Málaga Tech Park
Centro Público Integrado de FP

LaLiga Santander

accenture

Trabajo Fin de Master
2022 / 2023
Curso Especialización Inteligencia Artificial y Big Data

Jaime Salado Muñoz

Descubriendo el talento a través del arte del análisis deportivo (ScoutLiga)

ÍNDICE DE CONTENIDO

1 : JUSTIFICACIÓN Y DESCRIPCIÓN DEL PROYECTO

El Proyecto scoutLiga se justifica por la necesidad de mejorar la toma de decisiones en el proceso de fichaje de jugadores en los equipos de fútbol de la Liga Profesional Española (Laliga Santander). En este proceso, los equipos necesitan evaluar el rendimiento de los jugadores para identificar a aquellos que tienen el potencial de mejorar su desempeño en el campo y, por lo tanto, contribuir al éxito del equipo.

El objetivo de este proyecto es utilizar técnicas de análisis de datos y aprendizaje automático para predecir el rendimiento de un jugador de fútbol en la Laliga. Se recopilarán y analizarán datos de los jugadores de la temporada 2021/2022, incluyendo su equipo, su posición y su desempeño en el campo.

La solución resultante permitirá a los equipos de fútbol de la Laliga mejorar su proceso de scouting, identificar a los jugadores que tienen el potencial de tener un mejor desempeño en el campo y tomar decisiones de fichaje más informadas.

Además se introducirá síntesis de voz para que podamos escuchar el resultado final de la predicción y no tengamos que leerlo.

2 : ¿DE DÓNDE PROVIENEN EL CONJUNTO DE DATOS ?

Los datos que se utilizaron en este TFM provienen de Sofascore una plataforma en línea que proporciona estadísticas y análisis de fútbol y otros deportes.

Estos datos son recolectados por un equipo de especialistas en deportes con Sistemas de seguimiento en tiempo real para obtener datos sobre los eventos deportivos a medida que ocurren.

Estos sistemas pueden incluir sensores y dispositivos de seguimiento colocados en el campo, así como software que analiza las transmisiones en vivo de los partidos.

A continuación el enlace al datasets utilizado :

- LaLigaSantander2021-2022.csv

3 : DESCRIPCIÓN DE LOS DATOS

En nuestro conjunto de datos utilizado nos encontraremos con diferentes campos de información obtenidos a lo largo de la temporada describidos a continuación :

- Id: Identificador de nuestra fila de datos.
- Nombre: Nombre de jugador del cual mostramos información.
- Posición: Zona donde juega el jugador : Portero, Defensa, Mediocampista, Delantero.
- Apariciones: Cantidad de partidos en los que el jugador participó..
- Minutos: Cantidad total de minutos jugados..
- Goles: Cantidad total del goles marcados por el jugador.
- Tiros totales: Número de tiros realizados por el jugador.
- Conversión goles %: Porcentaje de efectividad de cara a gol por parte del jugador.
- Regates Exito %: Porcentaje de efectividad de regates realizados por el jugador.
- Perdida Posesión:Número de veces que el jugador perdió el balón..
- Asistencias: Número de pases decisivos por parte del jugador para realizar un gol.
- Pases Total:Cantidad total de pases realizados por el jugador.
- Pases Claves: Número total de últimos pases que conduce a un tiro a portería de un compañero.
- Pases Precisos %:Porcentaje de efectividad de pases realizados por el jugador.
- Entradas:Número total de entradas realizadas por el jugador..
- Intercepciones:Cantidadtotal de veces que el jugador durante la temporada corta,desvía o bloquea un pase del rival.
- Total Duelos Ganados:Cantidad total de veces durante la temporada que un jugador gana el balón en la disputa ante un rival.
- Total Duelos Ganado %:Porcentaje de efectividad de disputas ganadas por el jugador.
- Faltas:Número total de infracciones cometidas por el jugador.
- Tarjeta Amarilla:Número total de amonestaciones por parte del jugador.
- Tarjeta Roja:Número total de expulsiones por parte del jugador .
- Paradas Partido %:Porcentaje de efectividad de paradas realizadas por partido de un jugador (portero)
- Goles Encajados:Cantidad total de goles recibidas por el jugador durante la temporada. (portero)
- Imbatido:Número de partidos donde el jugador no recibió gol (portero)
- Rendimiento:Nota final del jugador dependiendo del desempeño realizado.

4 : IMPORTACIÓN DE LIBRERÍAS

Para la realización de este proyecto tenemos la necesidad de importar algunas librerias encargadas de hacer tareas específicas :

- pandas : proporciona estructuras de datos y herramientas de manipulación y análisis de datos en formato de tabla en Python.
- matplotlib proporciona visualización de datos nos permite crear diferentes tipos de gráficas.
- numpy : permite realizar cálculos numéricos y manipulación de datos.
- seaborn : permite crear visualizaciones estadísticas atractivas e informativas.

