Análisis de la efectividad goleadora de los futbolistas de La Liga con StatsBomb

Autores: Pedro Antonio Carnerero Molina
Víctor González García
David Pastor Sánchez
María Teresa Torres Aguilar

Fecha: 6 de junio de 2025

UNIVERSIDAD
DE GRANADA

Todos los datos han sido sacados de StatsBomb

StatsBomb

Introducción

El objetivo del análisis es calcular el impacto que produce un futbolista en una jugada de gol. Para ello propondremos distintas métricas que calcularemos para establecer un ránking de los jugadores más influyentes.

La idea principal sobre como medir este impacto se fundamenta en comparar la probabilidad estimada de marcar un gol en una jugada concreta, es decir la probabilidad a priori dadas las condiciones del tiro con el resultado real de la acción. Por ejemplo, si un disparo tenía solo un 0.2 de probabilidad de acabar en gol y el jugador logra marcar, podemos interpretar que su intervención fue muy positiva y añadió valor a la jugada.

Naturalmente, no existe una fórmula exacta que mida con total certeza cuánto influye un jugador en cada situación. Sin embargo, gracias a modelos estadísticos construidos con datos históricos, podemos ajustar un modelo probabilístico que evalúe adecuadamente cada jugada y asigne una probabilidad razonable de éxito, lo cual nos permitirá medir la aportación del jugador de forma objetiva y comparativa.

Este proyecto es muy completo, porque requiere habilidades de tratamiento de datos *en crudo*, como comentaremos en un instante. Para nosotros ha sido especialmente interesante ver como la estadística puede usarse para sacar conclusiones usando fenómenos cualitativos. Sin entrar mucho en detalle, en la base de datos la mayoría de variables responden a preguntas del tipo ¿está presionado el jugador? ó ¿qué tipo de jugada es? Córner, libre directo ó jugada abierta, por ejemplo. Es muy enriquecedor e intuitivo ver cómo este contexto influye en la predicción del modelo.

