

Kobe Bryant Shot Selection

Proyecto Kaggle

Laura Rodríguez Navas

Introducción

Este trabajo lleva a cabo un proyecto completo de Ciencia de Datos donde vamos a analizar, transformar, modelar y evaluar un conjunto de datos de *Kaggle*. Concretamente, para este trabajo se ha usado un conjunto de datos que describe los aciertos y los fallos de lanzamientos a canasta del jugador de baloncesto Kobe Bryant durante los 20 años de su carrera en la NBA (<https://www.kaggle.com/c/kobe-bryant-shot-selection/data/>).

El conjunto de datos contiene 30697 instancias y un gran número de variables explicativas (11 discretas y 14 numéricas). Estas 25 variables (incluyendo clase a predecir “*shot_made_flag*”) se centran en la descripción cualitativa y cuantitativa de multitud de aspectos de cada uno de los lanzamientos de Kobe Bryant.

```
## 'data.frame':   30697 obs. of  25 variables:
## $ action_type      : Factor w/ 57 levels "Alley Oop Dunk Shot",...: 27 27 27 27 6 ..
## $ combined_shot_type: Factor w/ 6 levels "Bank Shot","Dunk",...: 4 4 4 4 2 4 5 4 4 ..
## $ game_event_id    : int   10 12 35 43 155 244 251 254 265 294 ...
## $ game_id           : int   20000012 20000012 20000012 20000012 20000012 20000012 2..
## $ lat               : num   34 34 33.9 33.9 34 ...
## $ loc_x             : int   167 -157 -101 138 0 -145 0 1 -65 -33 ...
## $ loc_y             : int   72 0 135 175 0 -11 0 28 108 125 ...
## $ lon               : num  -118 -118 -118 -118 -118 ...
## $ minutes_remaining : int   10 10 7 6 6 9 8 8 6 3 ...
## $ period            : int    1 1 1 1 2 3 3 3 3 3 ...
## $ playoffs          : int    0 0 0 0 0 0 0 0 0 0 ...
## $ season            : Factor w/ 20 levels "1996-97","1997-98",...: 5 5 5 5 5 5 5 5 ..
## $ seconds_remaining : int   27 22 45 52 19 32 52 5 12 36 ...
## $ shot_distance     : int   18 15 16 22 0 14 0 2 12 12 ...
## $ shot_made_flag    : int    NA 0 1 0 1 0 1 NA 1 0 ...
## $ shot_type         : Factor w/ 2 levels "2PT Field Goal",...: 1 1 1 1 1 1 1 1 1 ..
## $ shot_zone_area    : Factor w/ 6 levels "Back Court(BC)",...: 6 4 3 5 2 4 2 2 4 2 ..
## $ shot_zone_basic   : Factor w/ 7 levels "Above the Break 3",...: 5 5 5 5 6 5 6 6 3..
## $ shot_zone_range   : Factor w/ 5 levels "16-24 ft.", "24+ ft.",...: 1 3 1 1 5 3 5 5..
## $ team_id           : int  1610612747 1610612747 1610612747 1610612747 1610612747 ..
## $ team_name         : Factor w/ 1 level "Los Angeles Lakers": 1 1 1 1 1 1 1 1 1 1 ..
## $ game_date         : Factor w/ 1559 levels "1996-11-03","1996-11-05",...: 311 311 ..
## $ matchup           : Factor w/ 74 levels "LAL @ ATL","LAL @ BKN",...: 29 29 29 29 ..
## $ opponent          : Factor w/ 33 levels "ATL","BKN","BOS",...: 26 26 26 26 26 26 ..
## $ shot_id           : int    1 2 3 4 5 6 7 8 9 10 ...
```

La tarea de este trabajo es predecir si los lanzamientos a canastas de Kobe Bryant entraron o no en el aro, es decir, los lanzamientos acertados (atributo “*shot_made_flag*”). Del conjunto de datos se han eliminado 5000 valores de este atributo (representados como valores faltantes). Estos datos estarán en el conjunto de evaluación (test) sobre el cual se realizará la predicción.

