



Proyecto Final de Máster

Sistema de análisis y predicción de rendimiento en jugadas a balón parado de carácter ofensivo para jugadores de fútbol existentes y nuevos utilizando aprendizaje automático

Luis Alejandro Ruiz

Madrid, 2023

1. Objetivos del proyecto de fin de máster

Objetivo general:

Desarrollar un modelo de aprendizaje automático para analizar y predecir el desempeño en acciones a balón parado de carácter ofensivo de jugadores de la primera división española y utilizar este modelo para evaluar a jugadores emergentes, nuevos o canteranos.

Objetivos específicos:

1. Revisar la literatura y los estudios existentes sobre la aplicación de modelos de aprendizaje automático en el ámbito del fútbol y la predicción del desempeño de los jugadores en acciones a balón parado.
2. Recolectar y consolidar datos históricos y estadísticas de jugadores de la primera división española relacionadas con acciones a balón parado de carácter ofensivo, como tiros libres, córners y penaltis.
3. Preprocesar y limpiar los datos recopilados para asegurar su calidad y compatibilidad con los algoritmos de aprendizaje automático, lo que incluye la eliminación de valores faltantes, la estandarización de variables y la codificación de variables categóricas.
4. Realizar un análisis exploratorio de los datos para identificar patrones, correlaciones y características relevantes que puedan influir en el desempeño de los jugadores en acciones a balón parado.
5. Seleccionar y entrenar diferentes modelos de aprendizaje automático, como regresión lineal, árboles de decisión, máquinas de soporte vectorial y redes neuronales, para predecir el desempeño en acciones a balón parado de carácter ofensivo de los jugadores.
6. Evaluar y comparar el rendimiento de los modelos entrenados utilizando métricas de evaluación adecuadas, como el error cuadrático medio, el coeficiente de determinación y la precisión, con el fin de seleccionar el modelo que proporcione las mejores predicciones.
7. Implementar el modelo seleccionado en una herramienta o aplicación que permita a los clubes de fútbol y otros interesados acceder a las predicciones de desempeño en acciones a balón parado de carácter ofensivo de los jugadores y utilizar esta información para evaluar a jugadores emergentes, nuevos o canteranos.

2. Propuesta de Solución

Lista de acciones y tareas en base a los objetivos específicos planteados

2.1. Revisar las características y creación del modelo que permiten medir el desempeño de los jugadores en acciones a balón parado.

Para realizar esta actividad se revisó la literatura del módulo 8 herramientas de video análisis, con lo cual se definió un listado inicial de categoría de metricas asociadas a este proyecto. En efecto, estas características se centran en acciones a balón parado ofensivas y describe aspectos clave para tener en cuenta al analizar y evaluar el desempeño de jugadores en diferentes situaciones de juego.

Basado en la literatura proporcionada, se pueden identificar las siguientes categorías de características relacionadas con el fútbol que apoyan al modelo y el sistema:

- ❖ Córner ofensivo, faltas escoradas y laterales: Se analizan los lanzadores principales, el tipo de golpeo, la dirección, el posicionamiento de los jugadores en zona de remate y rechace, y la efectividad anotadora, entre otros aspectos.
- ❖ Faltas directas: Se consideran los lanzadores principales, el tipo de golpeo, la dirección, la efectividad de los lanzadores y el posicionamiento en el área rival.
- ❖ Faltas lejanas: Se analizan los lanzadores principales, el tipo de golpeo, la dirección, el posicionamiento de los jugadores en zona de remate y los jugadores retrasados.
- ❖ Saques de banda: Se evalúa cómo se ejecutan los lanzamientos, la efectividad dependiendo de la zona y el posicionamiento de los jugadores.
- ❖ Penaltis: Se analizan los principales lanzadores, cómo ejecutan los lanzamientos y la efectividad en diferentes contextos y niveles de exigencia.

Estos aspectos se pueden utilizar para informar la creación del modelo y del sistema. A continuación, se explica cómo se han utilizado estos detalles para crear el modelo y el sistema:

- Creación del modelo de jugador: Basándose en las métricas descritas en el texto, se puede crear un modelo de jugador que incluya características relevantes para evaluar el desempeño en acciones a balón parado ofensivas. Por ejemplo, se pueden incluir atributos como lanzadores principales, tipo de golpeo, efectividad anotadora, entre otros.
- Extracción de características: A partir de los datos disponibles, se pueden calcular las métricas mencionadas en el texto para cada jugador. Esto puede implicar el análisis de datos históricos y la combinación de diferentes fuentes de datos para obtener una visión completa del desempeño de un jugador en acciones a balón parado ofensivas.
- Modelo de predicción: Utilizando las características extraídas, se puede entrenar un modelo de aprendizaje automático para predecir el desempeño futuro en acciones a balón parado ofensivas. Este modelo puede basarse en algoritmos de regresión, clasificación o agrupamiento, según el objetivo específico del análisis.
- Implementación del Backend: El Backend, en este caso, implementado usando FastAPI y MongoDB, se encarga de gestionar el almacenamiento y acceso a los datos de los jugadores y sus métricas. El texto proporciona una guía para los atributos y métricas que deben almacenarse y consultarse en la base de datos.

- Interfaz de usuario y visualización de datos: La capa de presentación, creada usando Angular, permite visualizar y analizar las métricas descritas en el texto de una manera intuitiva y fácil de entender. Esto incluye tablas, gráficos y otros elementos visuales que ayudan a los usuarios a comprender y evaluar el desempeño de los jugadores en acciones a balón parado ofensivas.

En resumen, los detalles proporcionados en el texto del módulo 8 se han utilizado para informar la creación del modelo de jugador, la extracción de características, el entrenamiento del modelo de predicción, la implementación del Backend y la creación de la capa de presentación.