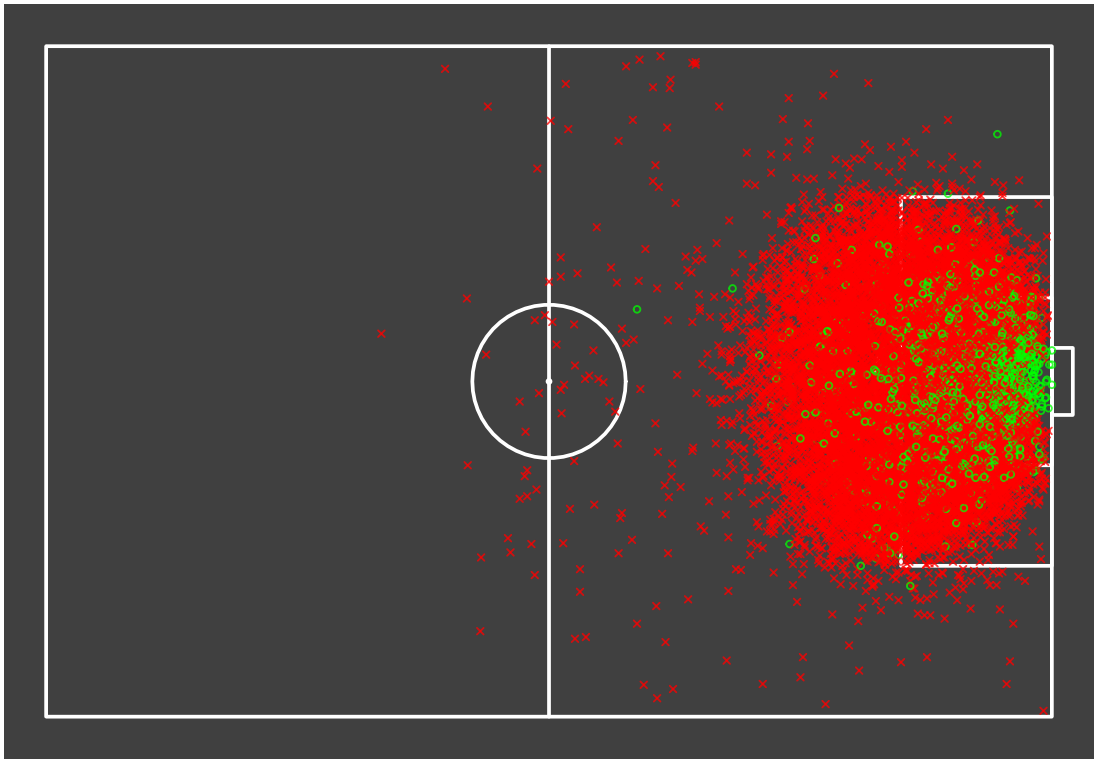
1. Los datos

El conjunto de datos se ha manufacturado usando la librería de StatsBomb para R. StatsBomb es un proveedor de datos que ha creado una variedad de funciones para conectarse a sus servidores. A través de ciertas llamadas se pueden obtener datos de *eventos*, como ellos los llaman. A lo largo de un partido cada evento (tiro, pase, sustitución, falta, etc.) se registra junto con ciertas variables contextuales que permiten saber qué estaba ocurriendo y quiénes estaban involucrados.

Esta parte es complicada y requiere estar atentos para filtrar bien los eventos que nos interesan, pero puede hacerse sin problema siguiendo la documentación que provee la empresa sobre su base de datos y cómo está organizada. Finalmente se obtuvo un conjunto de datos compuesto por 17573 disparos de futbolistas en La Liga a lo largo de 11 temporadas, desde 2010 hasta 2021. Se registraron 11 variables de las cuales `location.x` y `location.y` se desecharan en favor de `distance` lo que nos deja con 9 variables registradas, 8 predictoras y 1 de respuesta:

```
##   is_pressed          from                     player is_open technique
## 1      FALSE From Throw In             Anssumane Fati   FALSE    Normal
## 2      FALSE  Regular Play Edgar Antonio Méndez Ortega   FALSE    Normal
##    body_part situation is_goal distance location.x location.y
## 1 Right Foot Open Play       0 16.55174      108.6         28
## 2 Right Foot Open Play       0 19.74740      103.6         51
```

Claramente `is_goal` es la respuesta que queremos predecir. La razón de añadir `location.x` y `location.y` es poder graficar un mapa de disparos:



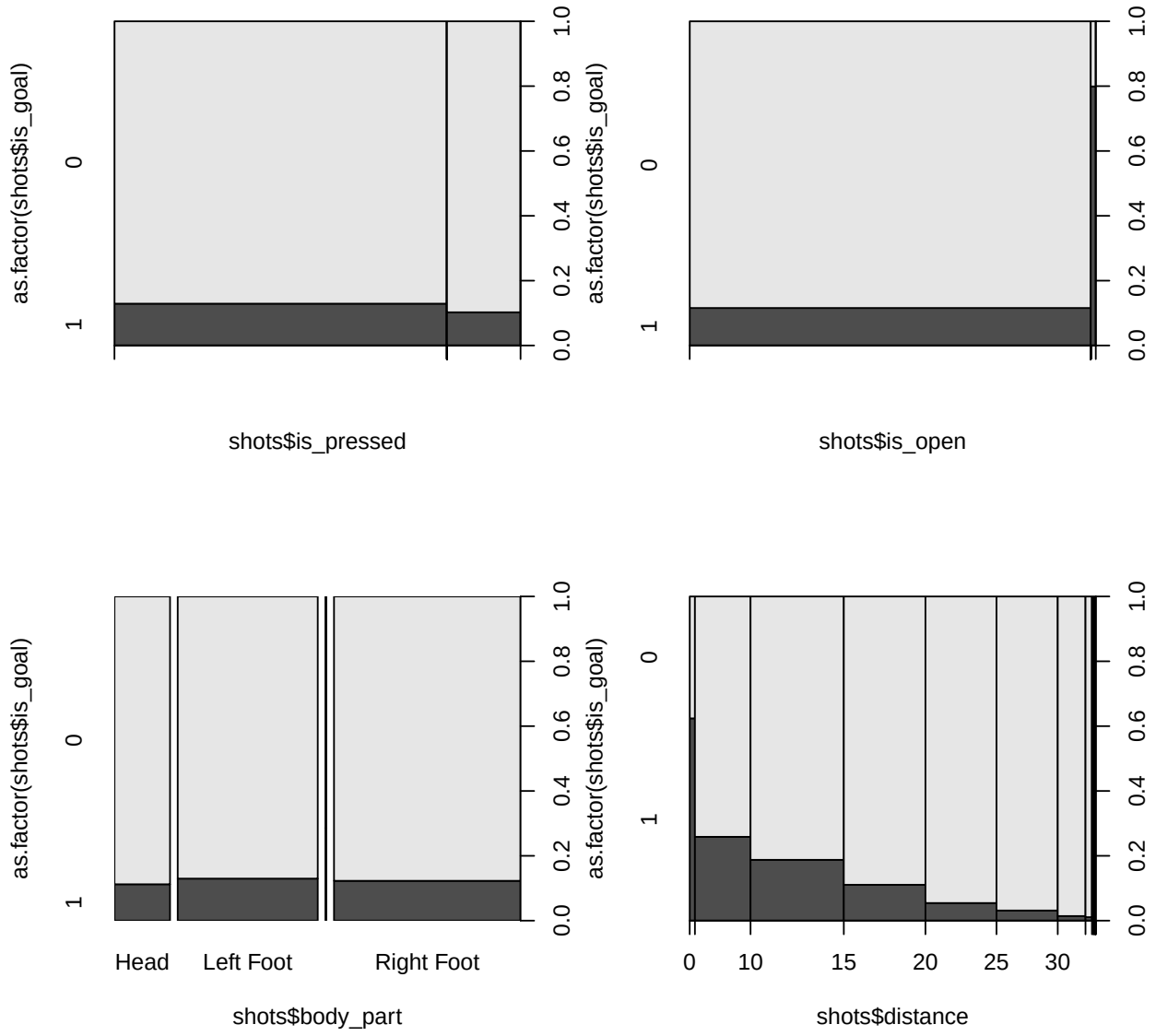
Vamos a seguir explorando los datos. Ya podemos deshacernos de los `location` y dejar `distance`. Además convertiremos `is_goal` en factor también.