Una vez descargados los datos, es necesario dividir el conjunto de datos en un conjunto de datos de entrenamiento y un conjunto de datos de evaluación (test). Para ello, primero analizamos la existencia de esos valores faltantes nombrados anteriormente (atributo “*shot_made_flag*”), que serán los valores que tendremos que predecir y que estarán en el conjunto de datos de test pero no en el conjunto de datos de entrenamiento.

```
train <- data[!is.na(data$shot_made_flag), ]  
any(is.na(train))
```

```
## [1] FALSE
```

```
## int [1:25697] 0 1 0 1 0 1 1 0 0 1 ...
```

```
test <- data[is.na(data$shot_made_flag), ]  
any(is.na(test))
```

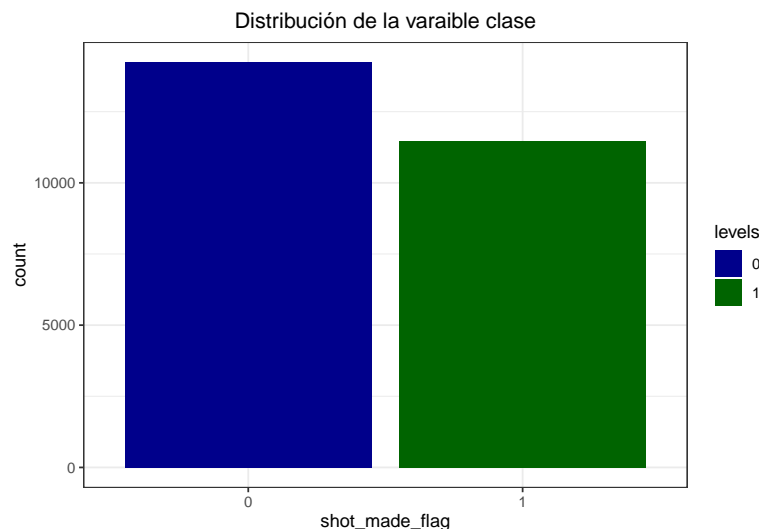
```
## [1] TRUE
```

```
## int [1:5000] NA NA NA NA NA NA NA NA NA NA ...
```

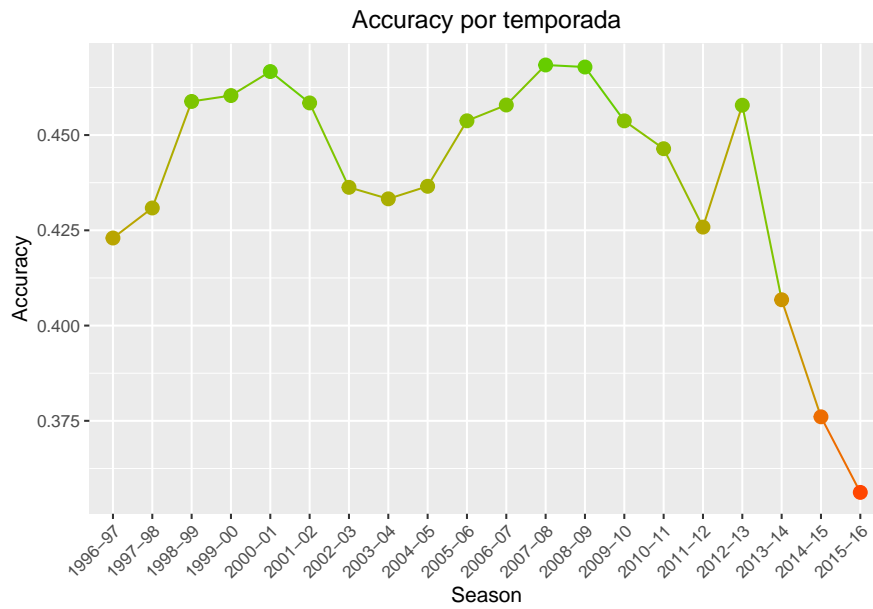
Una vez dividido el conjunto de los datos, es necesario realizar un análisis del conjunto de datos, así como un proceso de exploración, transformación y limpieza de estos datos con el objetivo de resaltar información útil para la fase de modelado. Este análisis nos permitirá controlar la presencia de valores fuera de rango, una idea inicial de la forma que tienen los datos, etc. así como las relaciones entre los distintos atributos. Aunque para sintetizar el análisis realizado solo se comentará aquello que se ha considerado más interesante durante éste.

