

Ciencia de Datos - Proyecto Kaggle

Gustavo Quevedo Garrán

Introducción y descripción del problema

Nos encontramos ante un problema en el cual se nos presentan características de **todos los lanzamientos** que Kobe Bryant realizó durante sus **20 años de carrera**. De los más de 30.000 lanzamientos, se ha **ocultado** el valor que indica si el lanzamiento entró o no en la canasta **para 5.000 de ellos**. El objetivo es **predecir** este valor para los lanzamientos en los que no se dispone de él.

Para ello se realizará un completo **análisis de los datos**, comprendiendo cada una de las variables proporcionadas, sus valores y de qué manera pueden afectar a la efectividad de los lanzamientos, con el objeto de utilizar esta información de la mejor manera en la predicción. Algunas variables serán **transformadas** o **eliminadas** antes de la **creación del modelo de aprendizaje** que se utilizará en la predicción.

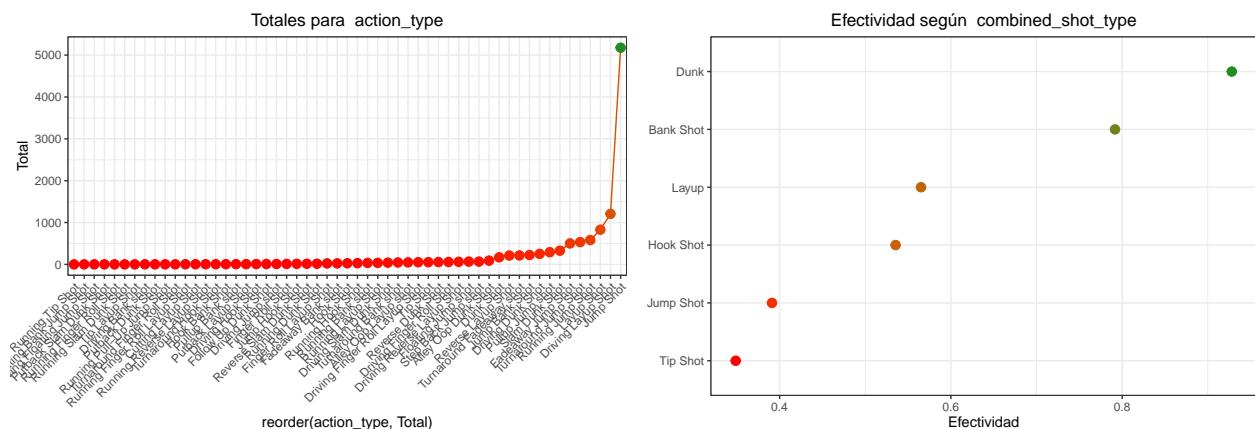
Finalmente se creará un **fichero con la predicción** de dicho modelo, que se subirá a la plataforma **Kaggle**, la cual devolverá un valor con la **evaluación** del mismo. Esta operación se podrá repetir tras realizar ajustes en el modelo, el cual se describirá finalmente en este documento.

Comprendión y preprocessado de datos

Exploración y visualización de datos

A continuación se cargarán los datos y se analizarán las características en tres *data frames*:

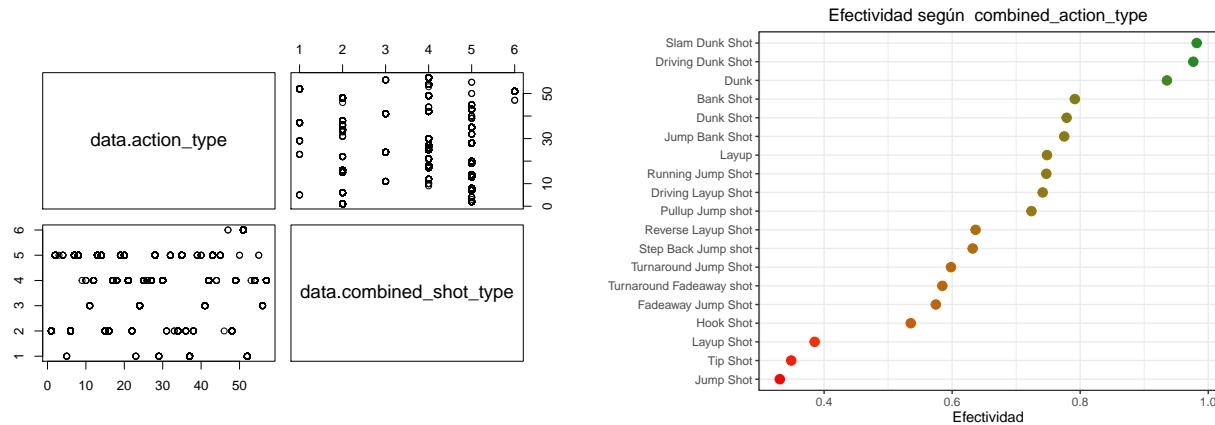
- **data**: Todos los registros. Útil para obtener información que no dependa del éxito del lanzamiento
- **train**: Sólo registros con cuyo resultado es conocido. Se utilizará para obtener gráficos y estadísticas en las que sí intervenga el éxito del lanzamiento (efectividad) y, posteriormente, para entrenar el modelo.
- **test**: Sólo registros cuyo resultado es desconocido. Se le aplicará el mismo preprocessado que a **train**.
- **action_type** y **combined_shot_type**



action_type se trata de una variable categórica. De los distintos valores que toma, se deduce que se trata de los diferentes estilos o acciones técnicas de los lanzamientos.

combined_shot_type es otra variable categórica. De los distintos valores que toma y del nombre de la variable se deduce que puede combinar o agrupar valores de la variable anterior **action_type**.