Estudiemos las relaciones de las variables del dataset con la respuesta. Con las siguientes gráficas podemos ver como las distintas categorías de cada variable influyen sobre la variable respuesta:

```
par(mfrow=c(2,2))
plot(as.factor(shots$is_goal)~shots$is_pressed)
plot(as.factor(shots$is_goal)~shots$is_open)

plot(as.factor(shots$is_goal)~shots$body_part)
```

```
plot(as.factor(shots$is_goal)~shots$distance)
```



Se aprecia como estar presionado parece aumentar ligeramente la probabilidad de gol. Que no haya portero (`is_open`) se muestra como una variable muy determinante como se puede observar. También se aprecia un aumento de la proporción de goles cuando se reduce la distancia a la portería y tirar con el pie también aumenta ligeramente cuando lo comparamos con los tiros de cabeza. Todo esto hace que el análisis se antoje prometedor.

Observamos también que ciertas variables como `technique` y `situation` tienen muchas categorías para distinguir entre las diversas situaciones. Esto será un problema luego cuando ajustemos el modelo:

```
summary(shots$technique)
```

```
##      Backheel Diving Header   Half Volley           Lob        Normal
##             62            69          2335           281         13576
## Overhead Kick        Volley
##           112          1138
```

Si seguimos explorando los datos nos daremos cuenta del principal fallo que tiene el dataset: la falta de registros. Desde luego las conclusiones que podamos sacar de este estudio solo pueden ser tan buenas como lo sea la base de datos.

```
players.df <- split(shots, shots$player)
total_goals <- sapply(players.df, function(x) sum(x$is_goal))
top <- tail(sort(total_goals), 5); top
```

```
##                   Antoine Griezmann Cristiano Ronaldo dos Santos Aveiro
##                                  43                                  43
##       Neymar da Silva Santos Junior            Luis Alberto Suárez Díaz
##                                  61                                 135
##      Lionel Andrés Messi Cuccittini
##                                 386
```

Con una simple búsqueda en Google podemos ver cuántos goles marcó Cristiano Ronaldo en su paso por La Liga: unos 300. Muy lejos de los 43 que parece que tiene registrados. Por supuesto, el resto de jugadores tiene también menos goles.

2. Modelo de regresión logística

En nuestro estudio, la variable respuesta es binaria: el jugador marca gol (1) o no lo marca (0). En este contexto, la regresión lineal no es adecuada, ya que su salida puede tomar cualquier valor, mientras que nosotros necesitamos probabilidades, es decir, valores entre 0 y 1. Por ello usamos **regresión logística**, cuyo modelo genera probabilidades ajustadas mediante la función sigmoide, σ:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

2.1. Modelo logístico

Llamamos $\pi_i := P(Y = 1 \mid X_i)$ donde X_i es el vector de variables observadas en el registro i-ésimo. El modelo asume que existe una relación lineal entre el log-odds de π_i y X_i, i.e.:

$$\log(\frac{\pi_i}{1 - \pi_i}) = \theta^T X_i$$

donde θ es el vector fila de parámetros que debemos ajustar.