Diagrama que muestra la arquitectura del sistema propuesto:

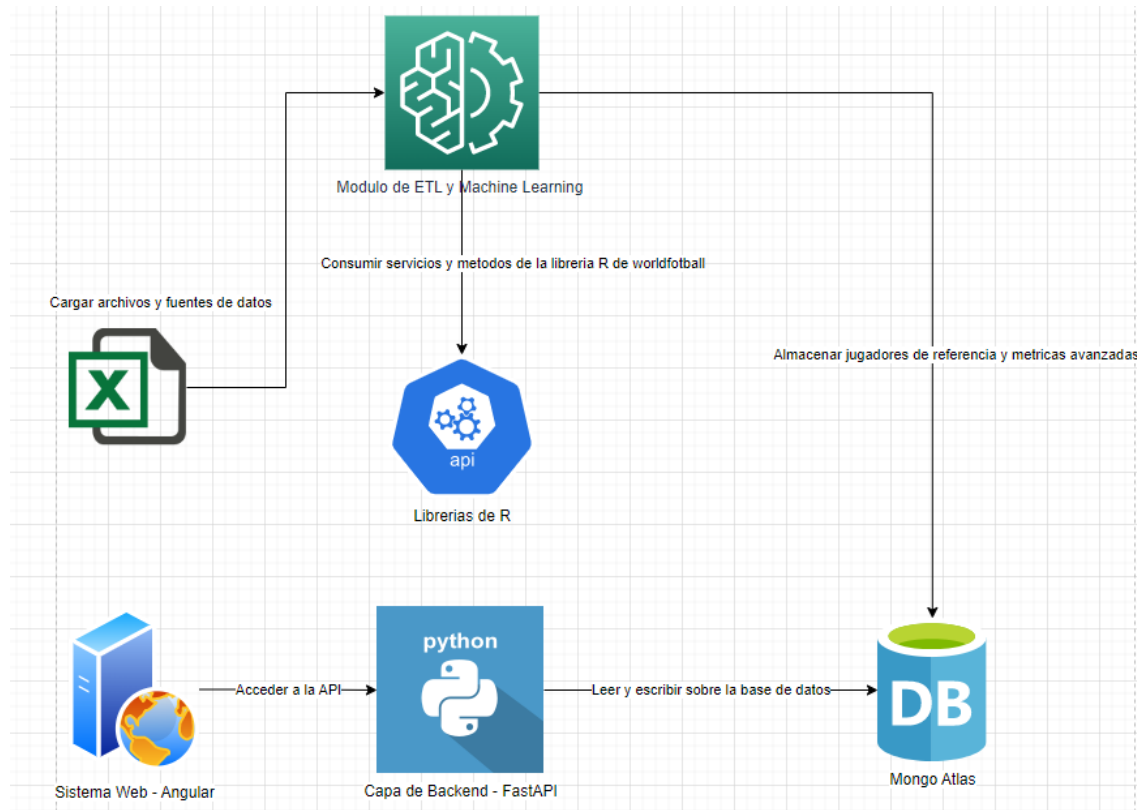


Imagen 1. Diagrama de arquitectura propuesta para el proyecto final

2.2.Recolectar y consolidar datos históricos y estadísticas de jugadores de la primera división española relacionadas con acciones a balón parado de carácter ofensivo, como tiros libres, córners y penaltis.

La carga de datos es el primer paso en el proceso, y su objetivo es obtener los datos necesarios para entrenar y evaluar los modelos de aprendizaje automático. En este proyecto, se utilizan datos de jugadores de fútbol provenientes de diferentes fuentes, como FootyStats (de bajo costo) y FBref (gratuito). Estos datos se encuentran en archivos CSV (valores separados por comas), que son archivos de texto que contienen datos tabulares con campos separados por comas.

Para cargar estos archivos CSV, se utiliza la biblioteca Pandas de Python, la cual es una herramienta muy popular para el manejo y análisis de datos en estructuras tabulares. Pandas permite leer archivos CSV y convertirlos en DataFrames, que son estructuras de datos bidimensionales similares a una tabla de una base de datos o una hoja de cálculo de Excel.

El proceso de carga de datos involucra los siguientes pasos:

Importar la biblioteca Pandas:

Para utilizar Pandas, primero es necesario importarla en el código de Python con la siguiente línea:

```
import numpy as np
import pandas as pd
import rpy2.robjects as robjects
from fuzzywuzzy import fuzz
from rpy2.robjects import pandas2ri
from rpy2.robjects.conversion import localconverter
from rpy2.robjects.packages import importr
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.tree import DecisionTreeRegressor
from mongodb_operations import setup_mongodb_connection, store_top_players_in_mongodb, store_feature_weights_in_mongodb
from tqdm import tqdm
```

Imagen 2. Generando las importaciones de las librerías

Generar archivos CSV a partir de librerías de R y descarga de archivos de FootyStats:

El método, `extract_stats_and_save_to_csv(stat_type, csv_file_name)`, es una función en Python que utiliza la biblioteca `rpy2` para ejecutar código en R y extraer estadísticas de jugadores de la primera división española de fútbol para la temporada actual, utilizando el paquete `R worldfootballR`. Luego, guarda las estadísticas en un archivo CSV para su posterior análisis y procesamiento.

```
def extract_stats_and_save_to_csv(stat_type, csv_file_name):
    # Activate the automatic conversion between pandas and R data.frames
    pandas2ri.activate()

    # Import worldfootballR package
    worldfootballR = importr("worldfootballR")

    # Define the R function to extract stats for La Liga's current season
    r_script = f'''
library(worldfootballR)

# Get {stat_type} stats for La Liga's current season
stats <- fb_league_stats(country = "ESP", tier="1st", season_end_year = "2023", gender="M",
, stat_type = "{stat_type}", team_or_player = "player")

return(stats)
'''
```

Imagen 3. Generando la creación de archivos CSV de FBref usando para ello la librería de R, worldfootballR

En el caso de FootyStats se realizó la descarga del archivo de jugadores de la liga española directamente desde la plataforma.

Spain - Current Season					
League					
Supercopa de Espana	⚽ League CSV	📅 Match Stats CSV	🏠 Teams CSV	📊 Team Pt.2	👤 Players CSV
Copa del Rey	⚽ League CSV	📅 Match Stats CSV	🏠 Teams CSV	📊 Team Pt.2	👤 Players CSV
La Liga	⚽ League CSV	📅 Match Stats CSV	🏠 Teams CSV	📊 Team Pt.2	👤 Players CSV

Imagen 4. Descarga del archivo de estadísticas de jugadores directamente desde FootyStats

Leer archivos CSV:

Se utilizan funciones como `pd.read_csv()` para leer cada archivo CSV y convertirlo en un DataFrame. Por ejemplo:

```
1 usage
def load_data():
    # Load the data
    fstats = pd.read_csv('historical/spain-la-liga-players-2022-to-2023-stats.csv')
    fbref_pt = pd.read_csv('historical/passing_types.csv')
    fbref_st = pd.read_csv('historical/shooting_types.csv')
    return fstats, fbref_pt, fbref_st
```

Imagen 5. Generando la carga de los archivos CSV de las fuentes de FootyStats y la FBref

Verificar los datos: Después de cargar los archivos CSV en DataFrames, se pueden explorar y verificar los datos utilizando funciones como `head()`, `tail()`, `info()` y `describe()`. Estas funciones permiten visualizar las primeras y últimas filas del DataFrame, obtener información sobre el tipo de dato y el número de valores no nulos de cada columna, y obtener estadísticas descriptivas de las columnas numéricas, respectivamente.