Sabemos que la efectividad varía en función de **action_type**, a la vez que hay mucha desigualdad entre la cantidad de lanzamientos realizados entre los tipos de acción (figura de arriba a la izquierda). También comprobamos que la efectividad varía en función de **combined_shot_type** (figura de arriba a la derecha).



Podemos comprobar en el gráfico de la izquierda que las variables están totalmente **correlacionadas**.

Optaremos por combinar ambas variables en una sola, de modo que los valores de **action_type** menos usados (menos de 100 repeticiones), serán sustituidos por su **combined_shot_type** asociado. **Creamos una nueva variable combined_action_type y eliminaremos las dos variables existentes.**

Como vemos, nos queda una variable mucho más clara, habiendo extraído y mantenido la información más relevante de ambas (figura de arriba a la derecha).

- **game_event_id**

Se trata de una variable que almacena enteros. Se ha comprobado que el mínimo valor de **game_event_id** va aumentando en función del periodo o cuarto. Se intuye que el saque inicial sería el evento 1 y el primer tiro del partido el evento 2.

No parece lógico que el orden del evento aporte información adicional respecto al tiempo restante o el periodo, por tanto **eliminamos game_event_id**

- **game_id y game_date**

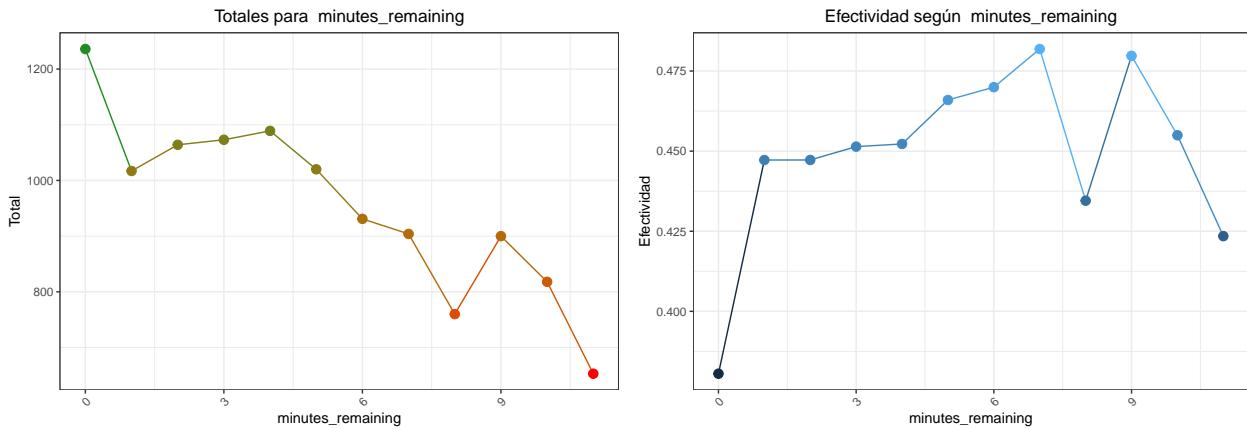
game_id es un identificador único para el partido mientras que **game_date** es la fecha del mismo, aunque en formato cadena de caracteres.

Lo hemos convertido a formato fecha y hemos comprobado que **game_id** y **game_date** están correlacionados, aunque en algunos partidos se traslada la numeración a otro rango diferente. **Eliminamos game_id** ya que no aporta ninguna información útil.

¿Afecta la fecha a la efectividad en los lanzamientos? Lo hemos comprobado con las componentes mes y día de la semana y no parece que tengan gran incidencia en la efectividad. Posiblemente la componente del año sí afecte pero ésta estará obviamente muy correlacionada con la temporada (**season**) por lo que nos deshacemos de **game_date**.