2.2. Verosimilitud y función objetivo

Dado que estamos modelando una variable binaria Y, asumimos que sigue una distribución Bernoulli con parámetro p. Debido a que la variable es Bernoulli, minimizar errores cuadráticos no es aconsejable para calcular los parámetros. En cambio, se prefiere seguir el principio de máxima verosimilitud. Así, la **función de verosimilitud** para un conjunto de datos $(X_1, Y_1), \ldots, (X_n, Y_n)$ es:

$$L(\theta) = \prod_{i=1}^{n} \pi_i^{Y_i} (1 - \pi_i)^{1-Y_i}, \quad \text{donde } \pi_i = \sigma(\theta^T X_i)$$

2.3. Maximización de la log-verosimilitud

Es estándar en este contexto maximizar la función log-verosimilitud:

$$\ell(\theta) = \sum_{i=1}^{n} [Y_i \log(\pi_i) + (1 - Y_i) \log(1 - \pi_i)]$$

Este valor es el que queremos **maximizar** para obtener los mejores parámetros θ. Esta maximización no tiene solución analítica cerrada, por lo que se realiza mediante métodos numéricos como el descenso de gradiente. En `R` se hace con `glm`.

3. Análisis estadístico

Partimos los datos en entrenamiento y prueba y ajustamos el modelo logístico con los datos de entrenamiento. Como el resultado no es satisfactorio de primeras no lo vamos a poner, en cambio explicamos los problemas que se observan::

```
modelo <- glm(is_goal ~ is_pressed + from + is_open + technique + body_part
              + situation + distance,
              data = train, family = binomial)
summary(modelo)
```

Se observa que hay algunas variables que no son significativas para el modelo y/o que tienen un error estándar muy grande. Vamos a depurar el modelo como sigue:

- `situation`. Pensamos que podrá deberse a que `from` y `situation` están asociadas, así que lo comprobamos.

  La tabla de contingencia más abajo muestra asociación y el test χ^2 lo confirma. Nos quedamos con `situation` y desechamos `from` por ser más complicada y tener una intepretación más complicada.

- `technique`. Se aprecia que hay categorías que no son significativas así que vamos a agrupar todas las que sean distintas de `Normal` y `Lob`.

- `bodypartOther`. Eliminamos la categoría porque el modelo considera que no es significativa. Es probable que no haya demasiados ejemplos de este caso y que tampoco suponga mucha diferencia.
- `situationCorner`. Igual que el punto anterior. Por tanto agrupamos `Corner` con `FreeKick`.

Ejecutamos este código para efectuar los cambios propuestos:

```
table(train$from, train$situation)
chisq.test(table(train$from, train$situation))

shots$technique <- as.character(shots$technique)
    # Como es un factor lo convertimos primero a character

shots$technique[shots$technique != "Normal" & shots$technique !="Lob"] <- "Other"
shots$technique <- as.factor(shots$technique)
summary(shots$technique) # Ahora hay solo tres categorías

shots <- shots[!(shots$body_part == "Other"), -2]
shots$body_part <- droplevels(shots$body_part)
summary(shots$body_part)

shots$situation <- as.character(shots$situation)
shots$situation[shots$situation == "Corner"] <- "Free Kick"
shots$situation <- as.factor(shots$situation)
summary(shots$body_part)
```

Reajustamos el modelo depurado:

```
set.seed(2002)
n <- nrow(shots)
tamano_entrenamiento <- 0.7*n
indice_entrenamiento <- sample(1:n, size = 0.7 * n)

train <- shots[indice_entrenamiento, ]
test  <- shots[-indice_entrenamiento, ]

## Ajustamos el modelo logístico
modelo <- glm(is_goal ~ is_pressed + is_open + technique + body_part
              + situation + distance,
              data = train, family = binomial)
summary(modelo)

##
## Call:
## glm(formula = is_goal ~ is_pressed + is_open + technique + body_part +
##     situation + distance, family = binomial, data = train)
```

```
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          1.948141   0.283145   6.880 5.97e-12 ***
## is_pressedTRUE      -0.231929   0.086400  -2.684  0.00727 **
## is_openTRUE          1.821966   0.224094   8.130 4.28e-16 ***
## techniqueNormal     -1.111207   0.181370  -6.127 8.97e-10 ***
## techniqueOther      -1.654237   0.189860  -8.713  < 2e-16 ***
## body_partLeft Foot   1.368795   0.106943  12.799  < 2e-16 ***
## body_partRight Foot  1.341607   0.104940  12.784  < 2e-16 ***
## situationOpen Play  -1.226300   0.162359  -7.553 4.25e-14 ***
## situationPenalty     1.467760   0.287538   5.105 3.32e-07 ***
## distance            -0.171841   0.006268 -27.417  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9121.9  on 12270  degrees of freedom
## Residual deviance: 7334.0  on 12261  degrees of freedom
## AIC: 7354
##
## Number of Fisher Scoring iterations: 6
```

Hemos obtenido un mejor modelo. Al menos todas las variables son significativas ya. Vamos a comprobar ahora que se cumplen las hipótesis para asegurarnos que el modelo es válido.