Una vez que los datos han sido cargados correctamente en los DataFrames, se procede a la siguiente etapa del proceso, que es la limpieza y el preprocesamiento de datos. Esto incluye unificar los diferentes DataFrames en uno solo, gestionar los valores faltantes y convertir las características categóricas en valores numéricos, entre otros pasos.

2.3. Preprocesar y limpiar los datos recopilados para asegurar su calidad y compatibilidad con los algoritmos de aprendizaje automático, lo que incluye la eliminación de valores faltantes, la estandarización de variables y la codificación de variables categóricas.

En este proceso se define dos tareas sobre el código realizado, `clean_and_preprocess_data` y `feature_engineering`, que realizan la limpieza, el preprocesamiento y la ingeniería de características de los datos recopilados de Footy Stats y FBref.

`clean_and_preprocess_data`: Esta función realiza las siguientes operaciones:

- Fusiona los DataFrames de FBref (`fbref_pt` y `fbref_st`) basándose en varias columnas comunes, como 'Player', 'Squad', 'Nation', 'Age', 'Born' y 'Pos'.
- Empareja los nombres y clubes de los jugadores en el DataFrame de Footy Stats (`fstats`) con los nombres y clubes en el DataFrame combinado de FBref utilizando la función `match_name`.
- Fusiona el DataFrame de Footy Stats con el DataFrame de FBref en función de los nombres y clubes emparejados.
- Rellena los valores faltantes (NaN) con ceros.
- Codifica la columna 'full_name' utilizando la codificación one-hot y concatena las columnas codificadas al DataFrame.
- Selecciona las columnas relevantes para el modelo de análisis ofensivo de acciones a balón parado y devuelve el DataFrame limpio y preprocesado.

```
def clean_and_preprocess_data(fstats, fbref_pt, fbref_st):
    # Merge the dataframes on the 'Player' column
    merged_df = fbref_pt.merge(fbref_st, on=['Player', 'Squad', 'Nation', 'Age', 'Born', 'Pos'],
                              suffixes=('_passing', '_shooting'))

    matched_names = []
    matched_clubs = []
    for index, row in fstats.iterrows():
        match = match_name(row['full_name'], row['Current Club'], merged_df['Player'].tolist(),
                           merged_df['Squad'].tolist(), 75)
        matched_names.append(match[0])
        matched_clubs.append(match[1])

    fstats['matched_name'] = matched_names
    fstats['matched_club'] = matched_clubs

    merged_data = fstats.merge(fbref_pt, left_on=['matched_name', 'matched_club'], right_on=['Player', 'Squad'],
                              how='left')
```

Imagen 6. Utilizando el método de limpieza y preprocesamiento de la data

feature_engineering: Esta función realiza la ingeniería de características en el DataFrame limpio y preprocesado de la siguiente manera:

- Calcula nuevas características a partir de las métricas existentes en el DataFrame, como tasas de éxito de pases y tiros libres, porcentajes de conversión de penales, entre otras.
- Normaliza las características numéricas utilizando MinMaxScaler, que escala las características en un rango de 0 a 1.
- Reemplaza los valores infinitos con NaN y rellena los valores NaN con la media (o mediana) de la columna correspondiente.
- Devuelve el DataFrame con las características procesadas y escaladas.

```
def feature_engineering(clean_data):  
    # FOOTY STATS #  
    clean_data['cross_completion_rate_percentile_overall'] = safe_divide(clean_data['crosses_total_overall'],  
                                                                           clean_data['crosses_per_game_overall'])  
    clean_data['crosses_per_90_overall'] = clean_data['crosses_total_overall'] / clean_data['passes_total_overall'] * 90  
    clean_data['through_passes_per_90_overall'] = clean_data['through_passes_total_overall'] / clean_data['  
        'passes_total_overall'] * 90  
    clean_data['long_passes_per_90_overall'] = clean_data['long_passes_total_overall'] / clean_data['  
        'passes_total_overall'] * 90  
    clean_data['shot_accuracy_percentage_overall'] = clean_data['shots_faced_per90_percentile_overall'] / clean_data['  
        'key_passes_total_overall']  
    clean_data['dribbles_successful_percentage_percentile_overall'] = safe_divide(  
        clean_data['dribbles_successful_total_overall'], clean_data['dribbles_total_overall'])  
  
    clean_data['crosses_pg_percentile_overall'] = clean_data['crosses_per_game_overall'] / clean_data['  
        'appearances_overall']  
    clean_data['key_passes_pg_percentile_overall'] = clean_data['key_passes_per_game_overall'] / clean_data['  
        'appearances_overall']  
    clean_data['through_passes_pg_percentile_overall'] = clean_data['through_passes_per_game_overall'] / clean_data['  
        'appearances_overall']  
    clean_data['long_passes_pg_percentile_overall'] = clean_data['long_passes_per_game_overall'] / clean_data['  
        'appearances_overall']
```

Imagen 7. Utilizando método que construye las metricas avanzadas a partir de las metricas iniciales recopiladas

En principio se seleccionaron 25 métricas avanzadas que logren explicar la influencia de crear y realizar jugadas de balón parado ofensivo, las cuales son los siguientes:

1. cross_completion_rate_percentile_overall: Tasa de finalización de centros, que indica la efectividad de un jugador al realizar centros. Se calcula dividiendo el total de centros por el promedio de centros por partido.
2. crosses_per_90_overall: Cantidad promedio de centros que un jugador realiza por cada 90 minutos de juego. Se calcula dividiendo el total de centros por el total de pases y multiplicando por 90.
3. through_passes_per_90_overall: Cantidad promedio de pases filtrados que un jugador realiza por cada 90 minutos de juego. Se calcula dividiendo el total de pases filtrados por el total de pases y multiplicando por 90.
4. long_passes_per_90_overall: Cantidad promedio de pases largos que un jugador realiza por cada 90 minutos de juego. Se calcula dividiendo el total de pases largos por el total de pases y multiplicando por 90.
5. shot_accuracy_percentage_overall: Porcentaje de precisión de disparos, que indica la efectividad de un jugador al realizar disparos. Se calcula dividiendo los tiros enfrentados por 90 en percentiles por el total de pases clave.