- **seconds_remaining** y **minutes_remaining**

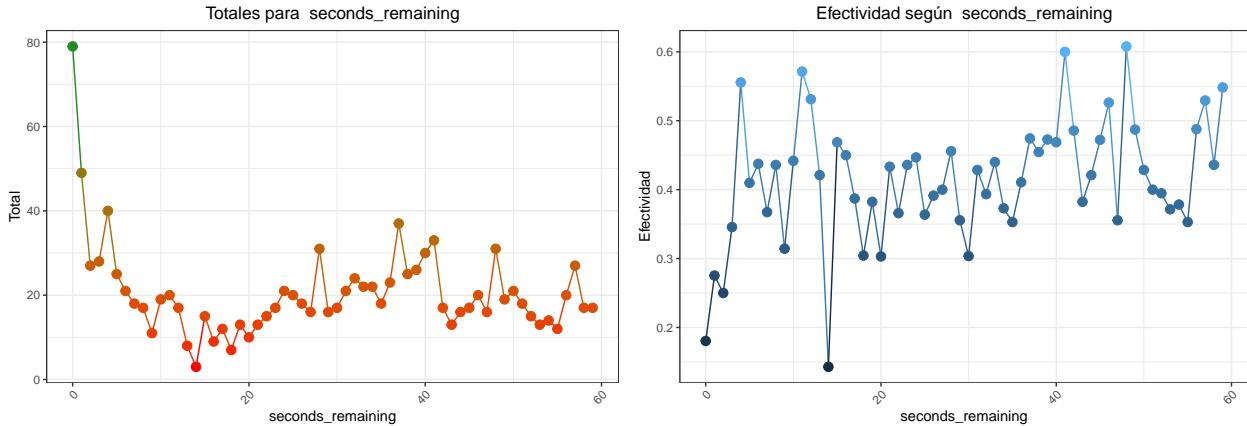
Se trata de las componentes de segundos y minutos para el tiempo restante para finalizar el periodo cuando se produjo el lanzamiento. Visualizamos la efectividad según los minutos restantes



Esta separación en dos variables no tiene sentido para nuestro modelo, ya que los valores de 0 a 59 de los segundos no se entienden sin los minutos. Combinaremos ambas en **seconds_remaining**, que computará también el tiempo de **minutes_remaining**, resultando en el total de segundos restantes.

De los gráficos anteriores se deduce que se producen muchos lanzamientos y el porcentaje baja durante el último minuto de cada cuarto. Esto posiblemente se debe a la importancia de Kobe en su equipo, que le llevó a jugarse los tiros finales decisivos, que por otro lado muchas veces fueron desesperados o muy lejanos lo que hizo bajar el nivel de acierto.

Veremos esto con detalle para el último minuto.



Vemos que en los **últimos tres segundos** se combinan un **alto número de lanzamientos y bajo acierto**.

Se ha comprobado que, sin contabilizar los lanzamientos de los últimos segundos, ya no hay tanta diferencia entre la efectividad del último minuto y el resto. Por tanto sólo vamos a distinguir entre estos segundos finales y el resto, **creando la variable last_seconds** y **deshaciéndonos tanto de minutes_remaining como de seconds_remaining**.

- **period**

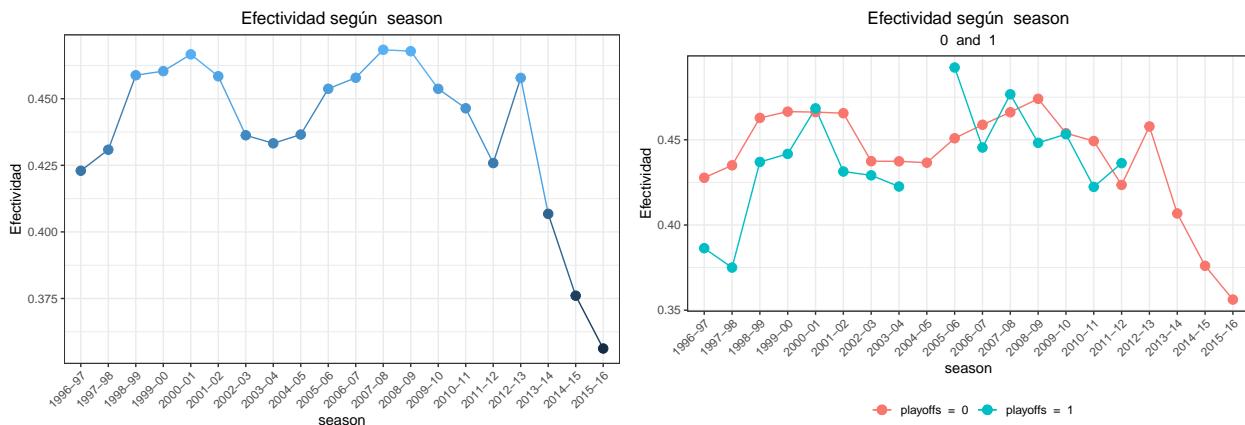
Se trata del valor del periodo o cuarto en el que se realizó el lanzamiento.

Un partido normalmente se divide en cuatro periodos por lo que es normal que el grueso de lanzamientos se produzcan dentro de los valores del 1 a 4. Si el cuarto periodo finaliza en empate entre los equipos, se disputarán nuevos periodos (prórrogas) hasta que alguno finalice sin empate.

Podemos comprobar que existe un decremento de la efectividad en el último cuarto (0.4137 frente a valores de 0.44-0.46). Esto probablemente se deba a tiros desesperados al final del partido intentando una remontada. Pasa lo mismo en el periodo 7 pero la cantidad de datos es tan baja que este caso puede ser ignorado. Por tanto **crearemos la variable last_period descartando el resto de valores de period**.

- **season y playoffs**

season indica la temporada mientras que **playoffs** indica si se trata de partido de eliminatorias finales por el título.