- Independencia de las observaciones. Es quizás el más polémico pues en la base de datos faltan registros. De hecho parece haber muchos más datos de Messi por ejemplo, lo cual puede sesgar el resultado.
- Ausencia de multicolinealidad y tratamiento de outliers.
- Hipótesis de linealidad del log-odds.

```
# Comprobamos que ya no exista multicolinealidad
library("car")

## Cargando paquete requerido: carData

vif(modelo) # Es buena # Si GVIF^(1/(2·Df)) < 2

##                GVIF Df GVIF^(1/(2*Df))
## is_pressed 1.160442  1        1.077238
## is_open    1.049752  1        1.024574
## technique  1.172721  2        1.040636
```

```
## body_part  1.417178  2        1.091079
## situation  1.287678  2        1.065251
## distance   1.580175  1        1.257050

            # Sin evidencia de colinealidad preocupante.

pchisq(deviance(modelo), df.residual(modelo),lower=FALSE)

## [1] 1

            # No hay multicolinealidad
```

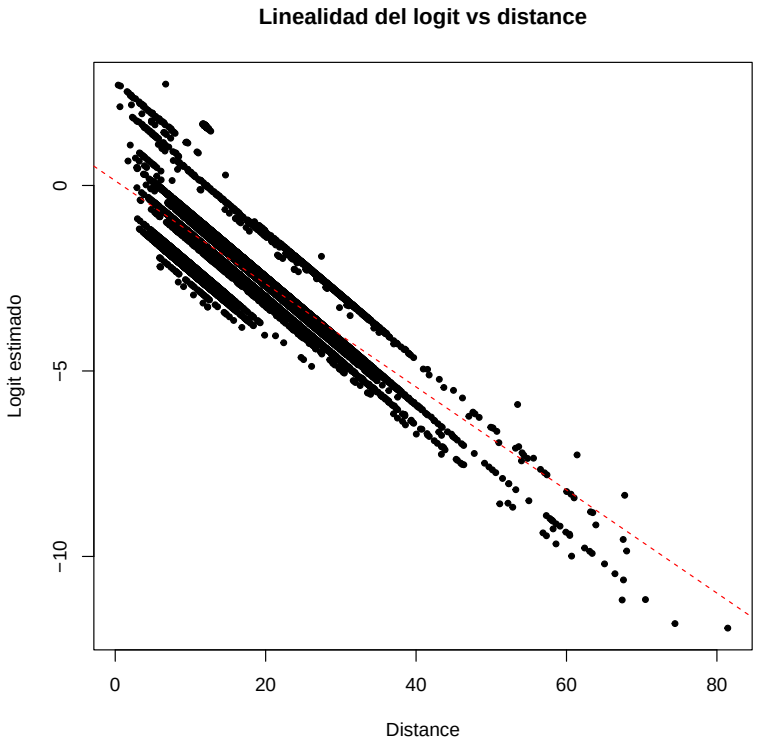
```
# Análisis de residuos para descartar outliers
res_deviance <- residuals(modelo, type = "deviance")

# Seguimos la regla: mayor de 3 outlier
umbral <- 3
outliers_deviance <- which(abs(res_deviance) > umbral)

obs_outliers_deviance <- train[outliers_deviance,]
# summary(obs_outliers_deviance)
```

Si bien hemos comprobado que según ciertos baremos podría haber outliers a nosotros nos parecen observaciones que se deben a la variabilidad natural de los datos y no son especialmente preocupantes. Todos los outliers (hay 11) son goles muy lejanos y en cierta medida el modelo tiene razón, son goles muy complicados de hacer, pero se hacen también goles difíciles y es en parte el objetivo del análisis que aprenda a diferenciarlos.

La única variable continua es la distancia así que es la única para la que vamos a comprobar la hipótesis de linealidad (el resto de variables son *dummies*).




```
summary(linear_model_distance)$r.squared

## [1] 0.7426779
```

Se observa linealidad en la gráfica y se obtuvo un coeficiente de determinación lineal alto por lo que concluimos que se cumple la última hipótesis.