Más adelante para ir completando el proyecto instalaremos :

- sklearn : proporciona herramientas simples y eficientes para análisis predictivo de datos.
- gTTS : permite convertir texto en habla mediante la API de Google,útil para crear aplicaciones que requieren la síntesis de voz.

```
# importar librerias

import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import seaborn as sns
```

5 : CARGAR CONJUNTO DE DATOS

Los datos utilizados a lo largo del proyecto están contenidos en el fichero estadisticasjugadoresligasantander2021-2022.csv

Para trabajar cómodamente guardaremos con pandas la ruta del archivo en una variable llamada laliga.

```
# cargar datos

file_path = '/content/drive/MyDrive/estadisticasjugadoresligasantander2021-2022.csv'
laliga = pd.read_csv(file_path)
```

6 : EXPLORACIÓN Y VISUALIZACIÓN DE LOS DATOS

Haremos un estudio del conjunto de datos que contiene el datasets :

6.1 - EXPLORAR DATOS

En este apartado inspeccionaremos el conjunto de datos mostrando :

- Las 5 primeras filas de nuestro conjunto de datos está dividido en dos partes debido a la gran cantidad de columnas encontradas.
  - 1º Mostramos las trece primeras columnas.

```
# Mostramos 5 primeras filas de datos que nos encontramos primeras 13 columnas

laliga[["Id", "Nombre", "Posicion", "Apariciones", "Minutos", "Goles", "Tiros Totales", "Conversion goles %", "Regates Exito %", "Perdida Posesion", "Asistencias", "Pases Totales", "Pases Claves"]].head()
```

	Id	Nombre	Posicion	Apariciones	Minutos	Goles	Tiros Totales	Conversion goles %	Regates Exito %	Perdida Posesion	Asistencias	Pases Totales	Pases Claves
0	1	Karim Benzema	Delantero	32	2596	27	128	21.09	70.59	277	12	1220	63
1	2	Kike Hermoso	Defensa	1	90	1	2	50.00	100.00	11	0	60	0
2	3	Ousmane Dembele	Delantero	21	1411	1	46	2.17	65.56	333	13	666	46
3	4	Toni Kroos	Mediocampista	28	2108	1	46	2.17	52.94	219	3	2150	51
4	5	Dani Parejo	Mediocampista	33	2708	2	16	12.50	75.76	408	10	1990	67

  - 2º Mostramos las restantes que son 12 columnas.

```
# Mostramos 5 primeras filas de datos que nos encontramos 12 columnas restantes

laliga[["Pases Precisos %", "Entradas", "Intercepciones", "Total Duelos Ganados", "Total Duelos Ganado %", "Faltas", "Tarjeta Amarilla", "Tarjeta Roja", "Paradas Partido %", "Goles Encajados", "Imbatido", "Rendimiento"]].head()
```

	Pases Precisos %	Entradas	Intercepciones	Total Duelos Ganados	Total Duelos Ganado %	Faltas	Tarjeta Amarilla	Tarjeta Roja	Paradas Partido %	Goles Encajados	Imbatido	Rendimiento
0	85.16	11	4	74	45.96	13	0	0	0	0	0	7.73
1	83.33	2	0	9	75.00	1	0	0	0	0	0	7.70
2	84.38	24	8	104	53.89	11	3	0	0	0	0	7.53
3	94.88	34	12	90	56.25	20	4	0	0	0	0	7.44
4	87.64	43	29	140	59.32	16	3	0	0	0	0	7.41

- Tipo de datos encontrados, si la columna contiene números(float,enteros),texto.

```
# Tipo de datos encontrados

laliga.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 592 entries, 0 to 591
Data columns (total 26 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Id                     592 non-null    int64
 1   Equipo                 592 non-null    object
 2   Nombre                 592 non-null    object
 3   Posicion               592 non-null    object
 4   Apariciones            592 non-null    int64
 5   Minutos                592 non-null    int64
 6   Goles                  592 non-null    int64
 7   Tiros Totales          592 non-null    int64
 8   Conversion goles %     592 non-null    float64
 9   Regates Exito %        592 non-null    float64
 10  Perdida Posesion       592 non-null    int64
 11  Asistencias            592 non-null    int64
 12  Pases Totales          592 non-null    int64
 13  Pases Claves           592 non-null    int64
 14  Pases Precisos %       592 non-null    float64
 15  Entradas               592 non-null    int64
 16  Intercepciones         592 non-null    int64
 17  Total Duelos Ganados   592 non-null    int64
 18  Total Duelos Ganado %  592 non-null    float64
 19  Faltas                 592 non-null    int64
 20  Tarjeta Amarilla       592 non-null    int64
 21  Tarjeta Roja           592 non-null    int64
 22  Paradas Partido %      592 non-null    int64
 23  Goles Encajados        592 non-null    int64
 24  Imbatido               592 non-null    int64
 25  Rendimiento            592 non-null    float64
dtypes: float64(5), int64(18), object(3)
memory usage: 120.4+ KB
```

