Predicción del resultado de los partidos de La Liga

Enrique Rodríguez Morón

29 junio 2018

Contenido

1. Introducción

El objetivo de este proyecto es **predecir el resultado de un partido de La Liga dada la alineación**.

Para ello, se utilizará los ficheros en https://www.kaggle.com/hugomathien/soccer con los resultados de las principales 6 ligas europeas desde el año 2008 al 2016. Entre estos fichero se encuentran:

- **Match o Partido**: contiene información sobre los partidos y, sobre todo, ids a las otras tablas:
  - Los ids de los equipos diferenciándolos entre local y visitante
  - La temporada y la fecha del partido
  - Los ids de los jugadores del 11 inicial del partido
  - La posición (X, Y) de cada jugador en el terreno de juego. Estos no serán utilizados en un principio pero, a partir de ellos, se podría adivinar la formación.
  - Lo que pagan algunas casas de apuestas por la victoria local, visitante o empate. Tampoco se está interesado en estos atributos
  - Estadísticas del partido. Puesto que se quiere adivinar el resultado del partido antes de jugarse, no se dispondrá de esta información por lo que se eliminará.
- **Team o Equipo**: contiene información sobre los equipos:
  - Id del equipo
  - Nombre
- **Team Attributes o Atributos de los equipos**: contiene ciertos atributos de los equipos los cuales se pueden ver en https://sofifa.com/squads/top:
  - La gran mayoría de los atributos categóricos son atributos calculados a partir de los numéricos.
  - Los numéricos son tales como agresividad y presión en defensa, rapidez de juego, regates, pases… Todos ellos están entre los valores 0 y 99.
  - Hay tres atributos categóricos que define la forma de jugar del equipo: BuildUpPlayPositioning - posicionamiento en plan de juego (organizado o forma libre), ChanceCreationPositioning - posicionamiento en ocasiones (organizado o forma libre) y DefenceDefenderLine - defensa (cubrir o fuera juego)
  - Cada fila tiene una fecha donde se establecieron dichos valores.
- **Player o Jugador**: contiene información sobre los jugadores:
  - Id del jugador
  - Nombre
  - Fecha nacimiento
- **Player Attributes o Atributos de los jugadores**: contiene ciertos atributos de los jugadores los cuales se peden ver en https://sofifa.com/players/top:
  - Hay atributos específicos para porteros (estirada, paradas, saques, colocación, reflejos), genéricos (general (muy importante), potencial), ataque (centros, definición, precisión de cabeza, pases

cortos, voleas), técnica (regates, efecto, precisión faltas, pases largos, control del balón), moviento (aceleración, velocidad, agilidad, reflejos, equilibrio), potencia (potencia, salto, resistencia, fuerza, tiros lejanos), mentalidad (agresividad, intercepción, colocación, visión, penaltis, compostura) y defensa (marcaje, robos, entrada agresiva). Todos ellos están comprendidos entre 0 y 99.

- Tiene 3 atributos categóricos: trabajo en ataque, trabajo en defensa, y pierna preferida.
- Cada fila tiene una fecha donde se establecieron dichos valores.

2. Primera iteración

En este apartado se muestra los resultados después de la primera iteración.

2.1. Carga de datos y limpieza

El primer dataset que se leerá es **Partidos**.

Es importante destacar que:

- Los ids de los jugadores locales vienen en las columnas llamadas "home*player*%i" donde %i es un número entero entre 1 y 11. Lo mismo para los visitantes: "away*player*%i".
- No se dice quien es el portero, los defensas, medios o delanteros, sino que se tiene que adivinar la posición X e Y en el campo de cada jugador con las columnas "home_player_X%i"/"away_player_X%i" y "home_player_Y%i"/"away_player_Y%i". El jugador con %i==1 corresponde al portero, los siguientes %i son los defensas, luego los medios y por último los delanteros.
- Para saber la posición en el campo, imagínese un campo vertical donde el córner de abajo a la derecha es la posicion (x=1, y=1) siendo el portero siempre la posicion y=1. En y=3, son los defensas donde el que está en x=2 estará más a la derecha del portero que el de la posicion x=4. Como máximo es x==9 e y==11.
- Se puede adivinar la formación del equipo con las posiciones de los jugadores sumando los jugadores que están en la misma y desde y>1 (ya que el y=1 es para el portero). Por ejemplo la formación 4-4-2 del local sería: home_player_X2=2, home_player_Y2=3, home_player_X3=4, home_player_Y3=3, home_player_X4=6, home_player_Y4=6, home_player_X5=8, home_player_Y5=3, home_player_X6=2, home_player_Y6=7, home_player_X7=4, home_player_Y7=7, home_player_X8=6, home_player_Y8=7, home_player_X9=8, home_player_Y9=7, home_player_X10=3, home_player_Y10=9, home_player_X11=6, home_player_Y11=9.

No obstante, los atributos de la tabla partidos son, casi exclusivamente, mapeos a otros datasets para obtener los datos de los jugadores y equipo.

Además, como se esta interesado únicamente en La Liga, se seleccionan sólo los partidos de esta liga europea.

Como se ha dicho, se dispone de los goles que ha marcado cada equipo, pero no el resultado: quien ha ganado o si se ha empatado (es decir la clase a la que pertenece el partido y se quiere adivinar). Por tanto, se añade el resultado como un valor string: "0" si ha ganado el equipo local, "1" si es empate, "2" si ha ganado el visitante. Este dato se conoce en todos los partidos.

Puesto que en el deporte, y sobretodo en el fútbol, suele haber rachas en las que un equipo suele ganar o perder, se añadirá (antes de eliminar cualquier partido por falta de información sobre los jugadores) la información de los últimos 5 partidos que ha jugado cada uno en la temporada correspondiente. También suele influir si se juega en casa o no, por lo que:

- Para el equipo local se añadirá la información de los últimos partidos locales y los últimos 5 independientemente si es local o visitante.
- Para el equipo visitante se añadirá la información de los últimos partidos fuera de su estadio y los últimos 5 independientemente si es local o visitante.

Se tiene que tener en cuenta que los resultados de primeros partidos de una temporada y los últimos pueden ser menos fiables que el resto de la temporada ya que están todavía de pretemporada o al final hay muchos que no se juegan nada. Por tanto los primeros 5 primeros y 5 últimos no ponderán igual que el resto. Así, si una temporada tiene 38 jornadas, las ponderaciones serán:

- jornadas 1 y 37: 0.75
- jornadas 2 y 36: 0.80
- jornadas 3 y 35: 0.85
- jornadas 4 y 34: 0.90
- jornadas 5 y 33: 0.95
- resto de las jornadas: 1.00

Examinando los datos, tenemos que hay 3040 repartidos en 8 temporadas consecutivas (siendo la primera 2008/2009) y cada una de ellas tiene el mismo número de partidos: 380.

Todos los jugadores pueden tener más o menos las mismas características excepto uno: el portero. Si falta por saber algún jugador en el 11, sus características pueden hallarse con la media de los otros, pero el portero tiene atributos específicos para esa posición. Por lo tanto, si falta el id del portero dicho partido se elimina.

Por otro lado, se tiene que comprobar que la posición de los porteros (acuérdese que es el número 1) tiene que ser (x=1, y=1). Si no fuese así, estos partidos también se tendrían que eliminar. Se Se tienen 2 partidos en los que esta posición es incorrecta por lo que se eliminan.

Al querer adivinar el resultado del partido a partir de los datos de los jugadores, se necesita saber el 10 inicial de los dos equipos (sin el portero ya que se ha limpiado ya). Si no se sabe, hay dos opciones:

- Eliminar dicho partido
- Dependiendo del número de jugadores que falten, se podrían sustituir los valores de los que faltan con la media de los demás.

La primera opción elimina 333 filas del total del dataset (3040), lo que supone casi un 11%.

Para la segunda opción se va a poner como condición que, como máximo, falte 1 jugador de cada equipo. Si faltan más, esa fila será eliminada.

De esta manera se eliminan sólo 29, por lo que se utiliza este método ya que no se dispone de muchos partidos para entrenar los modelos.

El siguiente paso es leer el fichero **Player** que contiene el nombre del jugador y el año de nacimiento. Se está interesado en este atributo para saber su edad e incluirlo en el modelo ya que puede afectar al rendimiento de los jugadores.

También se lee **Player_Attributes** que contiene los atributos de los jugadores. Dado que un jugador puede ir cambiando e ir mejorando o empeorando, se tiene una columna fecha para el valor de esos atributos.

Se tiene que tener en cuenta que estas características son de los jugadores de las 6 ligas top en europa, lo que quiere decir que no se puede extrapolar a La Liga: según dicen los expertos, La Liga es la mejor liga del mundo, por lo que si se sustituyesen los valores de los jugadores de La Liga que se desconocen por la media de todos, no sería muy correcto.

Se observa que todos los atributos tienen una puntuación entre 1 y 99 (como se esperaba) y la gran mayoría tienen valor: el que más nulos tiene es "defensive_work_rate" con 7659 filas de un total de 183978, esto es 4%, una proporción pequeña (una vez se han limpiado los atributos como se dice en el siguiente párragrafo).

Por último, las columnas "attacking_work_rate" y "defensive_work_rate" tienen tres posibles valores: "low", "medium", "high"; y "preferred_foot", 2 valores: "right", "left". Hay algunas filas con un valor distinto a estos valores por lo que el valor en estas filas se pone a None.

Se comprueba que todos los porteros de todos los partidos tienen, al menos, un registro en Player_Attributes. Si no fuese asi, sería como portero desconocido y se tendría que eliminar el partido.

A continuación se añaden los atributos de todos los jugadores, tanto locales como visitantes, al dataset con los partidos. Se coge la fila de los atributos de cada jugador con la fecha más cercana al partido.

Se tiene en cuenta que los atributos para el portero son distintos que para el resto de jugadores, por lo que se añadirán los específicos para el portero (todos los que empiezan por "gk_" más los comunes: overall_rating y potential) y para el resto, todos los atributos excepto los de porteros: una nueva columna por cada jugador y cada atributo. También se añade la edad de cada jugador.

El siguiente paso es limpiar los partidos donde no se tengan los sufientes valores de los atributos de los jugadores no porteros de cada equipo en un partido. Por cada equipo por cada partido:

1. Se mira que el número de valores nulos de todos los atributos del portero no supere un umbral (por defecto es 3). Si lo supera, este partido no cumple con los objetivos y tiene que ser eliminado. Si no supera dicho umbral, pasa al siguiente paso.
2. Se obtiene el mismo atributo de todos los jugadores no portero de este equipo en este partido
3. Si el número de valores nulos de este atributo para los 10 jugadores es superior al umbral (por defecto es 3), se elimina el partido.
4. Si no es superior al umbral, se obtiene el valor medio si es numérico y el más frecuente si es categórico.
5. El valor obtenido en el punto anterior es asignado a todos los valores nulos de este atributo para los jugadores no portero de este equipo en este partido.