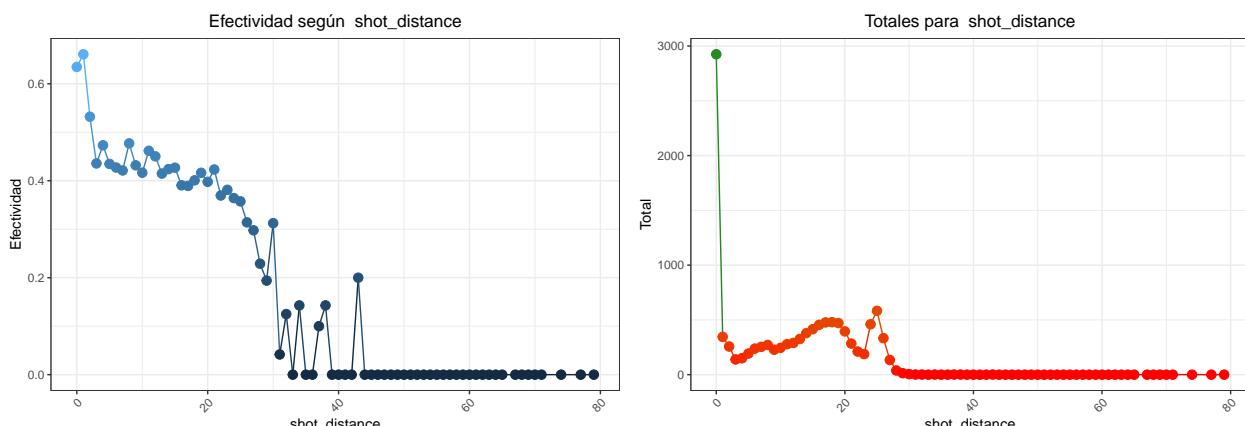


Vemos que los porcentajes varían bastante en función de la temporada (**season**) mientras que, excepto en sus dos primeras temporadas, el hecho de que el partido sea de **playoff** no influye demasiado.

Eliminaremos la variable **playoffs**. Para los cálculos nos dará más información el ordinal de la temporada, en lugar del valor de **season**. Ya que la primera temporada fue la 1996, restaremos 1995 de los primeros cuatro dígitos de **season** y la borramos.

- **shot_distance**

Al tener las coordenadas del tiro podemos calcular la distancia real y almacenarlo en **shot_real_distance**. **shot_distance** no es más la distancia real truncada.



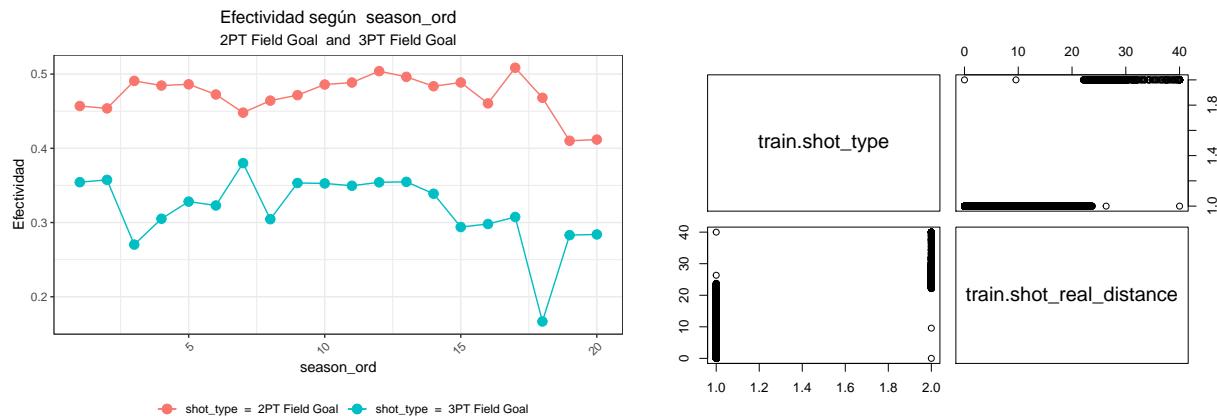
En los dos gráficos anteriores vemos cómo la efectividad y el número de lanzamientos **baja notablemente a partir de una distancia de 40**. Por tanto, por simplificación, fijaremos la distancia en 40 para todos los tiros más lejanos de 40 y nos quedaremos con shot_real_distance que aporta más información para los cálculos.

- shot_made_flag

Se trata de la variable clase que hemos de predecir y la que estamos usando para obtener todas las estadísticas de efectividad para los datos de entrenamiento.