- Resumen estadístico que incluye varias medidas de tendencia central, dispersión y forma de la distribución de los datos numéricos de cada columna. Muestra la cantidad de filas y columnas, la media, desviación estándar, valor mínimo, valor máximo y los percentiles de cada columna numérica del conjunto de datos. Esto puede ser útil para tener una idea general de los valores de las variables y para detectar valores extremos o valores perdidos.

```
# Resumen estadistico

laliga.describe()
```

	Id	Apariciones	Minutos	Goles	Tiros Totales	Conversion goles %	Regates Exito %	Perdida Posesion	Asistencias	Pases Totales	...	Intercepciones	Total Duelos Ganados	Total Duelos Ganado %	Faltas	Tarjeta Amarilla	Tarjeta Roja	Paradas Partido %	Goles Encajados	Imbatido	Rendimiento
count	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	...	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000	592.000000
mean	296.500000	19.907095	1273.832770	1.565878	15.099662	6.786368	50.968429	177.753378	1.155405	546.547297	...	11.719595	67.540541	49.680236	16.988176	3.199324	0.150338	4.648649	1.582770	0.398649	6.759476
std	171.039956	11.459034	955.322021	2.933392	18.238740	10.553476	29.635883	146.437030	2.045052	492.880601	...	12.894171	59.761696	16.910238	15.401832	2.939457	0.402236	17.390845	7.461797	2.065722	0.243918
min	1.000000	1.000000	8.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.900000
25%	148.750000	9.000000	367.500000	0.000000	2.000000	0.000000	37.500000	47.750000	0.000000	127.750000	...	1.000000	15.750000	43.092500	3.000000	1.000000	0.000000	0.000000	0.000000	0.000000	6.630000
50%	296.500000	21.000000	1177.500000	0.000000	9.000000	0.000000	54.635000	154.500000	0.000000	438.000000	...	7.000000	59.000000	50.000000	14.500000	3.000000	0.000000	0.000000	0.000000	0.000000	6.740000
75%	444.250000	30.000000	1963.250000	2.000000	20.250000	11.190000	68.322500	263.000000	2.000000	822.750000	...	19.000000	99.250000	56.652500	26.000000	5.000000	0.000000	0.000000	0.000000	0.000000	6.900000
max	592.000000	38.000000	3420.000000	27.000000	128.000000	100.000000	100.000000	756.000000	22.000000	2769.000000	...	71.000000	372.000000	100.000000	77.000000	15.000000	2.000000	89.000000	60.000000	19.000000	7.730000

8 rows × 23 columns

- Mostrar cantidad de datos nulos: Pasaremos a buscar si hay valores nulos y que cantidad exacta tenemos en cada campo, ya que la mayoría de algoritmo no trabajan bien cuando faltan datos. En nuestro caso no encontramos ningún valor nulo.

```
# Mostrar cantidad de datos nulos en los diferentes campos

laliga.isna().sum()
```

```
Id                      0
Equipo                  0
Nombre                  0
Posicion                0
Apariciones             0
Minutos                 0
Goles                   0
Tiros Totales           0
Conversion goles %      0
Regates Exito %         0
Perdida Posesion        0
Asistencias             0
Pases Totales           0
Pases Claves            0
Pases Precisos %        0
Entradas                0
Intercepciones          0
Total Duelos Ganados    0
Total Duelos Ganado %   0
Faltas                  0
Tarjeta Amarilla        0
Tarjeta Roja            0
Paradas Partido %       0
Goles Encajados         0
Imbatido                0
Rendimiento             0
dtype: int64
```

6.2 - VISUALIZACIÓN DE DATOS

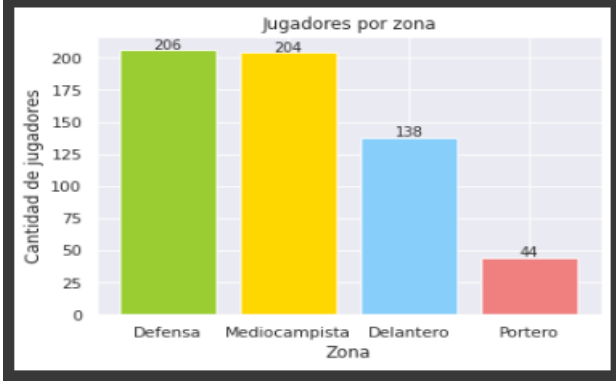
En este apartado mostraremos el conjunto de datos completo de una manera gráfica para poder tener una mejor visualización de datos relevantes de cada campo encontrado.