Es decir chequea tanto los atributos del portero como los que no son portero, pero solo sustituye estos últimos en caso de ser nulos.

Se han sustituido los valores de los jugadores no portero que faltaban pero no los de los porteros ya que hay un único portero por equipo y no se puede obtener el valor medio. Ninguno de los atributos de los porteros en los partidos falta pero, si los hubiese, se seguirían los siguientes pasos:

1. Se obtienen todos los partidos del equipo. Si alguno tiene valor para este atributo, se obtiene el valor medio y se asigna a este atributo-partido.
2. Si no se tiene ningún registro, se sustituiría por el valor medio de todos los partidos en ese atributo.

Por último se añade la formación de los dos equipos con el formato "%d-%d-%d" donde %d es un número. Puede ser que el equipo juegue con más o menos líneas por lo que el formato será distinto, por ejemplo "4-1-4-1". Este atributo puede ser interesante porque dice mucho de la forma de jugar de cada equipo.

Puesto que se tienen muchos atributos de los jugadores no porteros para un partido (33 atributos * 10 jugadores * 2 equipos), se va a hacer un promedio por equipo en el partido de cada atributo, por tanto se tendrán 33 atributos * 2 eqipos.

Se ponderará con un mayor peso a los atributos que corresponden a jugadores top ya que normalmente los mejores jugadores suelen influir más en el juego del equipo. Para saber si un jugador es top o no, se mira la columna "overall_rating" de player_attributtes. Se obtiene el percentil 0.95 de la columna "overall_rating" de todos los jugadores no porteros en los partidos de La Liga. Se ha elegido hacerlo así y no directamente sobre los player_attributtes porque se está analizando sólo la liga española y en player_attributtes están los atributos de todos los jugadores de las 6 ligas europeas más importantes. Este valor es 85'6.

Una vez obtenido el overall_rating a partir del cual un jugador no portero es considerado top, se ponderará sus atributos un 5% más que el resto. El algoritmo es:

1. Se obtiene el número de jugadores top en cada partido-equipo
2. Se calcula el rate para los jugadores top y no top
    A. Cada jugador aporta un 10% ya que seria 100% / 10 jugadores no portero por equipo
    B. Se calcula el porcentaje que aportaria un jugador top, esto es 10% * 1.05
    C. Si todos los jugadores son top, no se puede superar el 100% por lo que todos aportarían igual: 10%
    D. Se obtiene el porcentaje restante para los jugadores no top: 100% - 10% * 1.05 * número jugadores equipo top
    E. El % obtenido en el punto anterior se divide entre el número de jugadores no top
    F. Se multiplica cada atributo por el ratio dependiendo de si es jugador top o no y se suma

Después se eliminan los atributos individuales de los jugadores no portero.

Se añaden los **atributos del equipo**. Esta tabla tiene los atributos de los equipos y, como pasaba con los atributos de los jugadores, tiene una columna con la fecha del valor de los atributos.

Tiene una columna, "buildUpPlayDribbling", que tiene 969 nulos en las 1458 filas, es decir, alrededor del 66% están vacías. Por tanto se decide eliminar dicha columna.

Se van a deshechar la gran mayoría de las columnas cuyo nombre terminan en "Class" ya que son variables categóricas construidas a partir de las numéricas: pertenece a una clase u otra en función de los valores de su campo numérico. Los únicos que no tienen valor numérico, y por tanto se mantendrán son: "buildUpPlayPositioningClass", "chanceCreationPositioningClass", "defenceDefenderLineClass". También se comprueba que los valores en estas columnas son correctos y no pasa como con los de jugadores donde algunas columnas tenían valores no correctos.

Se ha decidido convertir estas tres variables categóricas en variables dummy ya que:

- Son solo 3 variables cada una con únicamente 2 valores
- Pueden ser importantes ya que describen el juego del equipo, la táctica
- No hay relación entre ellas

El último paso es convertir las variables categóricas "preferred_foot", "attacking_work_rate", "defensive_work_rate" y las formaciones. Todas se convertirán en valores numéricos ya que:

- "attacking_work_rate" y "defensive_work_rate" tienen valores escalonados por lo que se sustituirán por valores consecutivos.
- las formacions tienen un total de 22 posibles valores. Si se trasformasen a columnas dummies se crearían muchas columnas (22 * 2 equipos) donde 21 * 2 de ellas tendrían un valor igual a 0. Además los valores de estas columnas son variables y puede aumentar.

- "preferred_foot" no tiene mucho sentido una vez que se han juntado todos los atributos de los jugadores. A pesar de esto, en vez de deshecharlo se cambia a valor numérico y luego se mirará si es importante o no.

2.2. Relación entre las columnas

A continuación se muestra la matriz de correlación entre las columnas. Para ello:

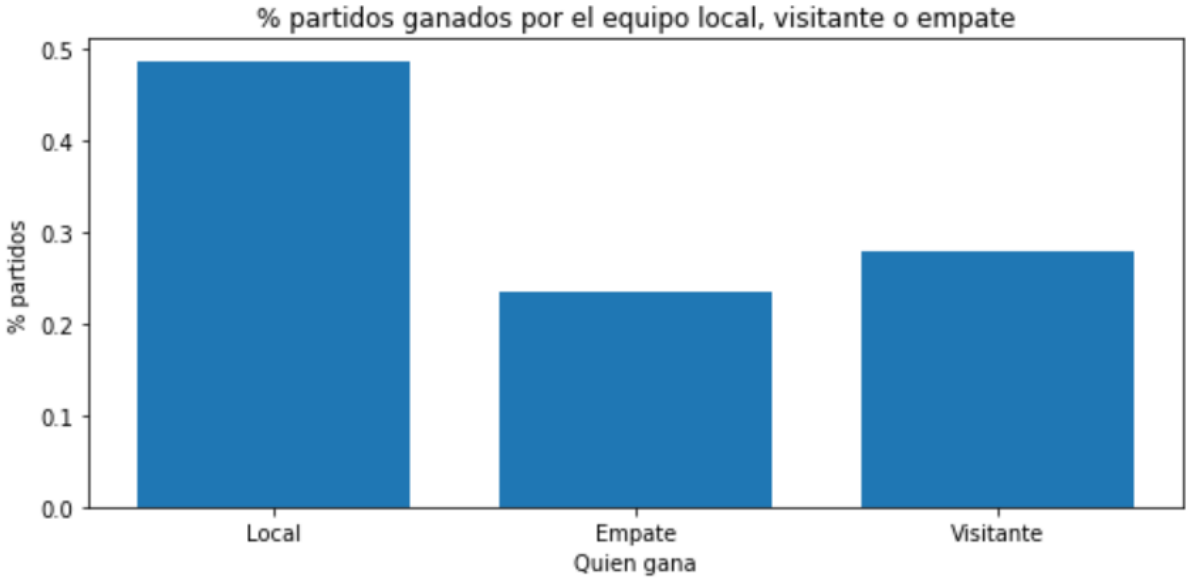
1. Se eliminan las columnas en match que no se pasarán al modelo, tales como los ids (jugadores, equipos, liga, partido), fecha del partido, temporada, posiciones de los jugadores, goles del equipo local y visitante. La jornanda se mantiene ya que podría influir como se ha dicho anteriormente.
2. Se añade la columna "result_int" con el resultado en formato numérico. Esto se hace así para ver si se tiene una relación entre algún atributo y el resultado. Después se elimina del dataframe.

No hay ninguna columna que esté relacionada con el resultado del partido. Si se miran los atributos equipo local y equipo visitante, el atributo que mayor correlación es "overall_rating" siendo mayor en el visitante (casi 0.33) que el local (-0.28). Es decir, que **la media de los jugadores es más importante fuera de casa que en casa** a la hora de pronosticar el resultado.

A continuación se eliminarán las columnas que están muy correladas, esto es, mayor que 0.90 en valor absoluto y que no son las variables dummies entre ellas (ya que la relación es 1.00). Por comodidad no se muestran las columnas ya que son 110 pero decir que casi todos los atributos que se eliminan están relacionados con el atributo overall_rating de su equipo. Esto parace lógico ya que este atributo es un resumen de todos los atributos de los jugadores.

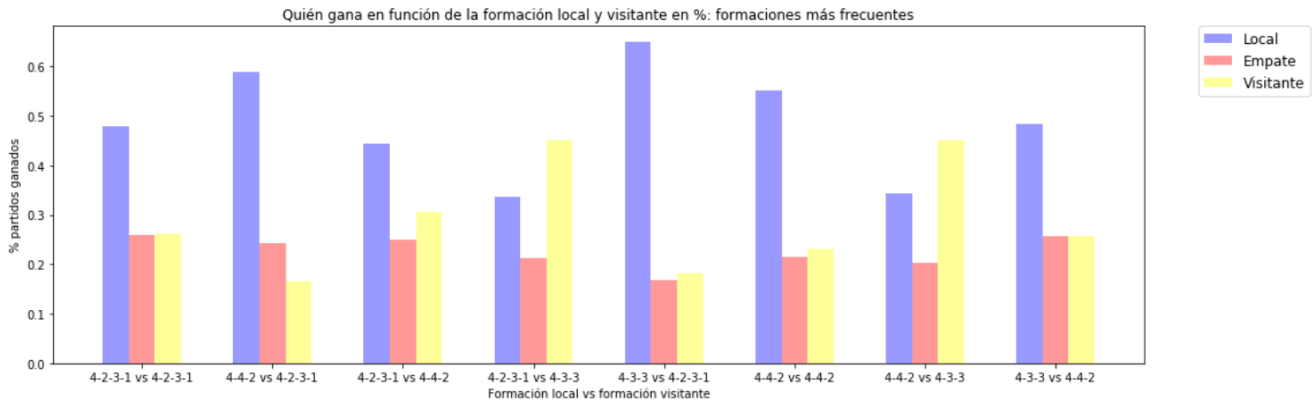
Por último en este apartado se van a mostrar algunas **gráficas**.

Empezaremos viendo la proporcion entre partidos que son ganados por el equipo local, por el visitante o empate:



Como se puede ver, el factor estadio es muy importante ya que casi la mitad de todos los partidos son ganados por el local. El resultado menos frecuente es el empate.