Exploración, Limpieza y Transformación

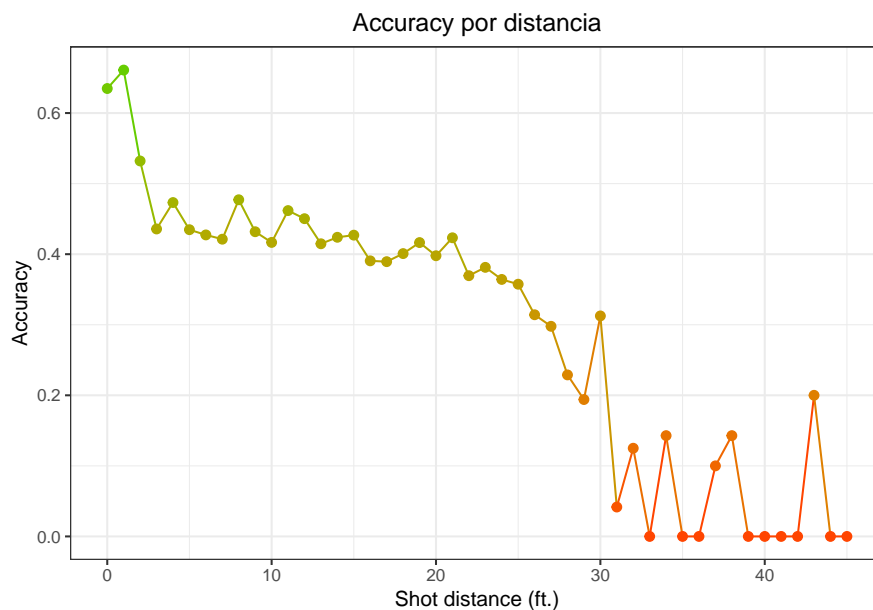
Empezamos analizando visualmente la variable de clase a predecir (atributo “*shot_made_flag*”), que es binaria y que se distribuye de manera bastante equitativa. Vemos que el número de canastas que no entraron en el aro es superior al número de canastas que sí que entraron. Así que, intentaremos averiguar si esto puede estar relacionado con la gran lesión que tuvo Kobe Bryant durante la temporada 2013-14.



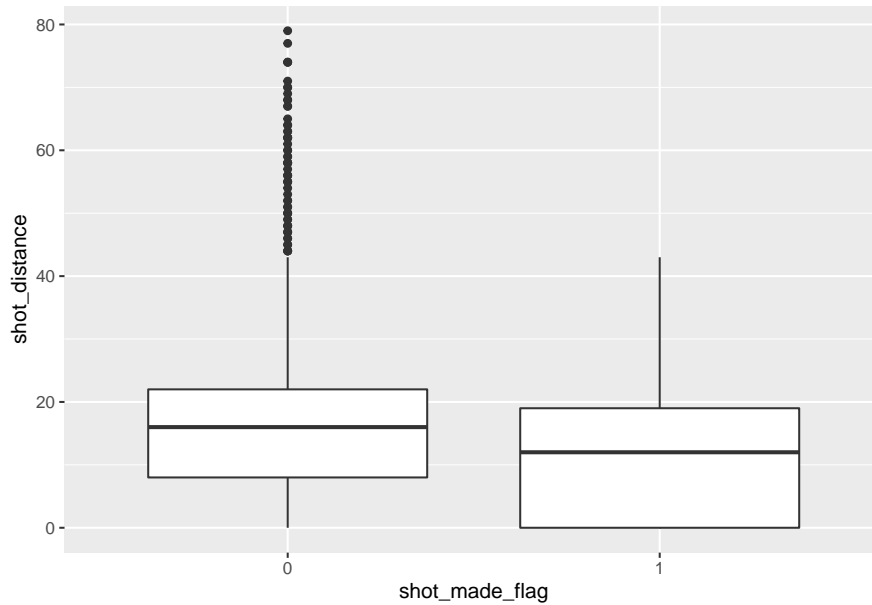
A continuación, analizamos visualmente la precisión de los lanzamientos realizados por temporada (atributo “*season*”), y vemos que a partir de la temporada 2013-14 la precisión de los lanzamientos baja drásticamente. ¿Así que, una gran cantidad de los lanzamientos que no entraron en el aro podrían estar relacionados con la gran lesión que tuvo Kobe Bryant en la temporada 2013-14? Seguramente que sí.