6.2.1 DISTRIBUCIÓN DE JUGADORES POR ZONAS

En esta sección mostraremos dos gráficos para mostrar la cantidad de jugadores por zona en la 1º División de la Liga profesional de fútbol en España, esto nos permite una rápida comparación entre las diferentes posiciones y es útil para entender la distribución de los jugadores de la competición.

- El primero es de barras y se puede ver que la mayoría de los jugadores se encuentran en las posiciones de defensa y mediocampista, con 206 y 204 jugadores respectivamente. Por otro lado, hay menos jugadores en las posiciones de delantero y portero, con 108 y 44 jugadores respectivamente.

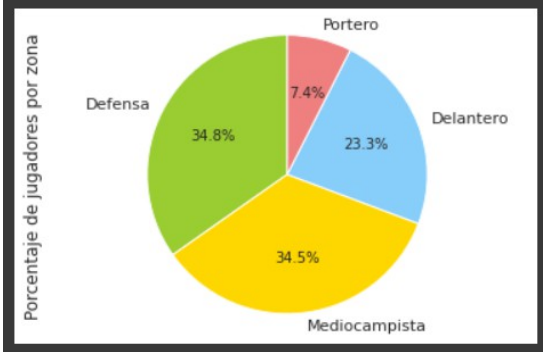
```
# contar la cantidad de jugadores en cada zona
jugadores_por_zona = laliga['Posicion'].value_counts()

# Crear una lista con los colores que se utilizarán en el gráfico
colores = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']

# crear un gráfico de barras
plt.bar(jugadores_por_zona.index, jugadores_por_zona.values, color=colores)

# mostrar los valores exactos en el gráfico
for i, v in enumerate(jugadores_por_zona.values):
    plt.text(i, v+1, str(v), ha='center')

# agregar títulos y etiquetas a los ejes
plt.title('Jugadores por zona')
plt.xlabel('Zona')
plt.ylabel('Cantidad de jugadores')

# mostrar la gráfica
plt.show()
```



- El segundo circular muestra la distribución porcentual de jugadores en cada zona. La zona de defensas es la más común, con un 34.8% de los jugadores. La zona de mediocampistas sigue muy de cerca con un 34.5%. La zona de delantero tiene un porcentaje significativamente menor de jugadores, con un 23.3%. Por último, la zona de portero es la menos común con un 7.4% de los jugadores.

  En general, la mayoría de los jugadores en La Liga se encuentran en las posiciones de defensa y mediocampista.

```
# Crear el gráfico circular
plt.pie(jugadores_por_zona, labels=jugadores_por_zona.index, colors=colores,
        autopct='%1.1f%%', startangle=90)

# Ajustar el tamaño del gráfico
plt.axis('equal')

plt.ylabel('Porcentaje de jugadores por zona')

# Mostrar el gráfico
plt.show()
```

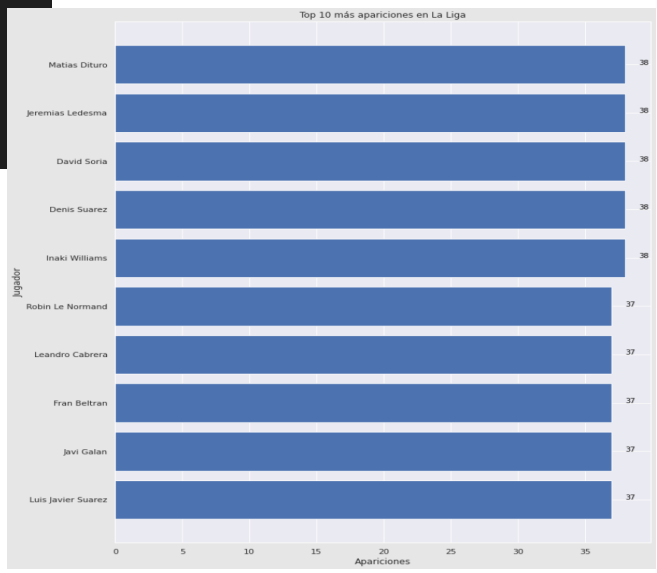


6.2.2 DISTRIBUCIÓN DE JUGADORES POR APARICIONES

En esta sección debido a la gran cantidad de datos mostraremos dos gráficos para mostrar la cantidad de apariciones de los jugadores a lo largo de la temporada.