Como se cumplen todos los supuestos podemos asegurar que es un buen modelo desde el punto de vista formal. Vamos ahora a evaluar el modelo con dos análisis:

1. *Accuracy*. Es el porcentaje de acierto que obtiene el modelo en las predicciones sobre los datos de prueba.

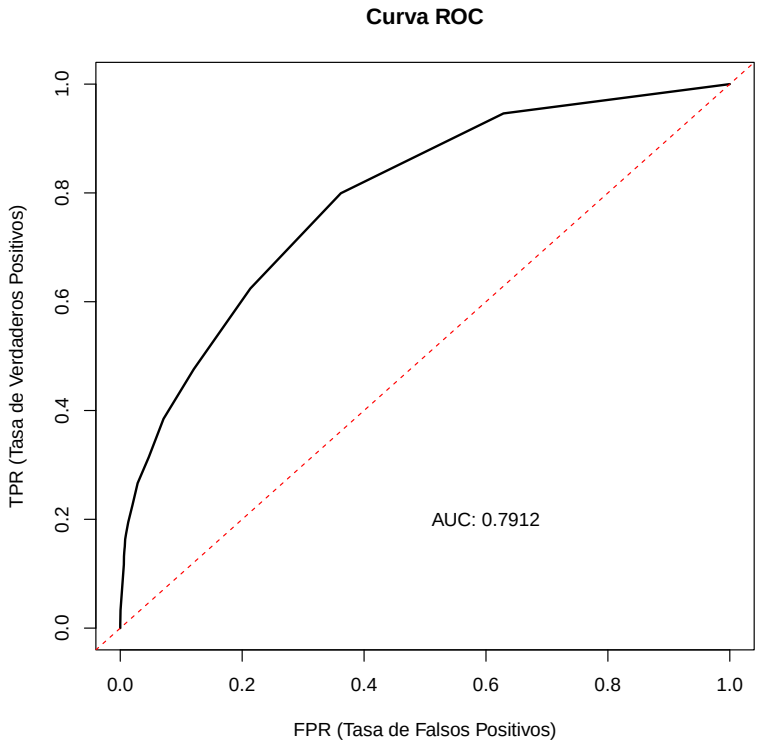
2. *Análisis ROC*.

```
# Accuracy
predicciones_prob <- predict(modelo, newdata = test, type = "response")
prediccion_clase <- ifelse(predicciones_prob > 0.5, 1, 0)
matrix_confusion <- table(Real = test$is_goal, Predicho = prediccion_clase)
accuracy <- (matrix_confusion[1]+matrix_confusion[4])/nrow(test); accuracy


## [1] 0.886692
```

Un 88 % de acierto nos parece buena precisión.

El código de como calcular la curva ROC se ha ocultado para ganar espacio. Aún así se puede comprobar en el archivo `.Rnw`.



El análisis ROC nos da también un buen resultado. Cuánto más parecida sea a la curva constante 1 mejor es el modelo. Para ello se calcula el AUC (*Area Under the Curve*), y cuanto más cerca de 1 mejor es el modelo porque más se parece a la curva 1.

Por último, vamos a calcular la probabilidad de marcar un gol en general, sin tener en cuenta la posición ni nada. Para ello calculamos la media de las predicciones del modelo y obtenemos que con un 95 % de confianza la media se encuentra entre 0.121 y 0.125, eso indica que el modelo asigna 1 de cada 10 tiros aproximadamente como gol, lo cual se corresponde con los datos que tenemos.

```
shots$predicciones <- predict(modelo, newdata = shots[,c("is_pressed", "is_open",
                                                         "technique", "body_part",
                                                         "situation", "distance")],
                              type="response")
mean(shots$predicciones)

## [1] 0.1232946

summary(shots$predicciones)

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 2.680e-06 3.607e-02 8.240e-02 1.233e-01 1.532e-01 9.403e-01
```

```r
par(mfrow = c(1,2))
# hist(shots£predicciones, freq = F)

library(bootstrap)
set.seed(1)
B <- 1000
Tx.boot<-bootstrap(shots$predicciones, nboot=B,theta=mean)$thetastar
quantile(Tx.boot,probs=c(0.025,0.975))

##      2.5%     97.5%
## 0.1211707 0.1253012

# la distribución bootstrap
# hist(Tx.boot,main='Histograma de las medias bootstrap')
```