En el siguiente gráfico analizamos visualmente la precisión de los lanzamientos respecto a la distancia del tiro (atributo “*shot_distance*”), porqué nos hemos fijado que diferentes exploraciones de datos que han realizado otros usuarios en *Kaggle*, se ha podido observar que el atributo “*shot_distance*” contiene valores fuera de rango que podríamos eliminar y que nos sería muy beneficioso para reducir los fallos durante la predicción. Los valores fuera de rango podrían encontrarse a partir de los lanzamientos realizados a más de 30 (ft.) ya que la precisión de baja drásticamente a partir de este punto.



Y en el siguiente grafico podemos observar la existencia de esos valores fuera de rango que estábamos buscando. Los valores que exactamente se encuentran fuera de rango se producen cuando el lanzamiento se realiza a una distancia superior a 40 (ft.). Así que, más adelante serán eliminados los valores del atributo *shot_distance* superiores a 40 (ft.).



Hemos realizado una pequeña exploración de los datos donde hemos analizado visualmente la precisión de los lanzamientos por temporada, los lanzamientos respecto a la distancia y hemos encontrado valores fuera de rango en el conjunto de datos que tendremos que eliminar si queremos realizar una buena predicción de la variable clase “*shot_made_flag*”.

Al trabajar con un conjunto de datos real y con muchos atributos, debemos tener en cuenta el hecho de que algunos datos pueden faltar o estar dañados, por lo tanto, es crucial realizar los procesos de transformación y limpieza del conjunto de datos para obtener un buen ajuste del modelo y una mejor capacidad predictiva.

Limpieza

En la limpieza de datos primero eliminamos los atributos que hemos creído independientes al modelado. Los atributos que pueden descartarse son:

- **game_event_id**. Independiente al modelado.
- **game_id**. Independiente al modelado.
- **loc_x**. Correlacionada con lat.
- **loc_y**. Correlacionada con lon.
- **lat**. Correlacionada con loc_x.
- **lon**. Correlacionada con loc_y.
- **shot_zone_area**. Independiente al modelado.
- **shot_zone_basic**. Independiente al modelado.
- **shot_zone_range**. Independiente al modelado.
- **team_id**. Siempre es el mismo número.
- **team_name**. Siempre es el mismo valor: *LA Lakers*.
- **game_date**. Independiente al modelado.
- **matchup**. Los atributos *oponent* y *matchup* contienen básicamente la misma información. Nos quedamos solo con el atributo *oponent*.

También eliminamos los valores fuera de rango, que como hemos visto anteriormente una buena opción sería descartar los lanzamientos realizados a una distancia superior a 40 (ft.). El proceso se muestra a continuación.

```
train$shot_distance[train$shot_distance > 40] <- 40
test$shot_distance[test$shot_distance > 40] <- 40
```