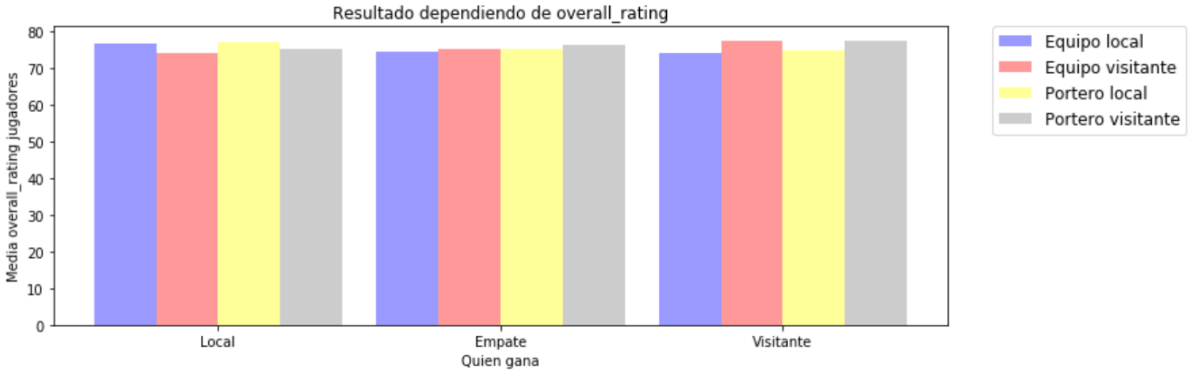
Ahora se va a crear un gráfico donde se muestre la formación de cada equipo "local vs visitante" y ver si hay algún patrón de comportamiento. Para ello se va a seleccionar las 8 formaciones "local / visitante" que más se repiten:



Primeramente se observa que la combinación más frecuente con mucha diferencia es la "4-2-3-1" en ambos equipos. Estas formaciones son un tanto conservadoras ya que se juega con 2 medios defensivos y un solo delantero.

Las únicas formaciones donde el equipo visitante tiene mayor porcentaje es con un "4-3-3" aunque no son muy representativas: cerca del 11%. Por su parte, esta formación ("4-3-3") en el equipo local tiene la mayor diferencia Local vs Empatados+Visitantes que en las demás combinaciones.

Por último se mostrará un gráfico donde se compara la media del atributo overall_rating de los jugadores no porteros/porteros (por separado) de los equipos locales y visitantes por resultado:

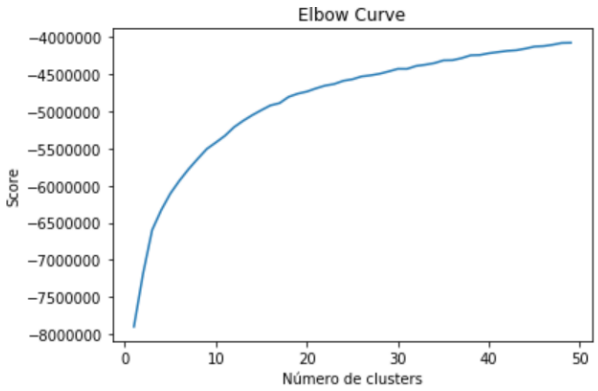


Se puede observar que cuando un equipo gana la media tanto del equipo (sin incluir al portero), como el del portero es superior al otro equipo. Pero para conseguir un empate, el equipo visitante tiene que tener mayor overall_rating.

2.3. Machine Learning: No Supervisado

Se va a utilizar Kmeans como algoritmo no supervisado para intentar ver si hay alguna relación entre algunos atributos o si para unos determinados valores el comportamiento es muy parecido.

Primero se utiliza el método de Elbow para ver el número óptimo de clusters:



Voy a elegir 8 clusters. Se entrena con la totalidad de los partidos. No se mostrarán los valores de los atributos de los centroides para no extender el documento; si se quisiese ver, sólo sería descomentar el código.

Vamos a agrupar por los centroides y por el resultado y ver si se puede sacar alguna conclusión donde "result" tiene los siguientes valores: "0" - gana el local, "1" - empate, "2" - gana el visitante.

		Centroide id							
		0	1	2	3	4	5	6	7
Resultado	0	182	314	247	249	232	117	48	38
	1	135	194	94	22	69	95	27	52
	2	159	167	82	17	50	122	33	190

- Centroides 3: predomina la victoria visitante:
  - Los overall_rating son bastante mayores
  - El posicionamiento del equipo local es buen formado frente al libre posicionamiento del visitante.
  - Es decir, es el tipo de partidos donde uno de ellos (en este caso el local) es peor a nivel de jugador pero tácticamente muy bueno, frente a un equipo con buenos jugadores pero sin construir.
- Centroides 0, 2, 6: predomina la victoria local:
  - En ambos la diferencia entre el equipo local y visitante en el atributo overall_rating hay una gran diferencia.
  - Las formaciones son prácticamente las mismas en el local y visitante: 4-4-2 vs 4-4-2 y 4-3-3 vs 4-2-2 en el centroide 6.
- Centroide 5: predomina la victoria local pero contiene muchos partidos empatados si se compara con las demás clases:
  - La diferencia en el atributo overall_rating es mínima, casi nula.
  - A pesar de esto, el número de partidos ganados por el local es bastante alto.
  - Será difícil adivinar si el partido es un empate ya que no se tiene un atributo claramente que lo identifique.

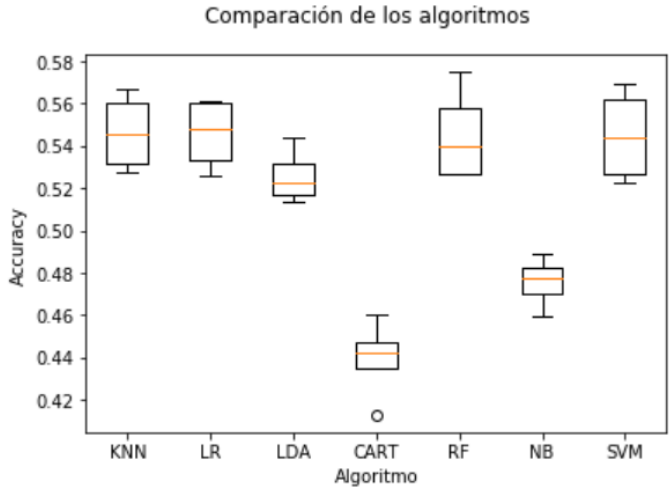
2.4. Machine Learning: Supervisado

En este apartado se van a entrenar varios modelos supervisados y compararos, para luego elegir el mejor.

Para ello, lo primero que se hará es separar el dataframe en dos: uno (X) con todas las columnas excepto la objetivo y otro (Y) con solo el objetivo. Luego se estandarizan todas las columnas de X.

Se van a entrenar varios modelos con distintos parámetros en cada uno y luego mostrar los mejores resultados. Para ello se utilizará GridSearchCV con un cross-validation igual a 4 para todos ellos. Después, con los mejores parámetros de cada algoritmo, se entrenan con cross validation con 4 folds obteniendo así la media y la desviación para mostratlo en boxplots:

```
KNN -> media: 0.546501, desviación: 0.016510
LR -> media: 0.545823, desviación: 0.015250
LDA -> media: 0.525722, desviación: 0.011419
CART -> media: 0.439524, desviación: 0.017104
RF -> media: 0.545138, desviación: 0.019970
NB -> media: 0.475640, desviación: 0.010922
SVM -> media: 0.545137, desviación: 0.019982
```



Como se puede observar hay 5 algoritmos con mejor resultado: KNN, LR, LDA, RF, SVM. Teniendo esto en mente, se crearán las matrices de confusión, curvas ROC y, después, se escogerá uno de ellos para ponerlo en producción.

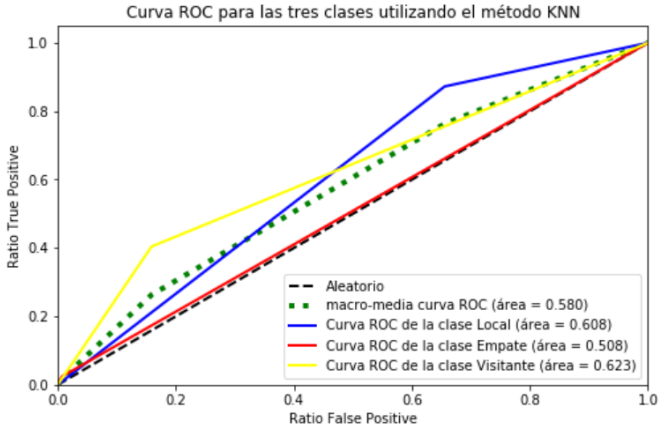
El primer paso es dividir X e Y entre conjunto de entrenamiento y de test con una relación 70/30 y luego se entrenan los algoritmos seleccionados con el conjunto de entrenamiento y se evalúan con el conjunto de test. Se muestran las matrices de confusión y el accuracy por cada algoritmo diferenciando entre test y entrenamiento:

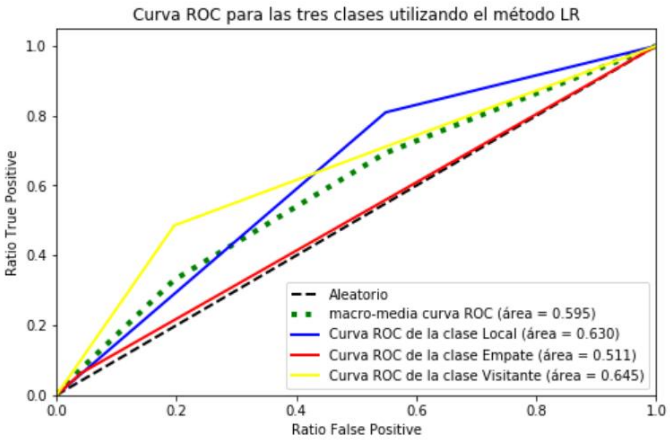
```
_____________
KNN
Test accuracy 0.545:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                376                3                  52
Verdadero Empate               151                5                  49
Verdadero Visitante            144                2                  99
_____________
LR
Test accuracy 0.546:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                349               17                  65
Verdadero Empate               132               13                  60
Verdadero Visitante            115               11                 119
_____________
LDA
Test accuracy 0.532:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                337               41                  53
Verdadero Empate               129               19                  57
Verdadero Visitante            107               25                 113
_____________
CART
Test accuracy 0.445:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                238              101                  92
Verdadero Empate                91               52                  62
Verdadero Visitante             95               48                 102

Entrenamiento accuracy 0.977:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                983                6                   7
Verdadero Empate                11              467                   5
Verdadero Visitante              8               11                 556
_____________
RF
Test accuracy 0.558:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                370                9                  52
Verdadero Empate               147               14                  44
Verdadero Visitante            131                6                 108

Entrenamiento accuracy 1.000:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                996                0                   0
Verdadero Empate                 0              483                   0
Verdadero Visitante              0                0                 575
_____________
NB
Test accuracy 0.465:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                205              165                  61
Verdadero Empate                52               98                  55
Verdadero Visitante             38              100                 107
_____________
SVM
Test accuracy 0.551:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                375                5                  51
Verdadero Empate               155                5                  45
Verdadero Visitante            137                3                 105
```

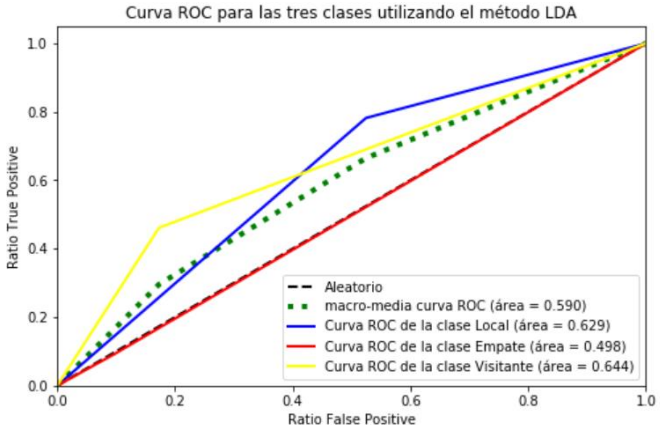
En los que mejor resultado dieron en el cross validation son los que peor clasifican los empates. De estos el que "mejor" clasifica los empates es LDA.