6. `dribbles_successful_percentage_percentile_overall`: Porcentaje de éxito en los regates, que indica la efectividad de un jugador al realizar regates. Se calcula dividiendo el total de regates exitosos por el total de regates.
7. `crosses_pg_percentile_overall`: Percentil de centros por partido, que indica la frecuencia con la que un jugador realiza centros. Se calcula dividiendo los centros por partido por el total de apariciones.
8. `key_passes_pg_percentile_overall`: Percentil de pases clave por partido, que indica la frecuencia con la que un jugador realiza pases clave. Se calcula dividiendo los pases clave por partido por el total de apariciones.
9. `through_passes_pg_percentile_overall`: Percentil de pases filtrados por partido, que indica la frecuencia con la que un jugador realiza pases filtrados. Se calcula dividiendo los pases filtrados por partido por el total de apariciones.
10. `long_passes_pg_percentile_overall`: Percentil de pases largos por partido, que indica la frecuencia con la que un jugador realiza pases largos. Se calcula dividiendo los pases largos por partido por el total de apariciones.
11. `total_corners`: Total de saques de esquina (in-swinging, out-swinging y straight).
12. `in-swinging_corner_percentage`: Porcentaje de saques de esquina in-swinging en relación con el total de saques de esquina.
13. `out-swinging_corner_percentage`: Porcentaje de saques de esquina out-swinging en relación con el total de saques de esquina.
14. `straight_corner_percentage`: Porcentaje de saques de esquina straight en relación con el total de saques de esquina.
15. `successful_corners`: Cantidad de saques de esquina exitosos
16. `corner_success_rate`: Tasa de éxito en saques de esquina, que indica la efectividad de un jugador en saques de esquina. Se calcula dividiendo los saques de esquina exitosos por el total de saques de esquina.
17. `in-swinging_corner_success_rate`: Tasa de éxito en saques de esquina in-swinging, que indica la efectividad de un jugador en este tipo de saques de esquina. Se calcula dividiendo los saques de esquina exitosos por los saques de esquina in-swinging.
18. `out-swinging_corner_success_rate`: Tasa de éxito en saques de esquina out-swinging, que indica la efectividad de un jugador en este tipo de saques de esquina. Se calcula dividiendo los saques de esquina exitosos por los saques de esquina out-swinging.
19. `straight_corner_success_rate`: Tasa de éxito en saques de esquina straight, que indica la efectividad de un jugador en este tipo de saques de esquina. Se calcula dividiendo los saques de esquina exitosos por los saques de esquina straight.
20. `pass_completion_rate`: Tasa de finalización de pases, que indica la efectividad de un jugador al completar pases. Se calcula dividiendo los pases completados (`Cmp_Outcomes`) por los pases intentados (`Att`).
21. `dead_ball_pass_rate`: Tasa de pases con el balón parado, que indica la proporción de pases realizados con el balón parado. Se calcula dividiendo los pases de balón parado (`Dead_Pass Types`) por los pases intentados (`Att`).
22. `free_kicks_rate_from_dead_ball`: Tasa de tiros libres realizados desde un balón parado. Se calcula dividiendo los pases de tiros libres (`FK_Pass Types`) por los pases de balón parado (`Dead_Pass Types`).
23. `cross_completion_rate_from_dead_ball`: Tasa de finalización de centros desde un balón parado. Se calcula dividiendo los pases de centros (`Crs_Pass Types`) por los pases de balón parado (`Dead_Pass Types`).

24. `through_ball_completion_rate_from_dead_ball`: Tasa de finalización de pases filtrados desde un balón parado. Se calcula dividiendo los pases de pases filtrados (TB_Pass Types) por los pases de balón parado (Dead_Pass Types).
25. `free_kicks_conversion_rate`: Tasa de conversión de tiros libres, que indica la efectividad de un jugador al marcar goles desde tiros libres. Se calcula dividiendo los goles totales (`goals_overall`) por los pases de tiros libres (FK_Pass Types).

2.4. Realizar un análisis exploratorio de los datos (EDA) para identificar patrones, correlaciones y características relevantes que puedan influir en el desempeño de los jugadores en acciones a balón parado.

Se realizó durante este proceso de EDA la eliminación de algunas variables y por ende se descartaron varias métricas avanzadas por falta de datos por parte de las fuentes de orígenes para así obtener un modelo más preciso y acorde.

Al ejecutar el siguiente método se logró esta tarea:

```
clean_data = scored_data.drop(columns=["Set_Piece_Performance", "full_name"])

# Estadísticas descriptivas
summary_data = clean_data.describe()
summary_data.to_excel("resumen_estadistico.xlsx")

# Correlación entre características numéricas
correlation_matrix = clean_data.corr()
correlation_matrix.to_excel("resumen-correlacion.xlsx")
```

Imagen 8. Método que permite obtener estadísticas descriptivas y de correlación entre las variables

Se descartaron las siguientes variables luego de realizar este análisis exploratorio por falta de datos:

- `through_passes_per_game_overall`
- `long_passes_per_game_overall`
- `through_passes_total_overall`
- `through_passes_per_90_overall`
- `long_passes_total_overall`
- `chances_created_total_overall`
- `chances_created_per_90_overall`
- `chances_created_per90_percentile_overall`
- `shots_per_goal_conceded_overall`
- `pen_save_percentage_overall`
- `distance_travelled_total_overall`
- `distance_travelled_per_90_overall`
- `man_of_the_match_total_overall`
- `long_passes_per_90_overall`
- `through_passes_pg_percentile_overall`
- `long_passes_pg_percentile_overall`

Realizando un análisis a través de la correlación de las variables, podemos descartar mas variables y de esta forma establecer mejores criterios para realizar el scoring inicial de nuestro modelo.