El conjunto de datos de entrenamiento y de test después de la limpieza quedan:

```
## 'data.frame': 25697 obs. of 12 variables:
## $ action_type : Factor w/ 57 levels "Alley Oop Dunk Shot",...: 27 27 27 6 27 ..
## $ combined_shot_type: Factor w/ 6 levels "Bank Shot","Dunk",...: 4 4 4 2 4 5 4 4 4 ..
## $ minutes_remaining : int 10 7 6 6 9 8 6 3 1 11 ...
## $ period : int 1 1 1 2 3 3 3 3 1 ...
## $ playoffs : int 0 0 0 0 0 0 0 0 0 0 ...
## $ season : Factor w/ 20 levels "1996-97","1997-98",...: 5 5 5 5 5 5 5 5 ..
## $ seconds_remaining : int 22 45 52 19 32 52 12 36 56 0 ...
## $ shot_distance : num 15 16 22 0 14 0 12 12 25 17 ...
## $ shot_made_flag : int 0 1 0 1 0 1 1 0 0 1 ...
## $ shot_type : Factor w/ 2 levels "2PT Field Goal",...: 1 1 1 1 1 1 1 1 2 1 ..
## $ opponent : Factor w/ 33 levels "ATL","BKN","BOS",...: 26 26 26 26 26 26 ..
## $ shot_id : int 2 3 4 5 6 7 9 10 11 12 ...

## 'data.frame': 5000 obs. of 12 variables:
## $ action_type : Factor w/ 57 levels "Alley Oop Dunk Shot",...: 27 27 13 13 27..
## $ combined_shot_type: Factor w/ 6 levels "Bank Shot","Dunk",...: 4 4 5 5 4 4 5 5 ..
## $ minutes_remaining : int 10 8 0 10 11 10 7 5 4 5 ...
## $ period : int 1 3 1 3 1 1 1 1 2 ...
## $ playoffs : int 0 0 0 0 0 0 0 0 0 0 ...
## $ season : Factor w/ 20 levels "1996-97","1997-98",...: 5 5 5 5 5 5 5 5 ..
## $ seconds_remaining : int 27 5 1 46 26 58 33 58 9 33 ...
## $ shot_distance : num 18 2 0 0 17 20 1 1 0 16 ...
## $ shot_made_flag : int NA NA NA NA NA NA NA NA NA NA ...
## $ shot_type : Factor w/ 2 levels "2PT Field Goal",...: 1 1 1 1 1 1 1 1 1 1 ..
## $ opponent : Factor w/ 33 levels "ATL","BKN","BOS",...: 26 26 31 31 32 32 ..
## $ shot_id : int 1 8 17 20 33 34 35 36 37 38 ...
```

Todas las operaciones de limpieza realizadas se han aplicado sobre el conjunto de datos de entrenamiento y el conjunto de datos de test.

Transformación

La primera transformación que tenemos que realizar es la categorización del atributo *“shot_made_flag”*, porque es la variable clase a predecir e inicialmente es de tipo entero.

```
## int [1:25697] 0 1 0 1 0 1 1 0 0 1 ...
## Factor w/ 2 levels "0","1": 1 2 1 2 1 2 2 1 1 2 ...
```

Si nos fijamos en los atributos *“minutes_remaining”* y *“seconds_remaining”* podríamos realizar la segunda transformación sobre el conjunto de datos ya que la información que contienen la podríamos combinar, los minutos del atributo *“minutes_remaining”* los podríamos convertir a segundos y sumar-los con los segundos del atributo *“seconds_remaining”*. La nueva información combinada la guardaríamos en un nuevo atributo *“time_remaining”*.

El proceso se muestra a continuación y al final los atributos *“minutes_remaining”* y *“seconds_remaining”* se eliminarán porque contendrán información irrelevante al combinar los valores en *“time_remaining”*.

```
train$time_remaining <- train$minutes_remaining * 60 + train$seconds_remaining
test$time_remaining <- test$minutes_remaining * 60 + test$seconds_remaining
train <- subset(train, select = -c(minutes_remaining, seconds_remaining))
test <- subset(test, select = -c(minutes_remaining, seconds_remaining))
```