- shot_type

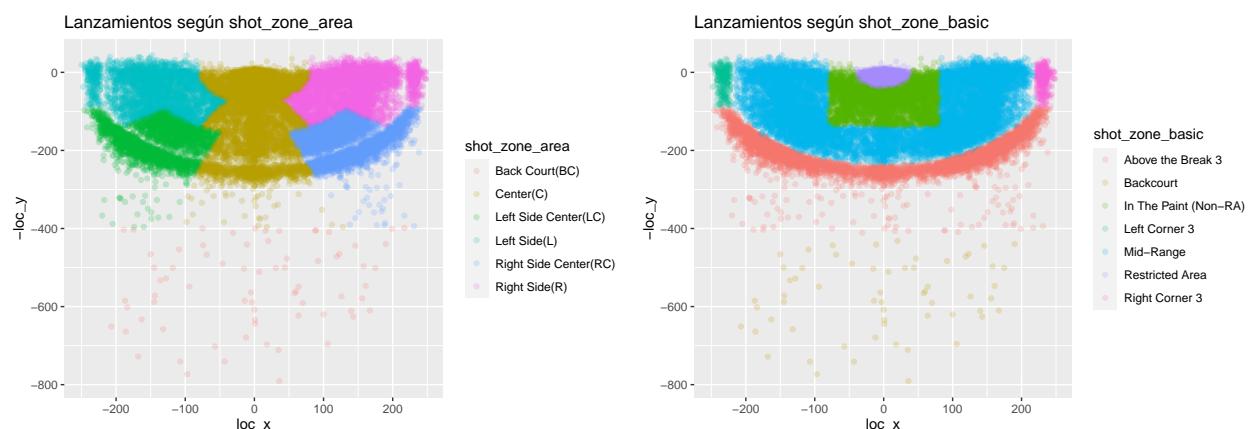
Indica si se trata de lanzamientos de 2 o 3 puntos. Vemos que los lanzamientos de tiro libre (1 punto) no han sido incluidos en el conjunto de datos.



En la figura de la izquierda vemos la efectividad en función de shot_type por temporada.

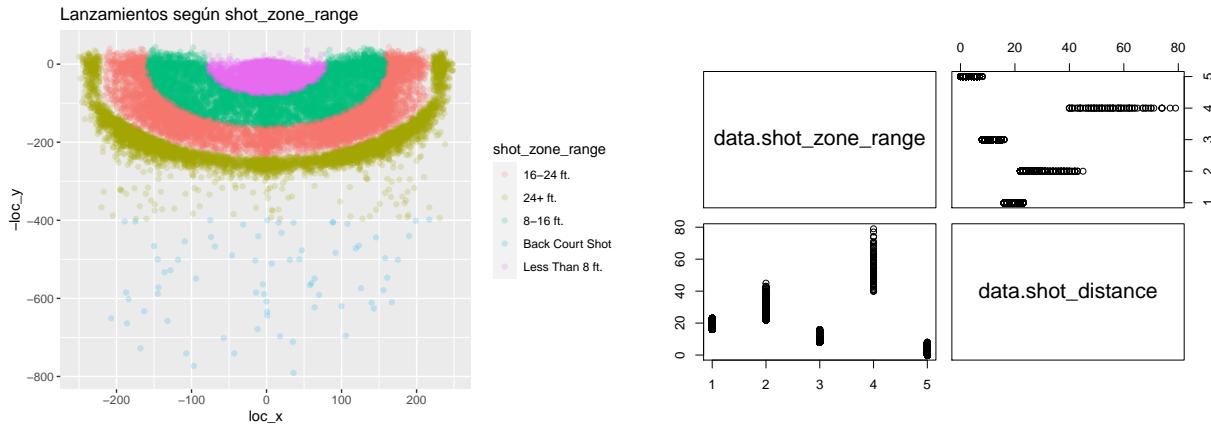
En baloncesto, la diferencia entre tiros de 2 y 3 puntos solo depende de la distancia. El gráfico de la derecha nos confirma que **shot_type está 100% correlacionada con la distancia** (además de mostrarnos algunos **valores erróneos** para esta variable). Eliminamos la variable.

- shot_zone_area y shot_zone_basic



Se trata de dos variables que nos indican la zona desde la que se ha realizado el lanzamiento. Como vemos en los gráficos, aportan cierta información adicional con respecto a la distancia como si es un lanzamiento desde el lado izquierdo o el derecho, o si estaba dentro de la zona (paint) que tiene unas reglas especiales.