metricas	Kicks	Str_Corner Kicks	Cmp_Outcomes	Off_Outcomes	Blocks_Outcomes
cross_completion_rate_percentile_overall	9266104	0,039266095	0,039267354	0,039266103	0,039266125
crosses_pg_percentile_overall	8819482	0,018819478	0,018819303	0,018819478	0,01881948
key_passes_pg_percentile_overall	7904496	0,017904494	0,017904361	0,017904494	0,017904495
total_corners	1	1	1	1	1
in-swinging_corner_percentage	1432644	-0,061432659	-0,061431428	-0,06143265	-0,061432626
out-swinging_corner_percentage	3912258	-0,06391228	-0,063911145	-0,063912271	-0,063912249
straight_corner_percentage	6860579	-0,916860582	-0,916860352	-0,916860581	-0,916860577
successful_corners	1	1	1	1	1
corner_success_rate	8771094	-0,018771114	-0,018769725	-0,018771104	-0,018771076
in-swinging_corner_success_rate	3229016	-0,032290183	-0,032288852	-0,032290174	-0,032290149
out-swinging_corner_success_rate	0214655	-0,030214672	-0,030213516	-0,030214664	-0,030214641
straight_corner_success_rate	0869635	-0,008696366	-0,008695479	-0,008696361	-0,008696354
pass_completion_rate	6276512	-0,076276522	-0,076274023	-0,076276509	-0,076276477
dead_ball_pass_rate	6049427	-0,476049435	-0,476048163	-0,476049428	-0,47604941
free_kicks_rate_from_dead_ball	8837539	-0,138837546	-0,138835225	-0,138837536	-0,138837519
cross_completion_rate_from_dead_ball	2669222	0,012669214	0,012670066	0,012669221	0,012669248
through_ball_completion_rate_from_dead_ball	9756399	-0,389756401	-0,389755632	-0,389756395	-0,389756379
free_kicks_conversion_rate	5210265	0,005210264	0,00521064	0,005210269	0,005210282
penalty_conversion_rate	7646551	0,007646548	0,007646641	0,007646548	0,00764655

Imagen 9. Análisis de correlación de variables

Se ha decidido descartar de las métricas avanzadas: straight_corner_percentage y se ha decidido asignarles mayor peso a las siguientes métricas: cross_completion_rate_percentile_overall (muy buena correlación con las variables del modelo) y las otras métricas: total_corners, successful_corners, corner_success_rate, out-swinging_corner_success_rate, free_kicks_rate_from_dead_ball.

2.5. Seleccionar y entrenar diferentes modelos de aprendizaje automático, como regresión lineal, árboles de decisión y Random Forest para predecir el desempeño en acciones a balón parado de carácter ofensivo de los jugadores.

Para empezar con esta tarea se decidió utilizar el análisis exploratorio anterior para poder establecer los pesos de las métricas, con lo cual se creó el promedio del puntaje inicial, lo cual es un scoring inicial que representa la sumatoria de las variables filtradas y seleccionadas con su respectivo peso.

Los pesos fueron divididos de la siguiente forma:

Variable	Peso
cross_completion_rate_percentile_overall	0,11
crosses_per_90_overall	0,04
dribbles_successful_percentage_percentile_overall	0,04
crosses_pg_percentile_overall	0,04
key_passes_pg_percentile_overall	0,04
total_corners	0,05
in-swinging_corner_percentage	0,03
out-swinging_corner_percentage	0,04
successful_corners	0,07
corner_success_rate	0,07
in-swinging_corner_success_rate	0,04
out-swinging_corner_success_rate	0,08
straight_corner_success_rate	0,04
pass_completion_rate	0,09
dead_ball_pass_rate	0,03
free_kicks_rate_from_dead_ball	0,08
cross_completion_rate_from_dead_ball	0,04
through_ball_completion_rate_from_dead_ball	0,03
free_kicks_conversion_rate	0,04

Modelos de entrenamiento utilizados:

- **Regresión Lineal:** Este modelo es simple y fácil de interpretar, basado en una relación lineal entre las variables independientes y la variable dependiente. A pesar de su simplicidad, puede ser útil para establecer una línea de base en el rendimiento del modelo y entender cómo cada característica influye en el rendimiento en acciones a balón parado. Sin embargo, su principal desventaja es que no puede capturar relaciones no lineales ni interacciones entre características.
- **Árbol de Decisión:** Este modelo divide los datos en subconjuntos basados en criterios específicos y realiza predicciones basadas en las proporciones de la variable objetivo en cada subconjunto. Los árboles de decisión son fáciles de interpretar y pueden capturar relaciones no lineales. No obstante, son propensos al sobreajuste, especialmente cuando se trata de conjuntos de datos con muchas características. En este caso, se utiliza GridSearchCV para optimizar los parámetros del árbol de decisión y reducir el riesgo de sobreajuste.
- **Random Forest:** Este modelo es un método de conjunto que construye múltiples árboles de decisión y combina sus predicciones. El Random Forest es ideal para este proyecto porque puede manejar una gran cantidad de características, capturar interacciones complejas y tiene una buena resistencia al sobreajuste. Además, este modelo puede proporcionar importancia de características, lo que puede ser útil para identificar qué variables tienen el mayor impacto en el rendimiento de acciones a balón parado. Aquí también se utiliza GridSearchCV para optimizar los parámetros del bosque aleatorio.

En efecto, se entrenan estos modelos de aprendizaje automático utilizando la función `train_models()`. La función toma como entrada los datos de características sin las columnas "Set_Piece_Performance" y "full_name" y la columna objetivo "Set_Piece_Performance". Además, se utiliza el parámetro `use_random_forest` para determinar si se debe entrenar un modelo de Random Forest o no. La función devuelve una lista de modelos entrenados llamada `trained_models`.

```
def train_models(x_train, y_train, using_random_forest):
    # Train and optimize the models

    # Linear Regression
    tqdm.write("Training Linear Regression...")
    lr_model = LinearRegression()
    lr_model.fit(x_train, y_train)

    # Decision Tree
    tqdm.write("Training Decision Tree...")
    dt_model = DecisionTreeRegressor(random_state=42)
    dt_params = {'max_depth': range(1, 11), 'min_samples_split': range(2, 11)}
    dt_grid = GridSearchCV(dt_model, dt_params, cv=5, scoring='neg_mean_squared_error', n_jobs=-1)
    dt_grid.fit(x_train, y_train)
    best_dt_model = dt_grid.best_estimator_
```

Imagen 10. Utilizando diferentes modelos de entrenamiento, tales como Regresión Lineal, Árboles de Decisiones y Bosque Aleatorio