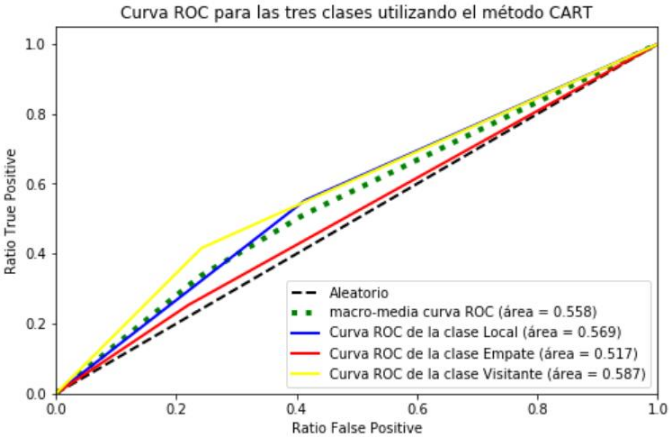
Resulta curioso el caso de los dos árboles de decisión ya que clasifican (casi) todos los ejemplos del conjunto de entrenamiento, es decir, están sobre-entrenados ya que con el de test obtiene un accuracy cerca de 0.45 uno y 0.55 otro. No obstante, los resultados del segundo (Random Forrest), siguen siendo bueno si se comparan con los demás métodos.

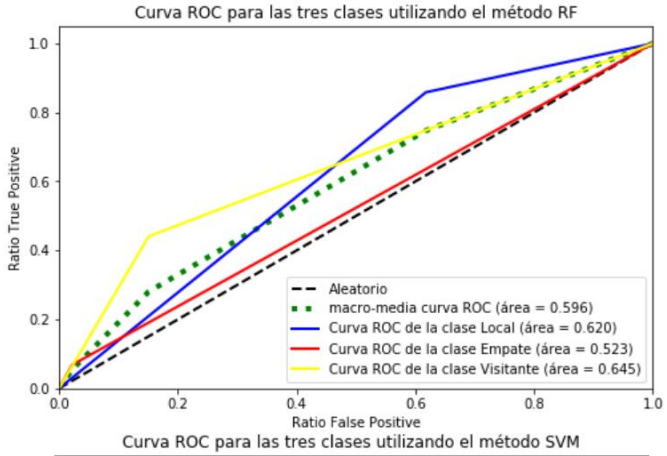
Ahora se mostrará la curva ROC de todos los modelos:

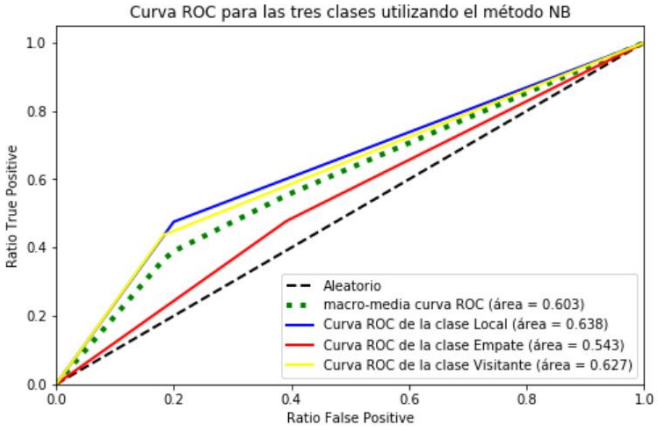


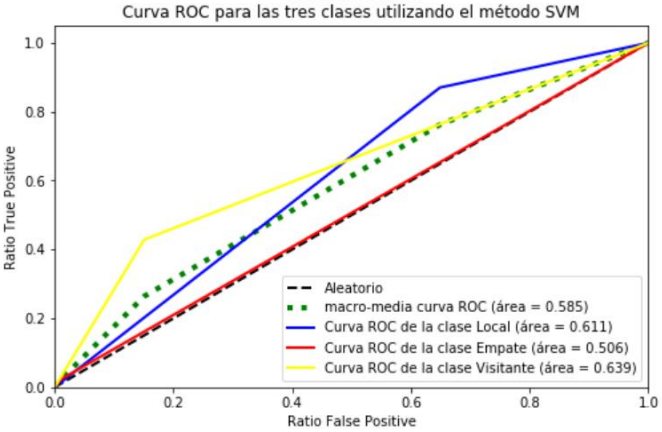












Casi todos los gráficos son iguales, los únicos que no lo son: Decision Tree y NB. Se va a hacer una nueva iteración de todo el ciclo pero si a estas alturas se tuviese que elegir un método sería LDA:

- El accuracy de los 5 top es muy parecido: entre 0.53 y 0.55

- Este es el metodo de todos estos que mejor clasifica los empates

3. Segunda iteración

Como se ha visto, los resultados no son muy buenos por lo que se intenta rehacer el trabajo.

3.1. Carga de datos y limpieza

Se hace lo mismo que en la primera excepto:

- Se han eliminado "attacking_work_rate" y "defensive_work_rate" ya que son datos calculados por la página web a partir de los atributos del jugador y los clasifica en bueno, normal o malo en ataque y defensa. Por otro lado, son los atributos con más valores nulos. También se elimina el atributo "preferred_foot" ya que a nivel de análisis de jugador podría ser interesante pero no por equipo.
- Para la reducción de todos los atributos de jugadores no porteros por cada equipo y partido:
    - Hay atributos de los jugadores específicos para ataque, otros para la defensa y los demás para todo el equipo tal y como se ha descrito en la introduccion.
    - Como se tiene la formación de cada equipo, se sabe la posición de cada jugador.
    - Para los atributos específicos de los delanteros sólo se tendrán en cuenta los delanteros (o incluso algunos medios en función de la formación). Asi, por ejemplo, para la formación 3-4-1-2, se calcula la media de cada atributo de ataque de los delanteros (los dos de arriba) y se multiplica por 0.85; luego se suma el valor de este atributo del jugador detrás de los delanteros multiplicado por 0.15; es decir para los atributos de ataque en esta formación los delanteros aportan 85% y el jugador de atrás de ellos el 15%. Los valores se pueden ver en la variable FORMATION_RATE_PLAYERS.
    - Lo mismo se hacen para los atributos específicos de defensa.
    - Para los penalties, se utiliza la media de los 3 mejores ya que, normalmente, sólo los mejores en este aspecto los lanzan.

3.2. Relación entre las columnas

En este apartado se obtienen los mismos resultados que en la anterior iteración.

3.3. Machine Learning: Supervisado

Como en la anterior, se entrenarán varios modelos. Esta vez se utilizará Recursive Feature Elimination (RFE) para reducir el número de columnas en la entrada de los algoritmos. También se utilizará grids como se hizo en la anterior iteración.

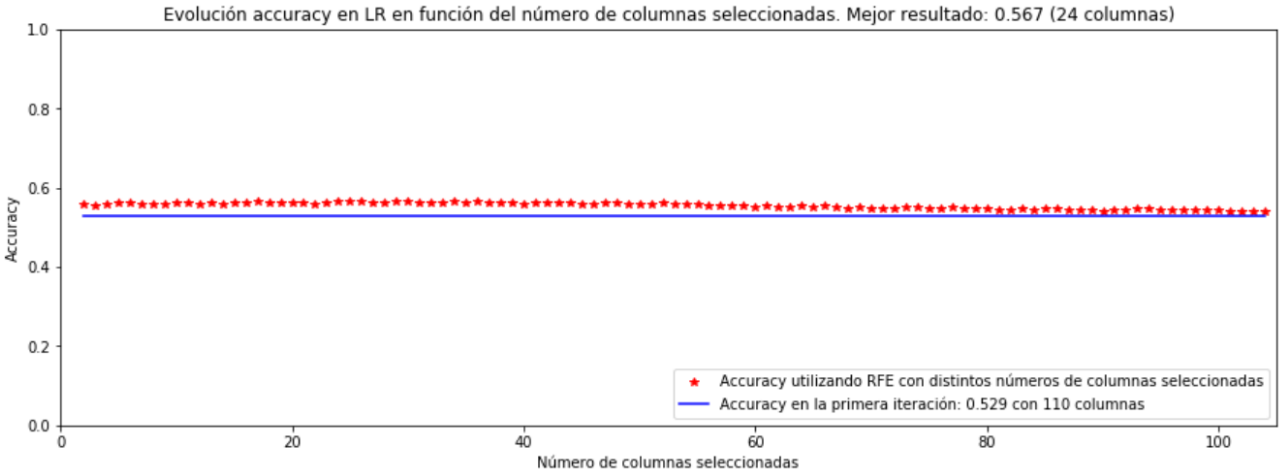
Primer paso: separar el dataframe en dos: por un lado todas las columnas menos la objetivo, y por otro sólo el objetivo; y se estandariza la primera.