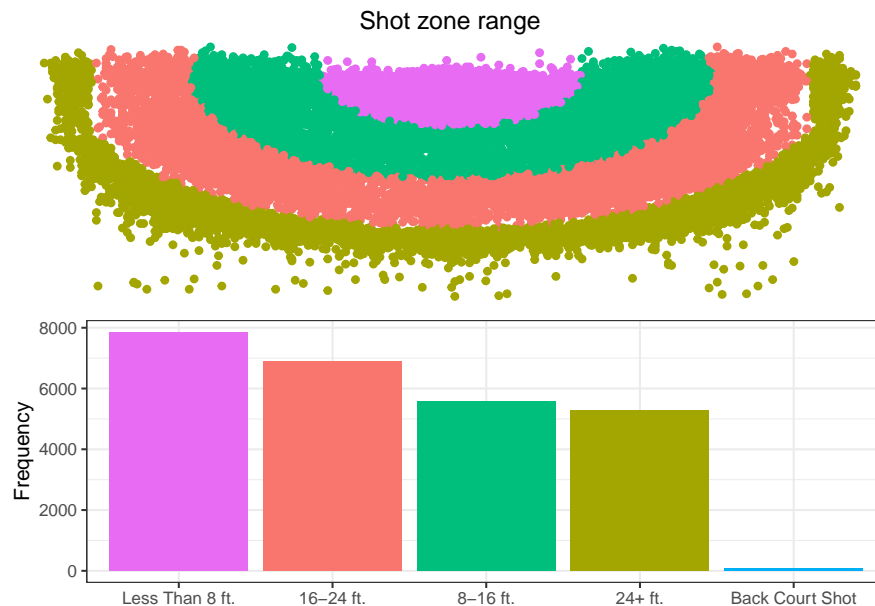
El siguiente paso es normalizar los atributos “*time_remaining*” y “*shot_distance*”. También se hace al final con los valores de la variable clase “*shot_made_flag*”, ya que vamos a predecir probabilidades. Sabemos que tenemos que predecir probabilidades por el fichero de ejemplo que nos facilita *Kaggle*.

```
##  shot_id shot_made_flag
## 1      1      0.5
## 2      8      0.5
## 3     17      0.5
## 4     20      0.5
## 5     33      0.5
```

Procedemos con la normalización.

```
normalize <- function (target) {
  (target - min(target))/(max(target) - min(target))
}
train$shot_distance <- normalize(train$shot_distance)
test$shot_distance <- normalize(test$shot_distance)
train$time_remaining <- normalize(train$time_remaining)
test$time_remaining <- normalize(test$time_remaining)
```

Solo normalizamos los atributos “*time_remaining*” y “*shot_distance*” porque queremos investigar la relación entre los aciertos y los fallos con la distancia de los lanzamientos, durante más tiempo. Como lo sugiere el gráfico siguiente: cuando la distancia se hace mayor, la frecuencia de lanzamiento disminuye.



Todas las operaciones de transformación realizadas se han aplicado sobre el conjunto de datos de entrenamiento y el conjunto de datos de test.

Modelado y Evaluación

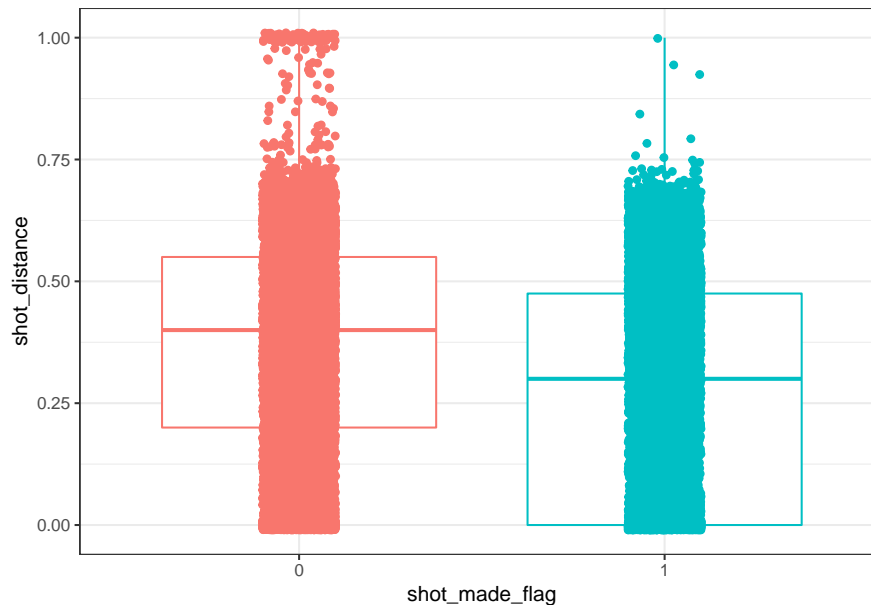
Una vez se ha realizado la exploración, la transformación y la limpieza de los datos, pasamos a la fase del modelado. Crearemos dos nuevos conjuntos de datos sobre los conjuntos de datos de entrenamiento y el conjunto de datos de test con los atributos (“*time_remaining*” y “*shot_distance*”) y la variable de clase a predecir (“*shot_made_flag*”), que usaremos para hacer la predicción. Que como se ha comentado anteriormente, porque queremos investigar la relación entre los aciertos y los fallos con la distancia de los lanzamientos, durante más tiempo.