Luego como paso final de este proceso, se ejecuta el siguiente método: **identify_top_players** es identificar a los 10 mejores jugadores en función de su rendimiento en acciones a balón parado ofensivas utilizando un conjunto de modelos entrenados previamente.

```

def identify_top_players(list_trained_models, data_players, lst_player_names):
    predictions = []

    for model_name, model in list_trained_models.items():
        model_predictions = model.predict(data_players)
        predictions.append(pd.Series(model_predictions, name=model_name))

    predictions_df = pd.concat(predictions, axis=1)
    ensemble_predictions = predictions_df.mean(axis=1)
    ensemble_predictions.index = data_players.index
    X_with_predictions = data_players.assign(Set_Piece_Performance=ensemble_predictions)
    X_with_predictions['full_name'] = lst_player_names

    lst_top_players = X_with_predictions.nlargest(10, 'Set_Piece_Performance')

    # Add feature values to the top_players DataFrame
    top_players_features = lst_top_players[feature_columns]
    lst_top_players = lst_top_players[['full_name', 'Set_Piece_Performance']]
    top_players_with_features = pd.concat([lst_top_players, top_players_features], axis=1)

    return top_players_with_features

```

Imagen 11. Selección de los jugadores de referencia (Top 10) para guardarse en un DataFrame y luego en base de datos

Para cada modelo en `list_trained_models`, se realiza una predicción utilizando los datos de los jugadores (`data_players`) como entrada. Las predicciones de cada modelo se almacenan en una serie de Pandas y se agregan a la lista `predictions`. Luego se calcula el promedio de las predicciones de todos los modelos para cada jugador, y se crea una nueva serie llamada `ensemble_predictions`. Esto se hace con el fin de combinar las predicciones de todos los modelos y obtener una estimación más robusta del rendimiento de cada jugador en acciones a balón parado ofensivas.

Finalmente, se devuelve el DataFrame `top_players_with_features`, que contiene información sobre los 10 mejores jugadores, incluyendo sus nombres, puntuaciones de 'Set_Piece_Performance' y los valores de las características utilizadas para realizar las predicciones. Por último, se ejecuta el método `store_top_players_in_mongodb` se encarga de almacenar la información de los jugadores top, identificados previamente, en una colección de MongoDB. El propósito de este método es guardar los resultados del análisis para que puedan ser consultados, visualizados o utilizados más adelante en otras partes del sistema o en futuros análisis.

2.6. Evaluar y comparar el rendimiento de los modelos entrenados utilizando métricas de evaluación adecuadas, como el error cuadrático medio, el coeficiente de determinación y la precisión, con el fin de seleccionar el modelo que proporcione las mejores predicciones.

Se realizó el siguiente método `evaluate_models` para evaluar el rendimiento de una lista de modelos previamente entrenados utilizando un conjunto de datos de prueba.

```
def evaluate_models(list_trained_models, x_test, y_test):
    # Evaluate the performance of the trained models
    list_evaluation_metrics = {}

    for model_name, model in list_trained_models.items():
        y_pred = model.predict(x_test)
        mse = mean_squared_error(y_test, y_pred)
        r2 = r2_score(y_test, y_pred)
        list_evaluation_metrics[model_name] = {'MSE': mse, 'R2': r2}

    # Convert the evaluation metrics dictionary to a pandas DataFrame
    evaluation_metrics_df = pd.DataFrame(list_evaluation_metrics).transpose()

    # Export the DataFrame to an Excel file
    evaluation_metrics_df.to_excel("model_evaluation_metrics.xlsx")

    # Return the evaluation metrics
    return list_evaluation_metrics
```

Imagen 12. Método que permite la evaluación de los modelos utilizados

Al revisar las evaluaciones de cada modelo luego de haber realizado el entrenamiento, estos fueron los valores logrados:

	MSE	R2
Linear Regression	0,00000000000000000545440631	1
Decision Tree	0,00037449362043521100000000	0,963859
Random Forest	0,00018888956914478700000000	0,981771

- La regresión lineal tiene un MSE muy cercano a 0 y un R2 igual a 1. Esto indica que el modelo de regresión lineal ajusta perfectamente los datos de prueba.
- El árbol de decisión tiene un MSE de 0.000374 y un R2 de 0.963859. Estos valores muestran que el modelo de árbol de decisión tiene un buen rendimiento, aunque no es perfecto. El R2 de 0.963859 indica que el modelo explica aproximadamente el 96.39% de la variabilidad en los datos.
- El bosque aleatorio tiene un MSE de 0.000189 y un R2 de 0.981771. Esto indica que el modelo de bosque aleatorio tiene un rendimiento aún mejor que el árbol de decisión. Un R2 de 0.981771 significa que el modelo explica aproximadamente el 98.18% de la variabilidad en los datos.

Dado que el resultado de la regresión lineal es sospechosamente perfecto, es posible que haya sobreajuste. Por otro lado, tanto el árbol de decisión como el bosque aleatorio muestran buenos resultados, siendo el bosque aleatorio el que tiene un mejor rendimiento en términos de MSE y R2. Por lo tanto, el modelo de bosque aleatorio podría ser la mejor opción entre estos tres modelos para predecir el rendimiento en acciones a balón parado.

2.7. Implementar el modelo seleccionado en una herramienta o aplicación que permita a los clubes de fútbol y otros interesados acceder a las predicciones de desempeño en acciones a balón parado de carácter ofensivo de los jugadores y utilizar esta información para evaluar a jugadores emergentes, nuevos o canteranos.

Se desarrollo una herramienta tanto a nivel de Backend como de Frontend para poder realizar este objetivo. La herramienta de Backend permite exponer un API con todas las funcionalidades necesarias para la capa de presentación. La herramienta de Frontend contiene un dashboard de cada jugador de referencia (jugadores con alto puntaje en el índice de rendimiento de ABP de carácter ofensivo) y con los módulos de agregar un nuevo jugador al sistema y de comparar el mismo con cualquiera de los jugadores de referencia obtenidos.

Antes de ejecutar el proceso cabe destacar que fue primordial hacer una ejecución del “motor del sistema”, el cual es un programa en Python que contiene toda la lógica para cargar los datos de los jugadores en la base de datos ejecutando a través de ello todo el proceso de los objetivos anteriores, con lo cual se obtienen los mejores 10 jugadores de la temporada de la liga española con el mayor puntaje de performance de ABP de carácter ofensivo.

El sistema de Backend contiene lo siguiente:

Se desarrolló este sistema utilizando FastAPI para crear una API RESTful que interactúa con una base de datos MongoDB. Este sistema es adecuado para el objetivo de implementar el modelo seleccionado en una herramienta o aplicación que permita a los clubes de fútbol y otros interesados acceder a las predicciones de desempeño en acciones a balón parado de carácter ofensivo de los jugadores.