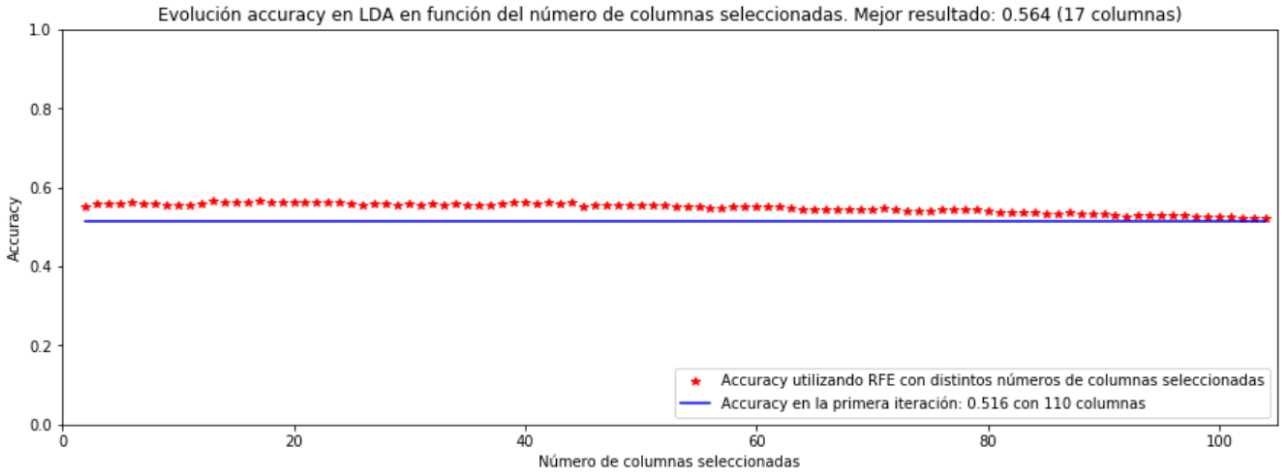
Para utilizar **Recursive Feature Elimination (RFE)** el estimador tiene que tener los atributos coef_ o feature*importances* por lo que no todos sirven. Se le tiene que decir el número de columnas que se quieren obtener y RFE obtiene los mejores para obtener el mejor accuracy. Esto lo consigue eliminando atributos de manera recursiva. De todos los anteriores se seleccionan: LR, LDA y CART.

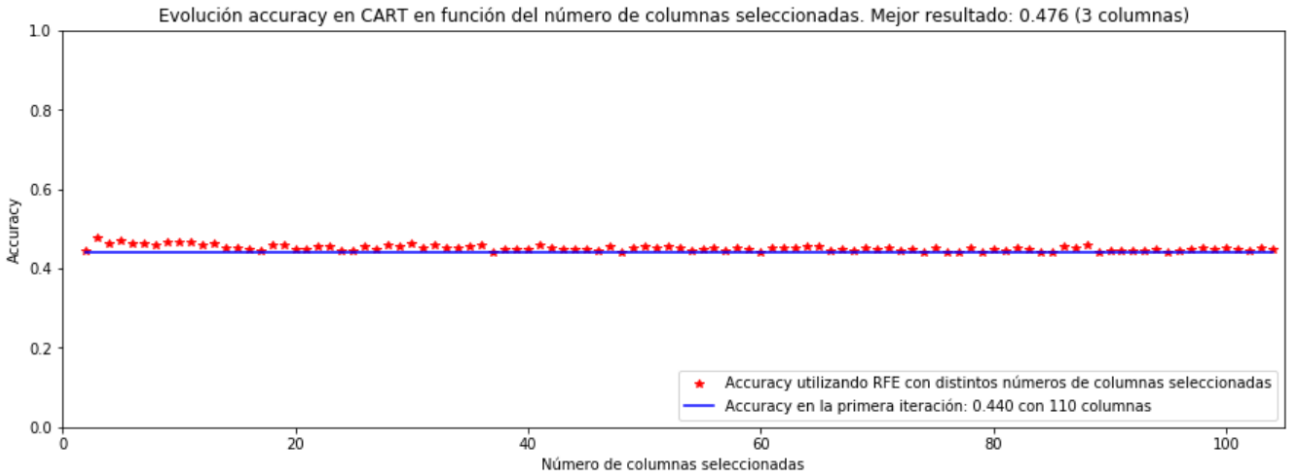
Los pasos que se darán:

1. Se recorre los algoritmos seleccionados con los atributos coef_ o feature*importances*
2. Empezando por 2 columnas seleccionadas hasta el total de ellas se aplica RFE
3. Con las columnas seleccionadas se utiliza un grid y cross=validation para ver cuales son los mejores parámetros para el estimador

Vamos a ver como evoluciona el accuracy de los grid a medida que el número de columnas va cambiando. Además se compara con los valores que se obtuvieron en la primera iteración







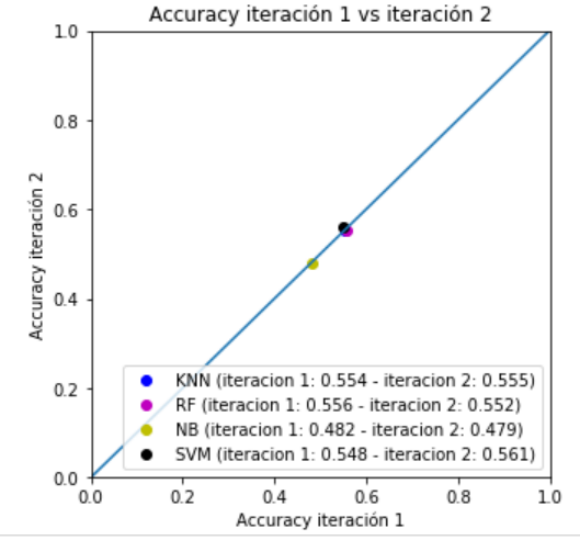
Tanto LR como LDA funcionan mejor con un número menor de atributos obteniendo una mejora en los resultado: en los dos se un 56% de aciertos con 24 columnas, frente al 53% con 110 en la iteración anterior. Se podría pensar

que la mejora podría ser debida a los cambios para calcular los atributos; pero no es asi ya que la tendencia es que a medida que se incluyen atributos, el accuracy disminuye.

El otro estimador, CART, varía más los resultados en función del número columnas seleccionadas: hay situaciones donde el accuracy es mejor, pero en cuanto se añade o elimna una columna puede ser peor que en la iteración anterior. No obstante, el mejor resultado en la segunda es mejor que en la anterior con sólo 3 columnas.

Los demás estimadores utilizarán grid como en la primera iteración para obtener los mejores parámetros para los datos de la segunda iteración:

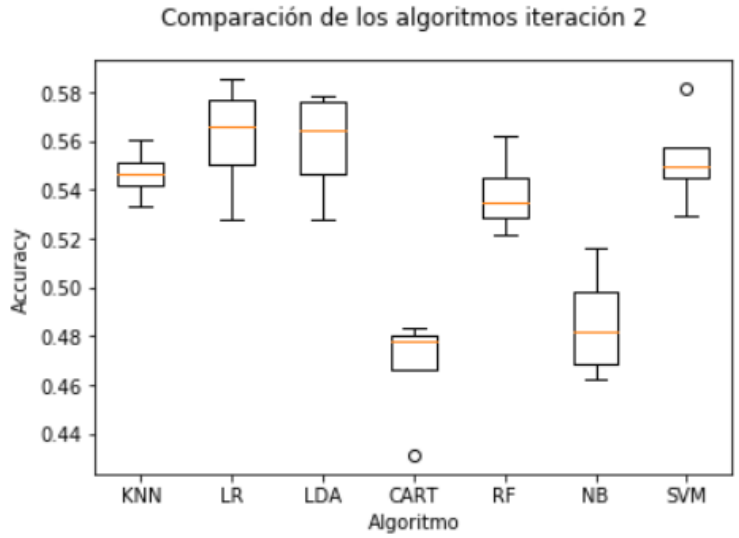
Se muestra gráficamente el accuracy de la iteración 1 y la iteración 2 para los algoritmos que no han sido calculados con RFE:

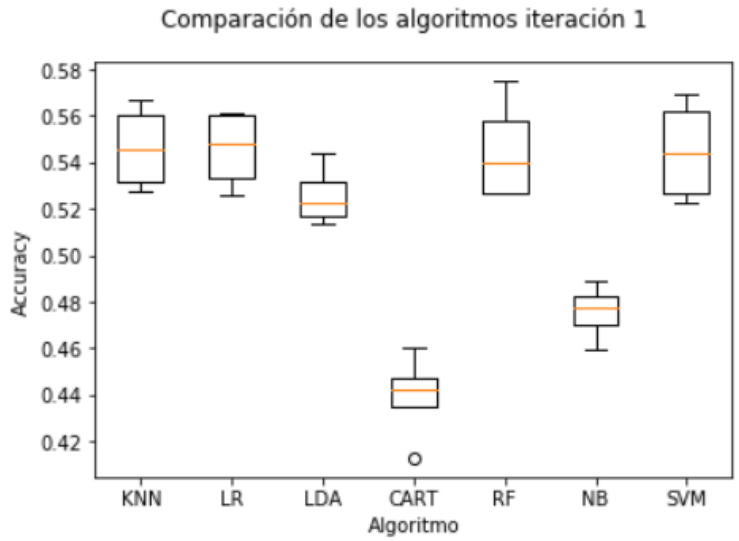


Se puede observar que en estos estimadores casi no hay diferencia entre la iteración 1 y la 2. Sólo en SVM la diferencia es mayor que un punto, mejora casi un 2%.

Como en la iteración 1, se obtiene los mejores parámetros y se entrena el modelo con cross-validation para mostrar el boxplot:

```
KNN -> media: 0.546726, desviación: 0.009761
LR -> media: 0.561392, desviación: 0.021603
LDA -> media: 0.558663, desviación: 0.020384
CART -> media: 0.467599, desviación: 0.021165
RF -> media: 0.538199, desviación: 0.015146
NB -> media: 0.485334, desviación: 0.020676
SVM -> media: 0.552524, desviación: 0.018531
```





Todos los estimadores mejoran, por lo que los pasos dados en esta segunda iteración son correctos. Los mejores son LR y LDA, que son 2 en los que se ha utilizado RFE, aunque la desviación también es mayor, por lo que el accuracy dependerá bastante de los datos del entrenamiento.

El siguiente paso: entrenar los modelos con un conjunto de entrenamiento y test siendo la relacion 70/30.