Los nuevos conjuntos de datos de entrenamiento y de test después de hacer esta selección de variables son:

```
##      shot_distance time_remaining shot_made_flag
## 16424      0.550      1.0000000      0
## 20392      0.550      1.0000000      1
## 20411      0.000      1.0000000      1
## 2963       0.200      0.9985994      0
## 4636       0.525      0.9985994      0

##      shot_distance time_remaining shot_made_flag
## 539      0.000      1.0000000      NA
## 18       0.175      0.9985915      NA
## 2601     0.425      0.9985915      NA
## 4065     0.000      0.9985915      NA
## 4979     0.000      0.9985915      NA
```

Para comprobar que ha sido una buena decisión, se han realizado todas las observaciones posibles de todos los atributos del conjunto de datos de entrenamiento con la variable clase “*shot_made_flag*”. Y hemos observado que parece existir diferencias entre los lanzamientos acertados y los no acertados con la distancia de estos. Así que, la variable escogida “*shot_distance*” puede ser un buen predictor. Lo podemos observar en el siguiente gráfico.



El modelo final que se ha escogido es un modelo de regresión binominal (clasificación binaria) ya que la respuesta de la predicción debe que ser numérica y sabemos que la variable a predecir es categórica de 2 niveles binaria.

La regresión logística, es un método de regresión que permite estimar la probabilidad de una variable cualitativa binaria, como la variable a predecir “*shot_made_flag*”, en función de alguna variable cuantitativa. Es importante tener en cuenta que, aunque la regresión logística permite clasificar, se trata de un modelo de regresión. Que además nos permite calcular la probabilidad de la variable dependiente para a cada una de las dos categorías en función del valor que adquiera la variable independiente.

Generamos el modelo de regresión logística.

```
model <- glm(shot_made_flag~., data=train_dat, family = binomial(link = "logit"))
summary(model)
```

```
##
## Call:
## glm(formula = shot_made_flag ~ ., family = binomial(link = "logit"),
##      data = train_dat)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.3696  -1.0519  -0.8694   1.2015   1.8258
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.30269    0.03058   9.899 < 2e-16 ***
## shot_distance -1.76008    0.05657 -31.111 < 2e-16 ***
## time_remaining  0.13940    0.04401   3.167  0.00154 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 35325  on 25696  degrees of freedom
## Residual deviance: 34298  on 25694  degrees of freedom
## AIC: 34304
##
## Number of Fisher Scoring iterations: 4
```

A la hora de evaluar la validez y la calidad del modelo se utiliza la métrica *Accuracy* y *LogLoss* (score utilizado en Kaggle como evaluación). Para calcular *Accuracy* se va a utilizar un intervalo de 0.5. Es decir, si la probabilidad de que la variable “*shot_made_flag*” adquiera el valor 1 (acierto) es superior a 0.5, se asigna a este nivel, si es menor a 0.5 se asigna al nivel 0 (no acierto).

```
newdata <- data.frame(train_dat[, -3])
pred <- predict(model, newdata, type = 'response')

newdf <- data.frame(shot_id=train$shot_id, shot_made_flag=pred)
normalize <- function(target) {
  (target - min(target))/(max(target) - min(target))
}
newdf$shot_made_flag <- normalize(newdf$shot_made_flag)
preds_th <- ifelse(as.numeric(pred) > 0.5, 1, 0)

# confusion matrix
cm <- table(newdf$shot_made_flag, preds_th)
accuracy <- (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
accuracy

## [1] 0.9294118

loglogg <- logLoss(train_dat$shot_made_flag, pred)
loglogg

## [1] 0.8921015
```


Con este modelo obtenemos un Accuracy de 0.9294118 y un valor de LogLoss de 0.8921015, lo cual parece bastante bueno para realizar la predicción con él.