Imagen 13. API que contiene todos los métodos para realizar mantenimiento sobre los jugadores y las métricas

A continuación, se presenta las tecnologías utilizadas en este sistema:

- **FastAPI:** FastAPI es un marco moderno y rápido para crear API's en Python, lo que permite desarrollar aplicaciones con alto rendimiento y facilidad de uso. Además, FastAPI genera automáticamente documentación interactiva utilizando OpenAPI y JSON Schema, lo que facilita la colaboración y el uso de la API por parte de los interesados.
- **MongoDB:** MongoDB es una base de datos NoSQL que puede almacenar datos en forma de documentos. Es una opción adecuada para este proyecto, ya que permite almacenar y consultar datos flexibles sin necesidad de una estructura rígida de tabla. Además, MongoDB es escalable y se puede utilizar tanto en aplicaciones pequeñas como en grandes.

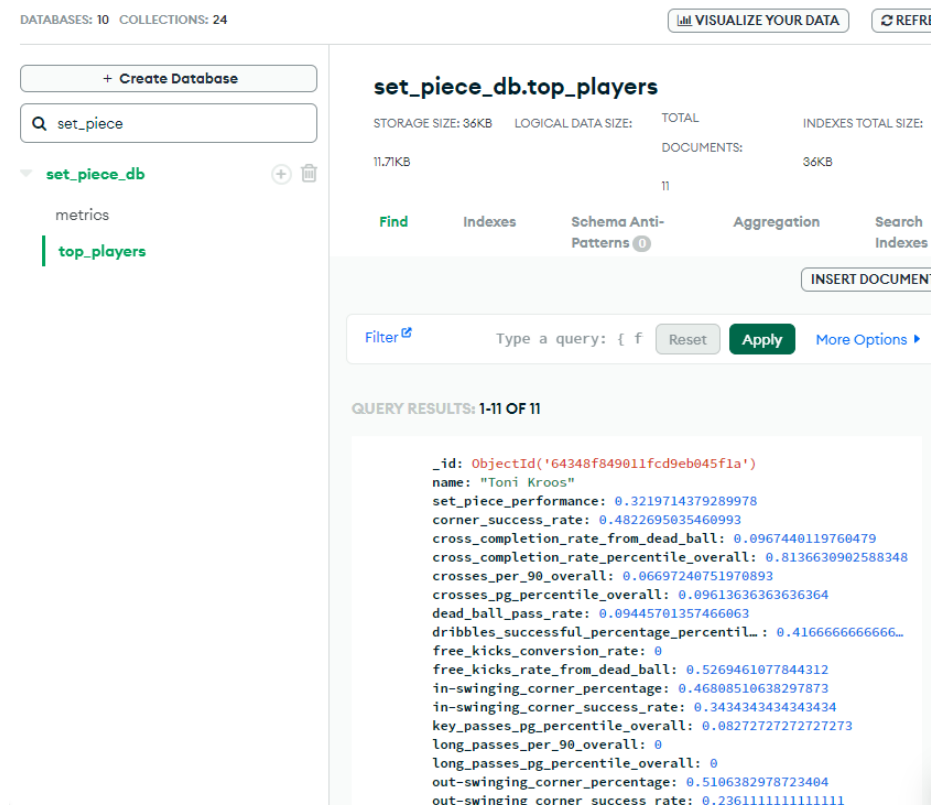


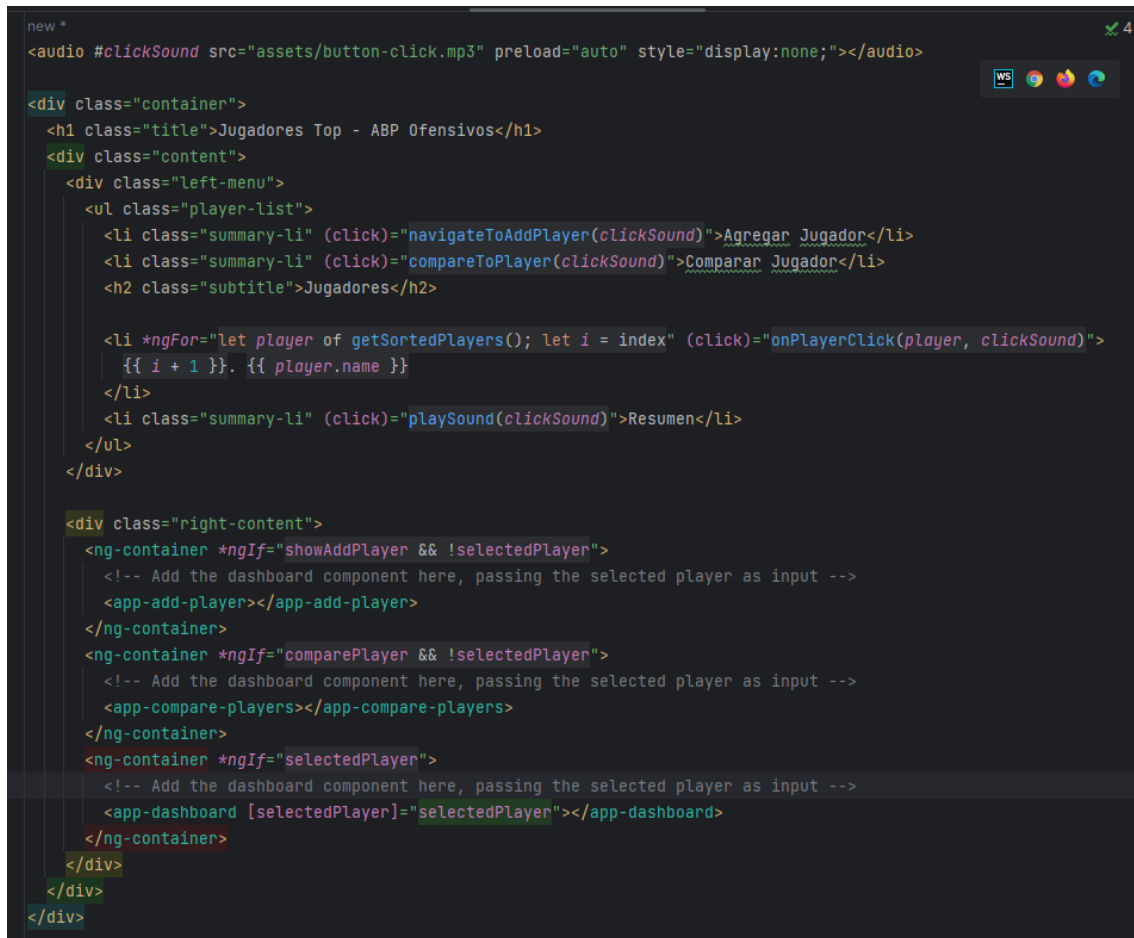
Imagen 14. Base de datos No-SQL que se ejecuta en la nube usando la capa gratuita de Mongo Atlas

En general, el sistema creado para el Backend parece adecuado para el objetivo del proyecto. Al utilizar FastAPI y MongoDB, se crea una API flexible y escalable que facilita el acceso a las predicciones de desempeño en acciones a balón parado de carácter ofensivo de los jugadores y la información relacionada.