Antes de mostrar las matrices y la curva ROC, se mostrarán las columnas que se han seleccionado para los mejores RFE:

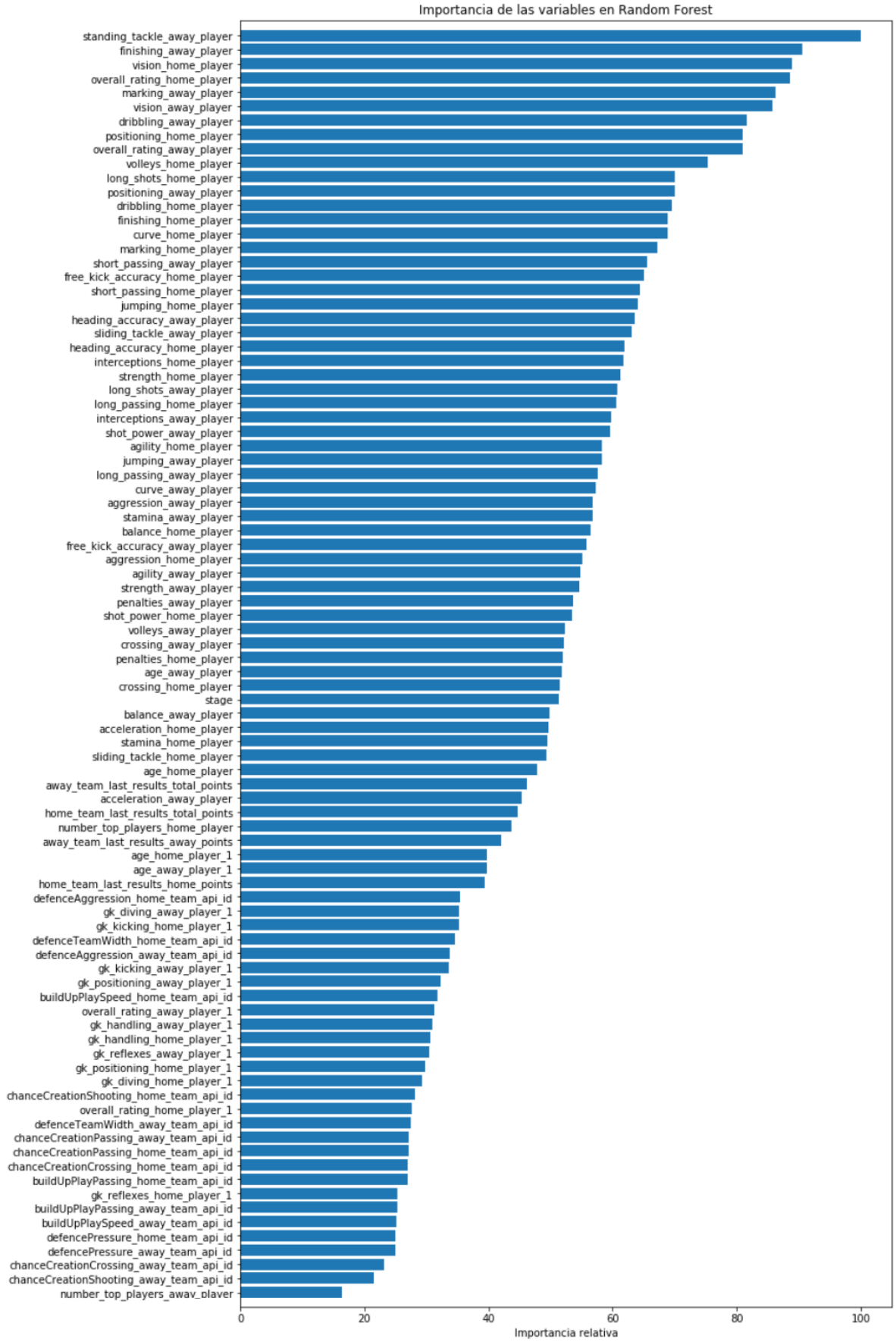
- LR: overall_rating_home_player, overall_rating_away_player, home_team_formation, number_top_players_home_player, finishing_home_player, volleys_home_player, free_kick_accuracy_home_player, positioning_home_player, vision_home_player, marking_home_player, age_home_player, number_top_players_away_player, crossing_away_player, finishing_away_player, heading_accuracy_away_player, dribbling_away_player, acceleration_away_player,

shot_power_away_player, strength_away_player, long_shots_away_player, marking_away_player, standing_tackle_away_player, chanceCreationPassing_away_team_api_id, defencePressure_away_team_api_id

- LDA: overall_rating_home_player_1, overall_rating_home_player, overall_rating_away_player, gk_positioning_home_player_1, gk_reflexes_home_player_1, number_top_players_home_player, dribbling_home_player, positioning_home_player, crossing_away_player, finishing_away_player, heading_accuracy_away_player, dribbling_away_player, long_passing_away_player, shot_power_away_player, long_shots_away_player, standing_tackle_away_player, defencePressure_away_team_api_id
- CART: overall_rating_home_player, overall_rating_away_player, strength_home_player

En todos el **overall_rating** de los jugadores no porteros es seleccionado. Resulta curioso que en LR no haya ningún atributo de porteros, pero sí hay muchos típicos de la delantera (terminación, voleas) y el número de jugadores top en ambos. En cambio en el LDA si hay algún atributo de portero (reflejos, posicionamiento). En ambos hay atributos generales de jugadores (pases largos, disparo, fuerza) y muy pocos de los equipos (presión en la defensa y creación en pases). CART utiliza sólo 3 atributos para obtener el mejor accuracy.

Respecto al RF, se muestra la importancia de las distintas columnas en el modelo:



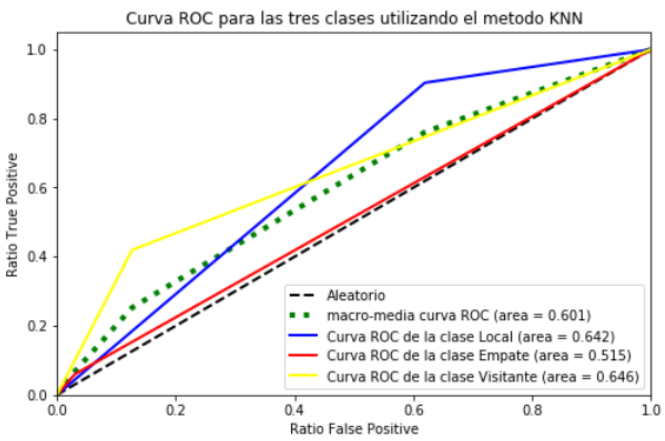
Depende de la simulación/entrenamiento pero siempre dribbling, overall y terminación están entre las primeras. Resulta curioso que esta vez la entrada agresiva esté en primera posición del equipo visitante. La teoría que se tenía

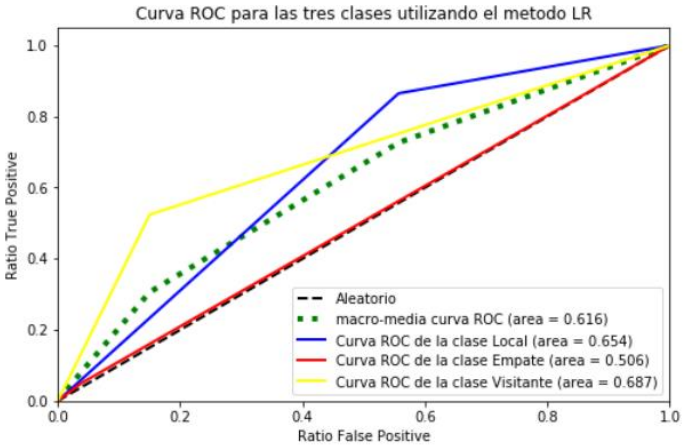
al principio sobre las rachas, parece que no es correcta (al menos en este modelo) ya que están por debajo de la mitad. Entre los atributos poco determinantes se encuentran los porteros y los relativos los equipos.

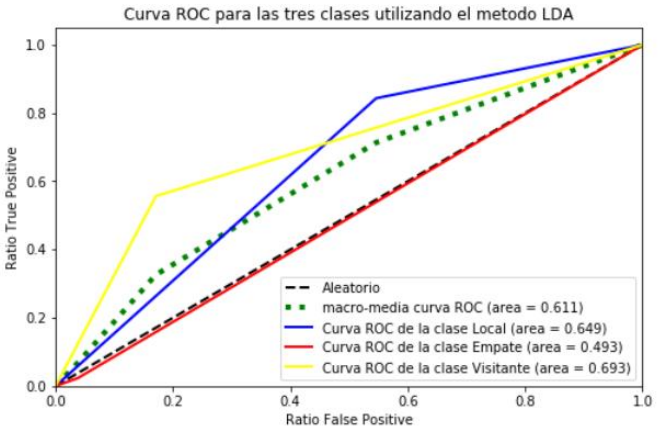
Se muestran las matrices de confusión así como el accuracy por cada modelo entrenado:

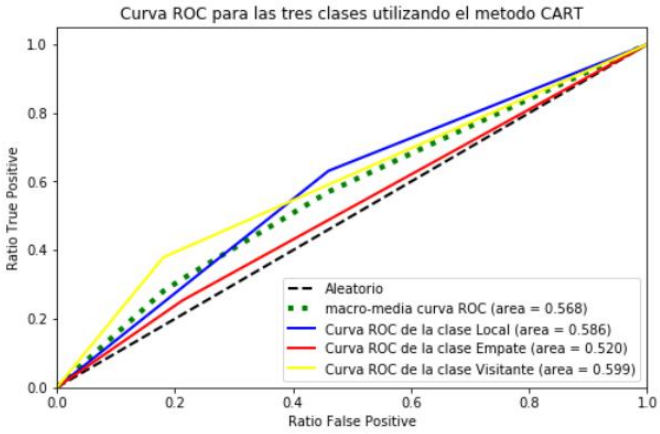
```
_____________
KNN
Test accuracy 0.559:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                375                6                  34
Verdadero Empate               158               13                  46
Verdadero Visitante            130               14                 104
_____________
LR
Test accuracy 0.568:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                359               10                  46
Verdadero Empate               157               11                  49
Verdadero Visitante            102               16                 130
_____________
LDA
Test accuracy 0.560:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                350               15                  50
Verdadero Empate               154                5                  58
Verdadero Visitante            100               10                 138
_____________
CART
Test accuracy 0.467:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                262               85                  68
Verdadero Empate               116               55                  46
Verdadero Visitante             98               56                  94
_____________
RF
Test accuracy 0.542:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                331               30                  54
Verdadero Empate               140               19                  58
Verdadero Visitante             99               22                 127
_____________
NB
Test accuracy 0.500:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                218              147                  50
Verdadero Empate                65               99                  53
Verdadero Visitante             41               84                 123
_____________
SVM
Test accuracy 0.564:
                    Predicho Local  Predicho Empate  Predicho Visitante
Verdadero Local                381                5                  29
Verdadero Empate               161                3                  53
Verdadero Visitante            128                8                 112
```