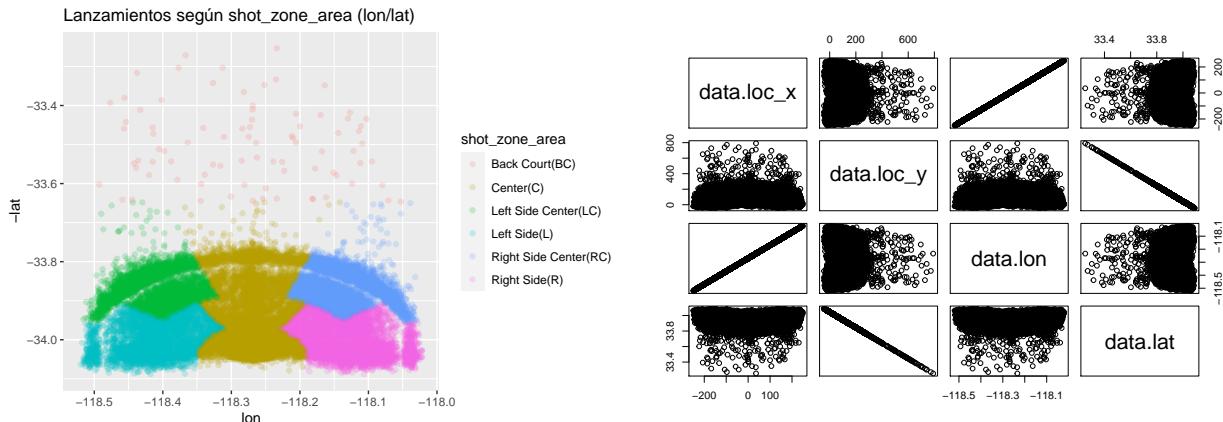
- **shot_zone_range**



Por lo que se observa en el gráfico de la izquierda, los valores de **shot_zone_range** son solo una discretización de la distancia. En la figura de la derecha lo confirmamos al ver que está correlacionada con **shot_real_distance**, por lo que **no necesitaremos shot_zone_range**.

- **lon/lat y loc_x/loc_y**

Para mostrar gráficas de las dos secciones anteriores se ha utilizado la ubicación proporcionada por las variables loc_x/loc_y. Vamos a comprobar ahora los lanzamientos según **shot_zone_area** utilizando las variables lon/lat.



Vemos en el gráfico de la izquierda que la disposición es la misma que cuando utilizamos loc_x/loc_y, aunque los valores no parecen ser relativos a la cancha. En la figura de la derecha confirmamos que los pares lon/loc_x y lat/loc_y están correlacionados.

Si hubiera que quedarse con uno de los pares loc_x/loc_y sería el elegido por incluir valores relativos a la pista. Sin embargo, finalmente **se decidió eliminar tanto loc_x/loc_y como lon/lat** ya que con la distancia de tiro y las diferentes variables categóricas shot_zone tenemos cubierta esta información de manera más simple aunque las guardaremos aparte para posteriores gráficos.

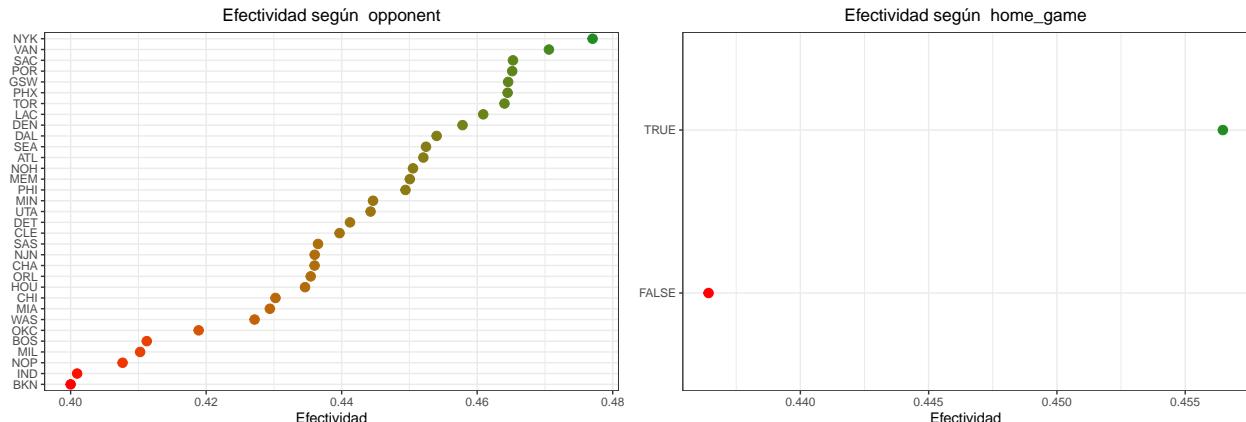
- **team_id** y **team_name**

Estas variables solo indican el equipo al que pertenecía Kobe cuando realizó los lanzamientos. En nuestro caso no aporta ningún valor, pero su presencia se puede deber a que el conjunto de datos fue extraído de un conjunto mayor en el que se encontraban todos los lanzamientos de todos los jugadores. Comprobado que es el mismo valor para todas las instancias, **quitamos las variables team_id y team_name**.

- **matchup** y **opponent**

opponent incluye una abreviatura del rival y **matchup** un texto que además informa sobre la cancha del partido. Utilizamos, por tanto, **matchup** para averiguar si es **local o visitante**.

Nota: Durante las comprobaciones comprobamos que hay 33 valores distintos en opponent y 74 en matchup. Debería haber el doble o menos en matchup (casa y visitante) lo que implica algún error en los prefijos utilizados.



Vemos que la efectividad sube ligeramente para los partidos de casa (`home_game == TRUE`). Sin embargo, **ignoraremos opponent** ya que, aunque el rival parece tener cierta incidencia, los equipos cambian mucho de una temporada a otra y las predicciones en base a esto pueden ser erróneas.

- **shot_id**

Se trata de un índice para los lanzamientos con lo que no tendrá incidencia alguna en el acierto del tiro. **Lo eliminamos** ya que no lo necesitamos para el entrenamiento

Resumen tras el preprocesamiento basado en la exploración

La dimensionalidad del conjunto de datos se ha visto reducida considerablemente tras todo el proceso anterior.