Resultado

Una vez generado y evaluado el modelo se realiza la predicción para estimar los datos restantes no empleados como entrenamiento, y guardamos el resultado en un fichero `.CSV` para la presentación final del resultado en *Kaggle*. También se normaliza la variable a predecir final.


```
newdata <- data.frame(test_dat[, -3])
pred <- predict(model, newdata, type = "response")
submission <- data.frame(shot_id=test$shot_id, shot_made_flag=pred)
normalize <- function(target) {
  (target - min(target))/(max(target) - min(target))
}
submission$shot_made_flag <- normalize(submission$shot_made_flag)
```

A continuación, vemos los primeros valores del resultado obtenido para tener una idea de como es el fichero que se ha subido.

```
submission <- read.csv("glm.csv")
head(submission, 10)
```

```
##      shot_id shot_made_flag
## 1         1      0.5253336
## 2         8      0.9240430
## 3        17      0.9199584
## 4        20      0.9928629
## 5        33      0.5574973
## 6        34      0.4785334
## 7        35      0.9459042
## 8        36      0.9351108
## 9        37      0.9481853
## 10       38      0.5428182
```

Resultado en Kaggle



Kobe Bryant Shot Selection
Which shots did Kobe sink?
1,117 teams · 4 years ago

OverviewDataNotebooksDiscussionLeaderboardRulesTeamMy SubmissionsLate Submission

Your most recent submission

Name	Submitted	Wait time	Execution time	Score
glm.csv	a few seconds to go	0 seconds	0 seconds	0.86488

Complete

[Jump to your position on the leaderboard](#)

Make a submission for [CDAA_LauraRodriguez](#)

Diferentes implementaciones personales que se ha utilizado para el trabajo se encuentran en el repositorio de *GitHub*: <https://github.com/lrodrin/masterAI/tree/master/A3/Proyecto%20Kaggle>

Conclusiones

El modelo creado para predecir la probabilidad de los lanzamientos acertados a partir de la distancia de estos durante más tiempo ha dado un LogLoss de 0.86488 en *Kaggle*, un valor mejor que el predicho sobre el conjunto de datos de entrenamiento, por tanto, mejor, pero muy parecido al calculado sobre el conjunto de datos de entrenamiento.

Realizar una técnica de regresión logística ha dado buenos resultados, aunque hay resultados mucho mejores en *Kaggle* con la aplicación de otros modelos. Un dato curioso es que muchos de los usuarios de *Kaggle* utilizan el modelo *XGBoost*. Es de mi opinión que el modelo de regresión logística es más entendedor y interesante ya que da muy buenos resultados pero que pocos usuarios lo han utilizado, y teóricamente es un buen candidato para este problema en concreto por sus características.

Además de los buenos resultados que el modelo da en *Kaggle* y del análisis del conjunto de datos, la decisión final de escoger el modelo de regresión logística también se basó en el hecho de que personalmente aún no había trabajado con una técnica de regresión logística durante la asignatura, así dio la posibilidad de investigar y aprender.

Del resultado de la investigación del modelo de regresión logística se comprobó la gran dificultad de la evaluación de este tipo de modelos. Se evalúan de diferente manera a lo que se ha podido practicar en la asignatura, partir de desviaciones, coeficientes, diferentes métricas como p-value, etc. un proceso de evaluación mucho más complicado que en este trabajo no ha sido necesario desarrollar, ya que las realizamos una clasificación binaria.

Citas para fuentes usadas

- Notebook en *Kaggle* de xvivancos (<https://www.kaggle.com/xvivancos/kobe-bryant-shot-selection/>).
- Notebook en *Kaggle* de khozzy (<https://www.kaggle.com/khozzy/kobe-shots-show-me-your-best-model/>).
- Notebook en *Kaggle* de dixhom (<https://www.kaggle.com/dixhom/data-analysis-for-beginners/>).