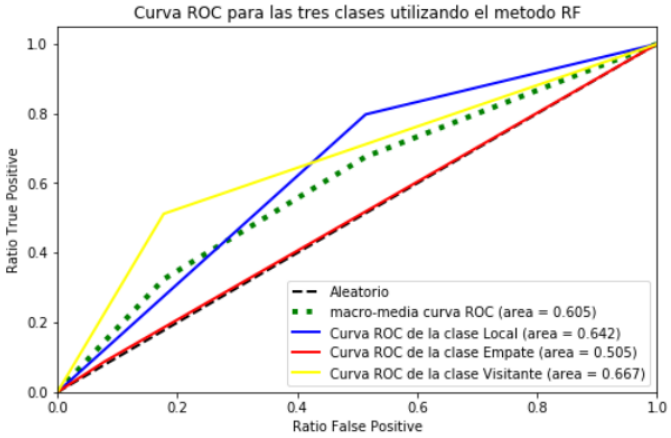
Se puede observar que cuanto mayor accuracy, mayor error de predicción en la clase empate como pasaba en la primera iteración. Esto se puede ver fácilmente en los siguientes gráficos donde se muestra la curva ROC:

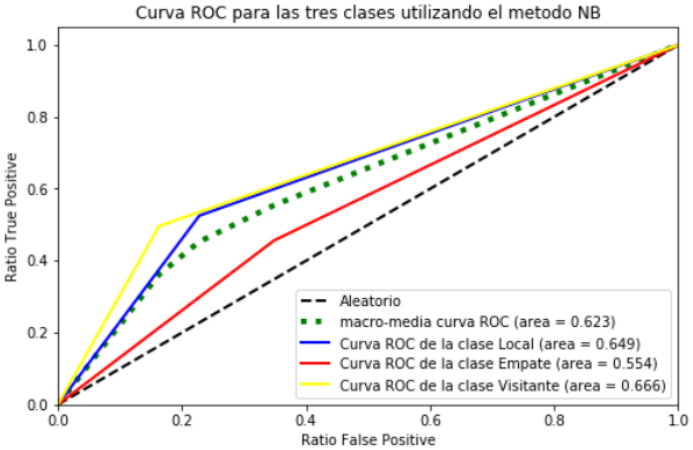


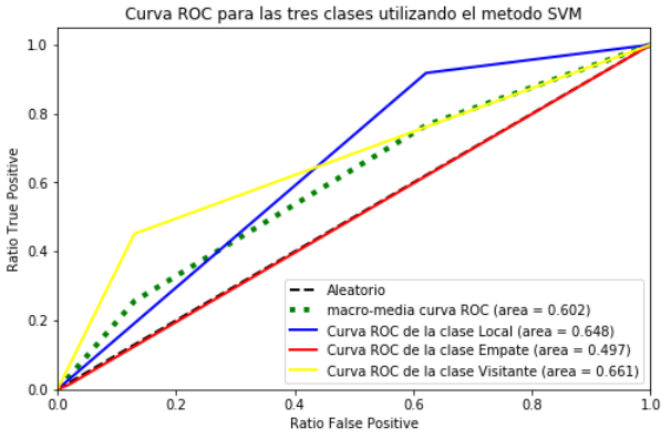












El algoritmo que tiene más área bajo la curva (0.623) es NB aunque su accuracy es del 50%. Por otro lado están LDA y LR, los cuales tienen un valor bajo la curva algo superior al 0.61 con un accuracy 6 puntos superior al anterior

(56%). Como se quiere obtener el mayor número de aciertos, se escogerá LR; además de estos dos es el que mejor predice el empate (LR es incluso peor que aleatorio).

4. Visualización

En este apartado se mostrarán los pasos que se han dado para mostrar los resultados:

- Se ha creado un servidor donde se pondrá el modelo en producción
- Y será consultado a través de una sencilla API llamada desde una sencilla página HTML

4.1. Serializar el modelo y los datos

Utilizando **pickle,** se serializa:

- El modelo entrenado elegigo de la segunda iteración
- Escalado de las variables
- Datasets:
    - Equipos: se necesitan los nombres de los equipos para el front-end. Sólo se necesitan los de La Liga
    - Jugadores: se necesitan los nombres de los jugadores para el front-end. Se exportarán todos ya que no se tiene si un jugador pertenece o no a La Liga. Se ordenarán primero si aparecen o no en algún partido de La Liga, y luego por el nombre.
    - Atributos de los equipos: se necesitan para crear el dataset de entrada. Solo los que pertenecen a La Liga.
    - Atributos de los jugadores: se necesitan para crear el dataset de entrada. Se exportaran todos.
    - Partidos en la iteración con los atributos de jugadores antes de eliminar ninguna columna: se necesitan para calcular la media de algún atributo de los porteros.
- Variables:
    - overall_rating a partir del cual un jugador es top.
    - Número de partidos que tienen distinto porcentaje a la hora de calcular los puntos.
    - Diccionarios para hacer la conversión atributos cualitativos y numéricos
    - Columnas que se pasaran al modelo y orden

El resto se pondrá en un fichero .py (**util_pfm.py**) y se importará en el servidor. De hecho, esto tendría que ser el modo en el que se tendría que hacer un proyecto: tener archivos .py, pero como se quería mostrar el código en el notebook se ha duplicado.

4.2. Servidor

Para la construcción del servidor se ha utilizado una instancia gratuita de Amazon EC2 en el que se ha instalado Flask. El servidor se puede encontrar en el fichero **server.py**.

El servidor será muy simple y sólo tendrá dos métodos:

1. get_populate_items que será accedido a través de la llamada POST populatefrontend. Se devolverá un diccionario con los datos de los equipos, jugadores, formaciones posibles para que se rellene la página con los datos.
2. get_predict que será accedido a través de la llamada POST predict para obtener la predicción del modelo para este partido y algunos datos estadísticos.
    - Los parámetros que recibe es en formato JSON:
        - date: fecha del partido
        - stage: jornada del partido
        - home_team: id del equipo local
        - away_team: id del equipo visitante
        - home_formation: id de la fomación del equipo local
        - away_formation id de la fomación del equipo local
        - home_player1: id del portero del equipo local
        - home_player2: id del jugador 2 del equipo local
        - home_player3: id del jugador 3 del equipo local
        - home_player4: id del jugador 4 del equipo local
        - home_player5: id del jugador 5 del equipo local
        - home_player6: id del jugador 6 del equipo local
        - home_player7: id del jugador 7 del equipo local
        - home_player8: id del jugador 8 del equipo local
        - home_player9: id del jugador 9 del equipo local
        - home_player10: id del jugador 10 del equipo local
        - home_player11: id del jugador 11 del equipo local
        - away_player1: id del portero del equipo visitante
        - away_player2: id del jugador 2 del equipo visitante
        - away_player3: id del jugador 3 del equipo visitante
        - away_player4: id del jugador 4 del equipo visitante
        - away_player5: id del jugador 5 del equipo visitante
        - away_player6: id del jugador 6 del equipo visitante
        - away_player7: id del jugador 7 del equipo visitante
        - away_player8: id del jugador 8 del equipo visitante
        - away_player9: id del jugador 9 del equipo visitante
        - away_player10: id del jugador 10 del equipo visitante

- away_player11: id del jugador 11 del equipo visitante
- home_last_results_0_home: referente al equipo local y al resultado en la última jornada como local. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_1_home: referente al equipo local y al resultado en la penúltima jornada como local. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_2_home: referente al equipo local y al resultado en la antepenúltima jornada como local. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_3_home: referente al equipo local y al resultado en la anteantepenúltima jornada como local. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_4_home: referente al equipo local y al resultado en la anteanteantepenúltima jornada como local. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_0_home_stage: jornada correspondiente al resultado de home_last_results_0_home o -1 si home_last_results_0_home es -1.
- home_last_results_1_home_stage jornada correspondiente al resultado de home_last_results_1_home o -1 si home_last_results_1_home es -1.
- home_last_results_2_home_stage jornada correspondiente al resultado de home_last_results_2_home o -1 si home_last_results_2_home es -1.
- home_last_results_3_home_stage jornada correspondiente al resultado de home_last_results_3_home o -1 si home_last_results_3_home es -1.
- home_last_results_4_home_stage jornada correspondiente al resultado de home_last_results_4_home o -1 si home_last_results_4_home es -1.
- home_last_results_0_total: referente al equipo local y al resultado en la última jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_1_total: referente al equipo local y al resultado en la penúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_2_total: referente al equipo local y al resultado en la antepenúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_3_total: referente al equipo local y al resultado en la anteantepenúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- home_last_results_4_total: referente al equipo local y al resultado en la anteanteantepenúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_0_away: referente al equipo visitante y al resultado en la última jornada como visitante. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_1_away: referente al equipo visitante y al resultado en la penúltima jornada como visitante. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.

- away_last_results_2_away: referente al equipo visitante y al resultado en la antepenúltima jornada como visitante. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_3_away: referente al equipo visitante y al resultado en la anteantepenúltima jornada como visitante. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_4_away: referente al equipo visitante y al resultado en la anteanteantepenúltima jornada como visitante. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_0_away_stage: jornada correspondiente al resultado de away_last_results_0_away o -1 si away_last_results_0_away es -1.
- away_last_results_1_away_stage jornada correspondiente al resultado de away_last_results_1_away o -1 si away_last_results_1_away es -1.
- away_last_results_2_away_stage jornada correspondiente al resultado de away_last_results_2_away o -1 si away_last_results_2_away es -1.
- away_last_results_3_away_stage jornada correspondiente al resultado de away_last_results_3_away o -1 si away_last_results_3_away es -1.
- away_last_results_4_away_stage jornada correspondiente al resultado de away_last_results_4_away o -1 si away_last_results_4_away es -1.
- away_last_results_0_total: referente al equipo visitante y al resultado en la última jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_1_total: referente al equipo visitante y al resultado en la penúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_2_total: referente al equipo visitante y al resultado en la antepenúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_3_total: referente al equipo visitante y al resultado en la anteantepenúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.
- away_last_results_4_total: referente al equipo visitante y al resultado en la anteanteantepenúltima jornada. -1 si no se tiene el resultado, 0 si ganó, 1 si empató, 2 si perdió.