El sistema de FrontEnd contiene lo siguiente:

El sistema de presentación utiliza Angular como marco principal para crear una aplicación web de una sola página (SPA) enfocada en mostrar información sobre jugadores en acciones de balón parado de carácter ofensivo. A continuación, se presenta un análisis de las tecnologías y módulos utilizados en el proyecto y cómo se ajustan a los objetivos del proyecto.

- Angular: Angular es un marco popular y maduro para construir aplicaciones web de una sola página. Ofrece una arquitectura modular y componentes reutilizables, lo que facilita el mantenimiento y la escalabilidad del proyecto. Además, Angular utiliza la detección de cambios y la renderización eficiente para ofrecer un rendimiento óptimo.

A screenshot of a code editor with a dark theme, showing Angular HTML code. The code defines a component structure with a 'container' class. Inside, there's a 'title' for 'Jugadores Top - ABP Ofensivos'. A 'content' div is split into a 'left-menu' and a 'right-content'. The 'left-menu' contains a 'player-list' with buttons for 'Agregar Jugador', 'Comparar Jugador', a list of players from 'getSortedPlayers()', and a 'Resumen' button. The 'right-content' uses three 'ng-container' blocks with 'ngIf' conditions to show 'app-add-player', 'app-compare-players', or 'app-dashboard' based on the state of 'selectedPlayer'. Comments indicate where dashboard components should be added.

```
new *
<audio #clickSound src="assets/button-click.mp3" preload="auto" style="display:none;"></audio>

<div class="container">
  <h1 class="title">Jugadores Top - ABP Ofensivos</h1>
  <div class="content">
    <div class="left-menu">
      <ul class="player-list">
        <li class="summary-li" (click)="navigateToAddPlayer(clickSound)">Agregar Jugador</li>
        <li class="summary-li" (click)="compareToPlayer(clickSound)">Comparar Jugador</li>
        <h2 class="subtitle">Jugadores</h2>

        <li *ngFor="let player of getSortedPlayers(); let i = index" (click)="onPlayerClick(player, clickSound)">
          {{ i + 1 }}. {{ player.name }}
        </li>
        <li class="summary-li" (click)="playSound(clickSound)">Resumen</li>
      </ul>
    </div>

    <div class="right-content">
      <ng-container *ngIf="showAddPlayer && !selectedPlayer">
        <!-- Add the dashboard component here, passing the selected player as input -->
        <app-add-player></app-add-player>
      </ng-container>
      <ng-container *ngIf="comparePlayer && !selectedPlayer">
        <!-- Add the dashboard component here, passing the selected player as input -->
        <app-compare-players></app-compare-players>
      </ng-container>
      <ng-container *ngIf="selectedPlayer">
        <!-- Add the dashboard component here, passing the selected player as input -->
        <app-dashboard [selectedPlayer]="selectedPlayer"></app-dashboard>
      </ng-container>
    </div>
  </div>
</div>
```

Imagen 15. Página html con el menú layout principal

- Material Design: La aplicación utiliza varios módulos de Angular Material, como MatButtonModule, MatCardModule, MatIconModule, MatToolbarModule, MatSidenavModule y MatListModule. Estos módulos proporcionan componentes de interfaz de usuario basados en el sistema de diseño Material Design de Google, lo que permite crear interfaces de usuario atractivas y coherentes con facilidad.
- NgChartsModule: Este módulo permite crear gráficos y visualizaciones de datos en la aplicación. Puede utilizarse para mostrar tendencias, distribuciones y comparaciones de métricas entre jugadores de manera visual e intuitiva.

En resumen, la capa de presentación creada con Angular y los módulos adicionales proporciona una solución sólida y escalable para mostrar y manipular información sobre jugadores en acciones a balón parado de carácter ofensivo. Al utilizar Angular Material, HttpClientModule y otros módulos, la aplicación ofrece una experiencia de usuario atractiva y eficiente que facilita la navegación y la interacción con los datos.

Las funcionalidades que tiene este sistema web son las siguientes:

- **Agregar Jugador** – Agregar un nuevo jugador proveniente de las canteras o exterior a La Liga de preferencia para comparar con los jugadores de referencia

Formulario para agregar un nuevo jugador al sistema. El formulario tiene un fondo azul oscuro con botones y campos de texto en un color más claro. En la parte superior hay tres botones numerados 1, 2 y 3. El campo 'Player Name' contiene el texto 'John Doe'. El campo 'Player Image URL (optional)' contiene el texto 'https://example.com/player-1'. A la derecha del formulario hay dos botones: 'Previous' y 'Next'.

Imagen 16. Componente de la aplicación para agregar un nuevo Jugador al sistema

- **Comparar Jugador**- Se puede comparar este jugador con otro jugador de referencia visualmente a través de la gráfica de radar superpuesta

Formulario para comparar dos jugadores del sistema. El formulario tiene un fondo azul oscuro con botones y campos de texto en un color más claro. Hay dos campos de selección de jugadores. El primer campo, 'Select Player 1', tiene 'Daniel Parejo' seleccionado. El segundo campo, 'Select Player 2', tiene 'Iago Aspas' seleccionado.

Imagen 17. Componente de la aplicación para comparar dos jugadores del sistema

- **Ver Jugador – Métricas Avanzadas**, incluyendo un Radar Chart y el puntaje de índice de performance de ABP de carácter ofensivo



Imagen 18. Jugadores Top- Acciones a balón parado de carácter ofensivo, pantalla que muestra el Radar Chart con métricas avanzadas y el índice de performance de ABP ofensivo