ "over 30"
))
```

## Agrupación por rango de valor de ingresos mensuales
dPlayers <- dPlayers %>%
  mutate(value_category = case_when(
    value_eur < 1000000 ~ "Low",
    value_eur >= 1000000 & value_eur < 10000000 ~ "Medium",
    value_eur >= 10000000 ~ "High"
  ))
```

Crear identificador a nivel nacionalidad y club
```{r}
# Crear un identificador único basado en la nacionalidad y club
dPlayers <- dPlayers %>%
  mutate(unique_player_id = paste(nationality, club, sep = "_"))
```

Indicar la cantidad de años que el jugador lleva en el club
```{r}
# Calcular la cantidad de años desde que el jugador se unió a su club
# Convertir 'joined' a tipo fecha y calcular la antigüedad del jugador en el club
dPlayers <- dPlayers %>%
  mutate(joined = as.Date(joined, format = "%Y-%m-%d"),
         years_at_club = as.numeric(format(Sys.Date(), "%Y")) - as.numeric(format(joined, "%Y")))
```

```

Relacionado a las transformaciones de datos se crea la matriz traspuesta de cantidad de jugadores por nacionalidad a través de los años analizados 2016-2020. Esto mediante las instrucciones group by y pivot winder.

```

Data Transformation
Descripción de datos por características específicas
```{r}
# Agrupación de jugadores por nacionalidad con conteo de jugadores ordenado de mayor a menor
nacionalidades <- data.frame(nationality = unique(dPlayers$nationality))

# cantidad de nacionalidades
nrow(nacionalidades)

# Pivot de datos por año
jugadores_nacionalidad_anio <- dPlayers %>%
  group_by(nationality, anio) %>%
  summarise(count = n()) %>%
  ungroup() %>%
  pivot_wider(names_from = anio, values_from = count, values_fill = 0)

jugadores_nacionalidad_anio
```

```

R Console

tbl\_df  
179 x 7

A tibble: 179 x 7

| nationality<br><chr> | 2017<br><int> | 2018<br><int> | 2019<br><int> | 2020<br><int> | 2015<br><int> | 2016<br><int> |
|----------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| Afghanistan          | 1             | 3             | 4             | 2             | 0             | 0             |
| Albania              | 34            | 35            | 39            | 43            | 32            | 31            |
| Algeria              | 47            | 56            | 58            | 50            | 40            | 40            |
| Angola               | 10            | 15            | 14            | 16            | 6             | 11            |
| Antigua & Barbuda    | 6             | 5             | 4             | 7             | 3             | 6             |
| Argentina            | 1075          | 964           | 943           | 886           | 861           | 1057          |
| Armenia              | 9             | 14            | 7             | 8             | 6             | 10            |
| Aruba                | 2             | 0             | 0             | 0             | 0             | 1             |
| Australia            | 195           | 196           | 206           | 196           | 196           | 179           |
| Austria              | 263           | 267           | 299           | 319           | 259           | 252           |

1-10 of 179 rows

Previous 1 2 3 4 5 6 ... 18 Next

Brindando como resultado que el país con mayor cantidad de jugadores es England seguido por German en años recientes y argentina en los años más antiguos.

Se ejecuta el algoritmo KMeans para obtener una clasificación de los jugadores en tres conjuntos y es añadida al conjunto de datos inicial.



```

ejecución de kmeans para categorizar a los jugadores por año
```{r}

md_overall <- median(dPlayers$overall, na.rm = TRUE)
md_pace <- median(dPlayers$pace, na.rm = TRUE)

dPlayers_km <- dPlayers %>%
  select(overall, pace) %>%
  mutate(overall = ifelse(is.na(overall), md_overall, overall))

dPlayers_km <- dPlayers_km %>%
  mutate(pace = ifelse(is.na(pace), md_pace, pace))

clusters <- kmeans(dPlayers_km, centers = 3)

dPlayers <- dPlayers %>%
  mutate(cluster_group = clusters$cluster)
```

```

Finalmente se realiza la normalización de valores para las columnas overall, potential, pace y shooting mediante la fórmula del mínimo sobre el rango, interpretado como la diferencia entre el máximo menos el mínimo.

```

Normalización de las columnas overall, potential, pace y shooting
```{r}

cols_to_normalize <- c("overall", "potential", "pace", "shooting")

dPlayers <- dPlayers %>%
  mutate(across(all_of(cols_to_normalize), ~ (. - min(.)) / (max(.) - min(.))))

head(dPlayers, 10)
```

```

Description: df [10 x 103]

| age_group<br><chr> | value_category<br><chr> | unique_player_id<br><chr>     | years_at_club<br><dbl> | cluster_group<br><int> |
|--------------------|-------------------------|-------------------------------|------------------------|------------------------|
| 20-30              | Low                     | Argentina_FC Barcelona        | 20                     | 3                      |
| 20-30              | Low                     | Portugal_Real Madrid          | 15                     | 3                      |
| 20-30              | Low                     | Netherlands_FC Bayern München | 15                     | 3                      |
| Over 30            | Low                     | Sweden_Paris Saint-Germain    | 12                     | 3                      |
| 20-30              | Low                     | Germany_FC Bayern München     | 13                     | 3                      |
| 20-30              | Low                     | Uruguay_FC Barcelona          | 10                     | 3                      |
| 20-30              | Low                     | Belgium_Chelsea               | 12                     | 3                      |
| 20-30              | Low                     | Netherlands_Manchester United | 12                     | 3                      |
| 20-30              | Low                     | Germany_FC Bayern München     | 22                     | 3                      |
| Over 30            | Low                     | France_FC Bayern München      | 17                     | 3                      |

1-10 of 10 rows | 100-104 of 103 columns