Comparamos con los datos que tenemos y obtenemos una proporción similar a la probabilidad obtenida anteriormente:

```r
aux <- table(shots$is_goal)
aux[2]/nrow(shots)

##         1
## 0.1237807
```

4. Ránking

Procedemos a calcular el impacto de los jugadores. Vamos a proponer dos métricas y comparar los resultados entre sí. Teníamos pensado hacer más pero tenemos que ajustarnos al límite de 1 páginas. Estas son:

- Primera métrica. La llamaremos *impacto*. Consiste en calcular para cada jugador la media entre la diferencia del resultado de cada tiro con la predicción que obtuvo el modelo.
- Segunda métrica. Interpolar para cada jugador entre su impacto obtenido por la primera métrica y la media de los impactos. Interpolaremos de forma que si el número de tiros es *pequeño* entonces el valor que obtendremos está más cercano a la media. Por el contrario si el número de tiros es más alto (tenemos, más muestra) dejaremos un valor muy parecido al impacto que se calculó en el apartado anterior.

Este último procedimiento se interpreta como **credibilidad**, cuantos más tiros tenemos más razones para pensar que el impacto calculado en el primer apartado es más creíble. Si tiene pocos tiros la intepretación es que el impacto calculado no es creíble y se tiende

a poner el valor medio de impacto. Esto lo hacemos porque esta primera métrica propuesta es muy sensible al tamaño de la muestra (en este caso la cantidad de tiros), como veremos ahora.

El factor interpolante w se calcula como

$$w = \frac{n0}{n0 + n}$$

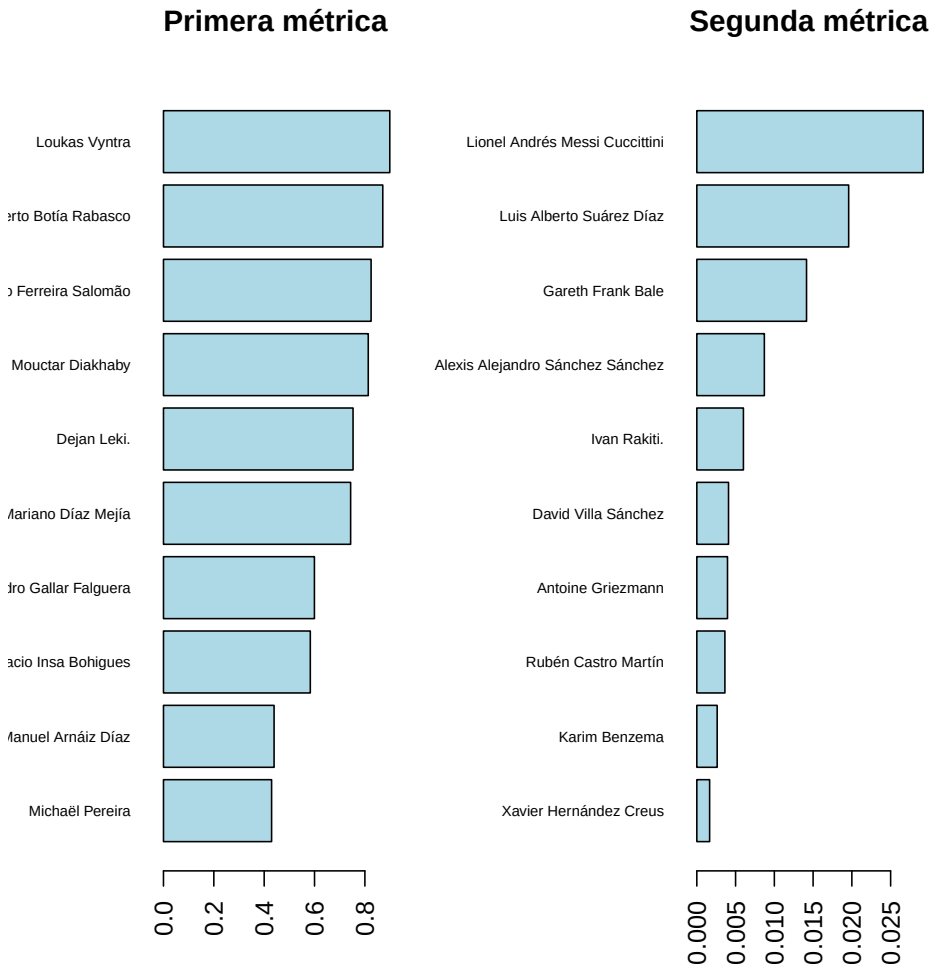
donde n_0 es el umbral de credibilidad y n es el número de tiros del jugador.