Vemos a continuación las variables iniciales del conjunto de datos:

```
## [1] "action_type"      "combined_shot_type" "game_event_id"
## [4] "game_id"          "lat"                  "loc_x"
## [7] "loc_y"            "lon"                  "minutes_remaining"
## [10] "period"           "playoffs"             "season"
## [13] "seconds_remaining" "shot_distance"       "shot_made_flag"
## [16] "shot_type"         "shot_zone_area"     "shot_zone_basic"
## [19] "shot_zone_range"   "team_id"              "team_name"
```

```

## [22] "game_date"           "matchup"            "opponent"
## [25] "shot_id"

```

Y éstas son las variables tras el análisis y posterior transformación de los datos basados en la misma.

```

## [1] "shot_made_flag"      "shot_zone_area"     "shot_zone_basic"
## [4] "combined_action_type" "last_seconds"       "last_period"
## [7] "season_ord"          "shot_real_distance" "home_game"

```

Algunas de las operaciones realizadas han sido:

Estudio de correlaciones Se han detectado correlaciones entre los siguientes pares de variables:

- `action_type <-> combined_shot_type`
- `game_date <-> game_id`
- `shot_type <-> shot_distance`
- `shot_zone_range <-> shot_distance`
- `loc_x/loc_y <-> lon/lat`

Discretizaciones u otras transformaciones

- Algunas de las que se podrían proponer ya venían incluídas en el conjunto de datos, como (`shot_type > shot_zone_range > shot_distance`) o (`combined_shot_type > action_type`).
- En cuanto al **periodo** y **tiempo restante** se ha optado por distinguir entre el último periodo o los últimos segundos, ya que es donde se apreció diferencia sustancial en la efectividad.
- Se ha realizado una normalización en la temporada (`season`) sustituyendo el año por el ordinal de la temporada en la carrera de Kobe.

Construcción de variables

- Se ha obtenido la distancia real (`shot_real_distance`) a partir de la posición de tiro, y si los Lakers jugaban en casa (`home_game`) a partir de `matchup`.
- Finalmente, se han combinado valores de `combined_shot_type` y `action_type` para formar `combined_action_type`

Procesado de variables categóricas (Dummificación) Algunos métodos de aprendizaje requieren valores numéricos en lugar de variables categóricas (factor). Este es el caso de los algoritmos de **regresión logística**. Otros no lo requieren y existen bibliotecas como caret que parecen realizar este proceso de manera implícita, en lo que se conoce como *one-hot encoding*. Sea como fuere, esta transformación no representa pérdida de información, ni añade complejidad computacional por lo que procedemos a *dummificar* o codificar las variables categóricas con la biblioteca **fastDummies**. De este modo se sustituirán por una columnas que contendrá solo ceros y unos para cada uno de los valores posibles de estas variables categóricas.

Selección horizontal y/o vertical Además de la supresión de variables predictoras realizada en los puntos anteriores, utilizaremos métodos computacionales para tratar de lograr el conjunto de variable más óptimo que nos sea posible. Para ello se hará uso de la biblioteca **MASS** y las funciones **lm** y **stepAIC**. Alguno de los métodos de aprendizaje que se utilizará a continuación como Random Forest (parámetro *importance*) incluye esta selección de variables de manera implícita pero no será así para todos ellos, por lo que realizaremos este proceso con antelación.

Finalmente, **quitaremos de nuestros conjuntos de entrenamiento y test la variable clase `shot_made_flag`**, la cual almacenaremos en una variable independiente para poder posteriormente entrenar nuestro modelo y evaluar su comportamiento.

Modelado y Evaluación

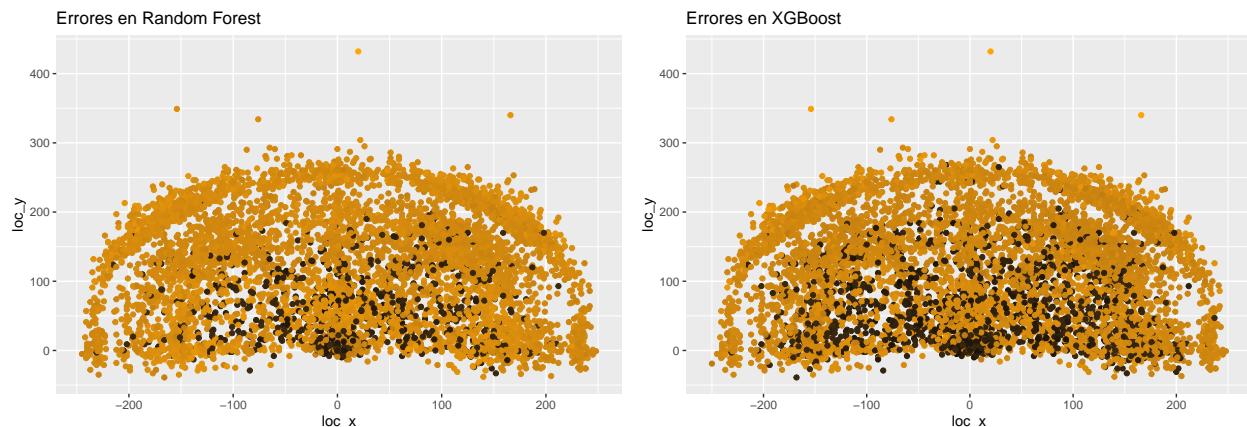
Debido a que la métrica utilizada en kaggle para evaluar nuestro modelo es LogLoss, se utilizarán modelos de clasificación probabilística. De esta manera se evitara enviar valores 0 o 1, ya que en esta métrica la penalización es muy severa cuando el error en la predicción es total en lugar de parcial, como será en las predicciones con estas técnicas.