- Se chequea que los parámetros son correctos, se trasforma añadiendo, modificando y eliminando estos datos para que se quede en el formato de entrada del estimador, y se obtiene la prediccion. Se devuelve un JSON con el resultado de la predicción y datos estadísticos obtenidos de los datos anteriores:
  - "dict_ok"
    - "input": si los datos de entrada fueron correctos (ok) o no (fail)
      - "ok"/"fail"
    - "data_problem": si los datos que se tienen para trasformar los datos de entrada en la entrada del modelo son correctos (ok) o no (fail)
      - "ok"/"fail"

- "dict_predicted"
  - "class": clase predicha por el modelo
    - clase
  - "probabilities": probabilidades entre 0 y 100 para todas las clases predichas por el modelo (se obtiene directamente a través del modelo)
    - "Local"
      - probabilidad
    - "Empate"
      - probabilidad
    - "Visitante"
      - probabilidad
- "dict_classes_simulated": se simula "Total" número de partidos (por defecto 1000) con las clases y probabilidades que da el modelo
  - "Local"
    - número
  - "Empate"
    - número
  - "Visitante"
    - número
  - "Total"
    - número
- "dict_results"
  - "teams_home_away": número de partidos históricos en el dataset que ganó el equipo local, visitante o empataron en el estadio del local
    - "Local"
      - número
    - "Empate"
      - número
    - "Visitante"
      - número
    - "Total"
      - número

- "teams_home_away_total": número de partidos históricos en el dataset que ganó el equipo local, visitante o empataron en el global
  - "Local"
    - número
  - "Empate"
    - número
  - "Visitante"
    - número
  - "Total"
    - número
- "dict_formations"
  - "formation_home": número de partidos históricos en el dataset que ganó el equipo local con esta formación
    - "Gana"
      - número
    - "Empate"
      - número
    - "Pierde"
      - número
    - "Total"
      - número
  - "formation_away": número de partidos históricos en el dataset que ganó el equipo visitante con esta formación
    - "Gana"
      - número
    - "Empate"
      - número
    - "Pierde"
      - número
    - "Total"
      - número

El programa servidor solo tiene que llamar a util_pfm.call_api con el JSON que ha recibido y otros parámetros cargados con pickle (dataframe de players, players_attributes, teams_attributes, laliga_matches, el escalador standard_scaler y las columnas de laliga_matches para éste, y el estimador y el nombre de sus columnas).

Cuando el servidor es cargado, los objetos de pickle son cargados en memoria.

Imporante: para simular el modelo necesita todos los datos:

- Equipos
- Fecha
- jornada
- Formación
- Todos los jugadores de los dos 11's
- El resultado de las jornadas anteriores si aplica. Tienen que ser lógicos, es decir, no se puede poner el resultado del penúltimo partido si no se tiene la del último

4.3. Front-end

El front-end es un simple archivo HTML que utiliza las librerias jQuery y Vue.js para manejar los datos y conectarse a la API. Si se abre con Chrome se puede tener problemas con la cabecera Access-Control-Allow-Origin (el error es el siguiente: No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore not allowed access), el cual se puede solucionar instalando el plugin "Allow-Control-Allow-Origin: *" y habilitándolo. El fichero se llama **frontend_pfm.html**

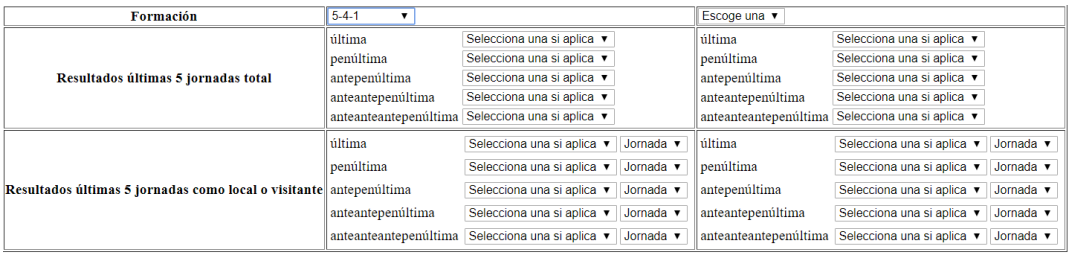
Si se quiere utilizar otra IP del servidor, éste se tiene que poner en la variable de la línea 22 del HTML.

Al cargar, se conecta al servidor y obtiene los equipos, jugadores y formaciones. Se obtendrán todos los jugadores de la base de dato ya que no se puede saber si un jugador que no ha jugado ningún partido en La Liga como titular pertence o no a esta liga.

Para obtener un resultado se tiene que cumplir:

1. Equipo local y visitante tiene que ser elegido
2. Formación del equipo local y visitante tiene que ser elegido
3. Todos los jugadores en sus posiciones
4. Fecha del encuentro
5. Jornada del encuentro
6. Resultados de los últimos enfrentamientos como equipo local y visitante, y la jornada. Si la jornada penúltima está seleccionada, entonces la última también tiene que estar seleccionado

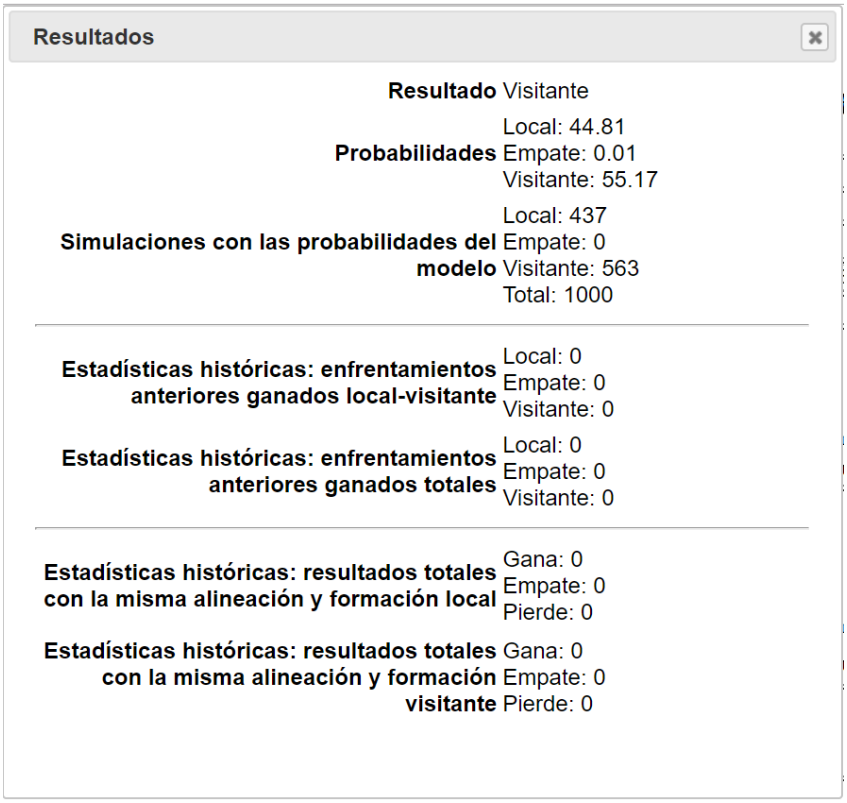
Se puede ver que las posiciones de los jugadores se modifican automáticamente en función de la fomación seleccionada:



Alineación equipo local

Una vez que hace click sobre Obtener predicción abajo de la página, un popup se desplegará arriba de la página con la siguiente información:

- Información que viene del modelo:
  - Resultado predicho por el modelo
  - Probabilidades de cada resultado dadas por el modelo
  - Resultados simulados 1000 veces con las probabilidades del modelo
- Estadísticas utilizando el histórico de datos:
  - Resultados a nivel de enfrentamientos de los equipos cuando el equipo local es local y el visitante es visitante; también se calcula el total de enfrentamientos entre sí.
  - Resultados a nivel de formación del equipo local: misma formación y mismos jugadores en las mismas posiciones.
  - Resultados a nivel de formación del equipo visitante: misma formación y mismos jugadores en las mismas posiciones.



Resultados

Resultado	Visitante
Probabilidades	Local: 44.81 Empate: 0.01 Visitante: 55.17
Simulaciones con las probabilidades del modelo	Local: 437 Empate: 0 Visitante: 563 Total: 1000
Estadísticas históricas: enfrentamientos anteriores ganados local-visitante	Local: 0 Empate: 0 Visitante: 0
Estadísticas históricas: enfrentamientos anteriores ganados totales	Local: 0 Empate: 0 Visitante: 0
Estadísticas históricas: resultados totales con la misma alineación y formación local	Gana: 0 Empate: 0 Pierde: 0
Estadísticas históricas: resultados totales con la misma alineación y formación visitante	Gana: 0 Empate: 0 Pierde: 0

5. Futuros trabajos y conclusiones

5.1. Futuros trabajos

Hay dos vías para mejorar el modelo:

1. Utilizar los datos que se tienen en la base de datos data:
    - se podría incluir la posición de cada jugador y ver su influencia en el juego
    - utilizar otros modelos o incluso una combinación de ellos y ver si da mejor resultado o no
2. Incluir datos externos tales como:
    - árbitros
    - situación meteorológica
    - si juegan o no otras competiciones y ver cómo influye

5.2. Conclusiones

El objetivo de este proyecto era predecir el resultado (1x2) de un partido de fútbol de La Liga.

Para ello se seleccionó un dataset con los partidos y resultados de las temporadas desde 2008/2009 hasta 2015/2016. Aparte se tenían datos subjetivos de los jugadores y equipos.

A partir de aquí, se ha visto cómo tratar los datos del dataset que estaba bastante limpio y que, aplicando conocmiento en el campo, se han añadido nuevos datos como la formación/sistema, puntos que ha conseguido un equipo o número de jugadores top que influirán de forma distinta que los que no lo son.

Se ha reducido el dataset y se han entrenado varios modelos:

- 1 no supervisado para ver si se podía sacar alguna características de las clases a adivinar
- varios supervisados: KNN, Logistic Regression, árboles de decisión, Random Forest, Gaussian NB y SVM
    - se ha aplicado RFE para reducir el número de atributos y se ha comprobado que con menos atributos los modelos funcionan mejor ya que no se mete tanto ruido
    - entre los atributos más importantes para los modelos son el overall_rating de los jugadores
    - se ha obtenido un acierto del 56% con 0.61 bajo la curva ROC donde el empate es muy difícil de pronosticar
    - normalmente los modelos que tienen menor accuracy, tienen mayor aciertos en empate

Se tiene que tener en cuenta que los atributos utilizados son subjetivos, por lo que también podría interoducir ruido.

Por último se ha creado:

1. una API en un servidor que obtiene el valor del resultado dada cierta información como los equipos, las alineaciones, la formación, la fecha del partido, la jornada y el resultado en los últimos partidos.
2. una página HTML la cual es utilizada para rellenar los datos que necesita la API y despliega los resultados.

Firmado digitalmente por: RODRIGUEZ MORON, ENRIQUE (FIRMA)