Para nosotros el umbral de credibilidad serán 200 tiros. Con menos de 200 tiros se tiende a la media y con más se tiende al impacto del jugador. La implementación se presenta a continuación.

```
players.df <- split(shots, shots$player)

# Métrica uno
impact <- sapply(players.df, function(x) mean(x$is_goal-x$predicciones))
top <- tail(sort(impact), 10)
par(mfrow = c(1, 2), mar = c(5, 5, 4, 5))
barplot(top, las = 2, cex.names = 0.6, col = "lightblue",
        main = "Primera métrica", horiz = TRUE)

# Métrica dos
ave.impact <- mean(impact)
# n0 <- mean(sapply(players.df, nrow))
# n0 <- median(sapply(players.df, nrow))
n0 <- 200
w <- n0/(n0+sapply(players.df, nrow))
credibility <- ave.impact*w+(1-w)*impact
top <- tail(sort(credibility), 10)
barplot(top, las = 2, cex.names = 0.6, col = "lightblue",
        main= "Segunda métrica", horiz = TRUE)
```



Con la segunda métrica aparecen jugadores de talla mundial como Messi, Suárez, Cristiano, etc. Lo cual es una señal de que la segunda métrica capta mejor la idea que teníamos en mente. Como mencionábamos, una de las desventajas de la métrica es que beneficia a jugadores con una cantidad menor de tiros. Por ejemplo, según la primer, Loukas Vyntra es el jugador con mayor impacto:

```
players.df["Loukas Vyntra"]


## $`Loukas Vyntra`
##       is_pressed        player is_open technique body_part situation is_goal
## 14716       TRUE Loukas Vyntra   FALSE    Normal      Head Open Play       1
##       distance predicciones
## 14716      9.1    0.1011013
```

5. Conclusiones

Se ha ajustado un modelo de regresión logística para calcular la probabilidad de gol a partir de variables contextuales. Los datos revelan que jugadores como Leo Messi o Luis Suárez, mundialmente conocidos, son los los jugadores que más partido sacan a sus tiros, porque son capaces de convertir tiros de baja probabilidad de acierto en gol. Si bien estas conclusiones parecen ajustadas a la realidad hay que tomarlas con cuidado debido a la falta de registros. La principal limitación que hemos encontrado para este modelo es la falta de una base de datos adecuada para este tipo de análisis. La hemos tenido que confecccionar nosotros y como se comentaba en la sección de datos faltan bastantes registros. Esto puede sesgar las predicciones del modelo, pues se requiere que la muestra sea aleatoria y simple. Como no sabemos cuántas ni que observaciones faltan no podemos saber si esta ausencia está produciendo un modelo sesgado o si por el contrario, aunque de forma improbable, las observaciones no disponibles no suponen un cambio significativo. Una forma de mejorar este análisis sería mejorar la calidad de los datos.

Por otro lado los coeficientes ajustados nos permiten interpretar la influencia de cada variable sobre la respuesta.

```
library("MASS")
confints <- confint(modelo)

## Waiting for profiling to be done...

exp(confints)

##                           2.5 %     97.5 %
## (Intercept)          4.0149642 12.1907937
## is_pressedTRUE       0.6683364  0.9378509
## is_openTRUE          4.0388722  9.7505569
## techniqueNormal      0.2316795  0.4722411
## techniqueOther       0.1322560  0.2786873
## body_partLeft Foot   3.1919913  4.8548720
## body_partRight Foot  3.1189419  4.7066573
## situationOpen Play   0.2148053  0.4063666
## situationPenalty     2.5102295  7.7727518
## distance             0.8317218  0.8524121
```

Con un 95 % de confianza, estar presionado disminuye los odds de marcar un gol entre 6.2 y 33.2 en comparación con no estarlo. Por el contrario, lanzar un penalti incrementa los odds de gol entre 2.51 y 7.77 veces respecto a lanzar un tiro libre directo, que es la categoría base. También se aprecia como tirar con el pie (ya sea derecho o izquierdo) aumenta entre 3 y 5 veces el odds de marcar gol frente a tirar con la cabeza. Por último observamos que, de media, por cada metro que nos alejamos de la portería el odds de marcar gol se disminuye entre 0.15 y 0.17. Por mencionar algunos coeficientes.

Si bien el modelo puede no ser el mejor por todos los problemas que ya comentamos, las conclusiones que podemos extraer son intuitivas y razonables.

6. Referencias

- Faraway, J.J. (2006) Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models. Chapman Hall/CRC. Texts in Statistical Science Series.