Los métodos utilizados serán los siguientes: **LDA (Análisis discriminante lineal)**, **Regresión logística**, **Random Forest** y **XGBoost**.

Desde un principio, se comprobó que con LDA y regresión logística los resultados eran levemente menos satisfactorios que con Random Forest y XGBoost. Por tanto se incidió más en el ajuste de parámetros para los dos últimos. Mediante el uso de la función LogLoss aportada en el enunciado del problema, se obtuvieron los siguientes resultados para la métrica **LogLoss** en la evaluación de **nuestros modelos** con los **datos de entrenamiento**:

```
## [1] "LDA" : 0.611957641488579"  
## [1] "Regresión logística": 0.610874893143176"  
## [1] "Random Forest" : 0.537269933601782"  
## [1] "XGBoost" : 0.585760920675654"
```

Por lo tanto, centrándonos en Random Forest y XGBoost, intentaremos visualizar en los gráficos siguientes dónde se produjeron los **errores más importantes** de nuestra predicción.



Solo se han renderizado puntos en los que el fallo en la predicción ha sido superior a 0.5:

- En los puntos de color **naranja**, se **predijo** un porcentaje **inferior al 50%** de canasta y el lanzamiento fue **exitoso**
- En los puntos de color **negro**, se **predijo** un porcentaje **superior al 50%** de canasta y el lanzamiento fue **fallido**

Se pueden extraer varias **conclusiones** de estos gráficos:

- Hay **más errores** en el modelo de **XGBoost**, lo cual no es sorprendente ya que el resultado de su evaluación fue peor.
- Hay **más errores de lanzamientos que se predijeron como fallo** y finalmente terminaron en canasta (naranja) que el caso contrario (negro).
- La mayoría de lanzamientos predichos como exitosos que fueron **fallados** (negro) se encuentran **bajo la canasta**. Esto posiblemente se debe a **tapones recibidos**, los cuales son imposibles de predecir con el conjunto de datos utilizado.

Conclusiones

El proyecto se ha realizado durante las **tres primeras semanas de mayo**. Se han dedicado un total de **50 horas** a su realización.

La sensación al comienzo del proyecto fue de que eran demasiados aspectos nuevos que aprender y poner en práctica y el alumno se sintió en cierto modo **sobrepasado**. Sin embargo, los comentarios en el **foro** de la asignatura y la documentación aportada por otros usuarios en *kaggle* mediante los *notebooks* publicados fue de gran ayuda para ir progresando en los diferentes pasos requeridos en el proyecto.

El **análisis exploratorio**, la **creación de gráficos** y la **documentación** son los aspectos que han requerido de **mayor esfuerzo**.

Por otro lado, se podría haber empleado más tiempo en búsqueda de otros métodos de aprendizaje o en parametrizar de una manera óptima los finalmente utilizados, así como en el apartado de evaluación, en el cual se visualizan gráficamente los errores pero no se han podido detectar soluciones.

La documentación ha sido realizada mediante **RMarkdown**, lo cual ha resultado un acierto y será sin duda el formato a utilizar en futuros proyectos similares. Es cierto que la generación del documento final requiere de varios minutos una vez incluye todo el contenido. Sin embargo, la integración de comentarios, gráficos, código y salida por pantalla en la generación del documento desde cero lo hacen el formato perfecto para un proyecto como este.

El **código fuente** del proyecto se incluye dentro del fichero de RMarkdown. Está disponible en este enlace.

Resultado en Kaggle

Finalmente se presenta en este proyecto el resultado del modelo creado con **XGBoost**. Esto es debido a que se trata del modelo que mejor puntuación (*score* más bajo) ha obtenido en kaggle, lo cual es de algún modo sorprendente ya que el modelo de Random Forest obtenía mejores resultados en nuestra evaluación. Estos han sido los resultados de los diferentes modelos:

- **LDA:** 0.61797
- **Regresión logística:** 0.61583
- **Random Forest:** 0.60663
- **XGBoost:** 0.60511

El usuario utilizado para cargar las predicciones es **gustavoquevedo** y a continuación se pueden ver el *score* final obtenido:

Name	Submitted	Wait time	Execution time	Score
20200523_xgboost.csv	15 hours ago	0 seconds	0 seconds	0.60511
Complete				

Referencias

- Creación modelo de regresión logística. Utilizado para el uso de la función *glm*
- Explicación de por qué la clasificación probabilística es mejor para mejorar el LogLoss
- Notebooks en la competición Kobe Bryant Shot Selection. Se han observado y sacado ideas en general de varios de ellos
- Notebook con un modelo creado en XGBoost, en el que se basó la solución propuesta