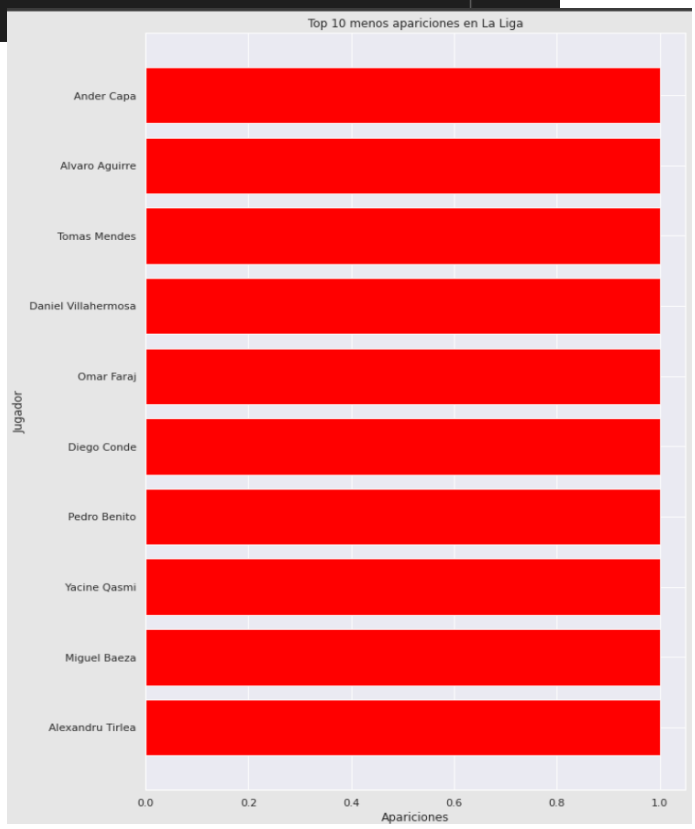
- El primer gráfico muestra los 10 jugadores con más apariciones a lo largo de la temporada.
  - 38 apariciones
    - Matias Dituro (Portero Celta Vigo) , Jeremías Ledesma (Portero Cádiz CF), David Soria (Portero Getafe CF), Denis Suarez (Mediocampista Celta Vigo), Iñaki Williams (Delantero Athletic Bilbao)
  - 37 aparciones
    - Robin Le Normand (Defensa Real Sociedad), Leandro Cabrera (Defensa Espanyol), Fran Beltran (Mediocampista Celta Vigo), Javi Galan (Defensa Celta Vigo), Luis Javier Suarez (Delantero Granada)

```
# Seleccionar los 10 jugadores con más apariciones
topmasapariciones = laliga.sort_values('Apariciones', ascending=False).head(10)
topmasapariciones = topmasapariciones.iloc[::-1]


# Crear el gráfico de barras
fig, ax = plt.subplots(figsize=(12, 15))
ax.barh(topmasapariciones['Nombre'], topmasapariciones['Apariciones'])

# Añadir etiquetas y título
ax.set_xlabel('Apariciones')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 más apariciones en La Liga')

# Agregar etiquetas a las barras
for i, v in enumerate(topmasapariciones['Apariciones']):
    ax.text(v + 1, i, str(v), color='black')

plt.show()
```



- El segundo gráfico muestra los 10 jugadores con menos apariciones a lo largo de la temporada.
  - 1 aparición
    - Ander Capa (Defensa Athletic Bilbao), Alvaro Aguirre (Delantero Rayo Vallecano), Tomas Mendes (Mediocampista Deportivo Alavés), Daniel Villahermosa (Mediocampista Espanyol), Omar Faraj (Delantero Levante), Diego Conde (Portero Getafe CF), Pedro Benito (Delantero Cádiz CF), Yacine Qasmi (Delantero Rayo Vallecano), Miguel Baeza (Mediocampista Celta Vigo), Alexandru Tirlea (Defensa Deportivo Alavés)

```
# Seleccionar los 10 jugadores con menos apariciones
topmenosapariciones = laliga.sort_values('Apariciones', ascending=True).head(10)

# Crear el gráfico de barras
fig, ax = plt.subplots(figsize=(10, 15))
ax.barh(topmenosapariciones['Nombre'], topmenosapariciones['Apariciones'], color='red')

# Añadir etiquetas y título
ax.set_xlabel('Apariciones')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 menos apariciones en La Liga')

plt.show()
```

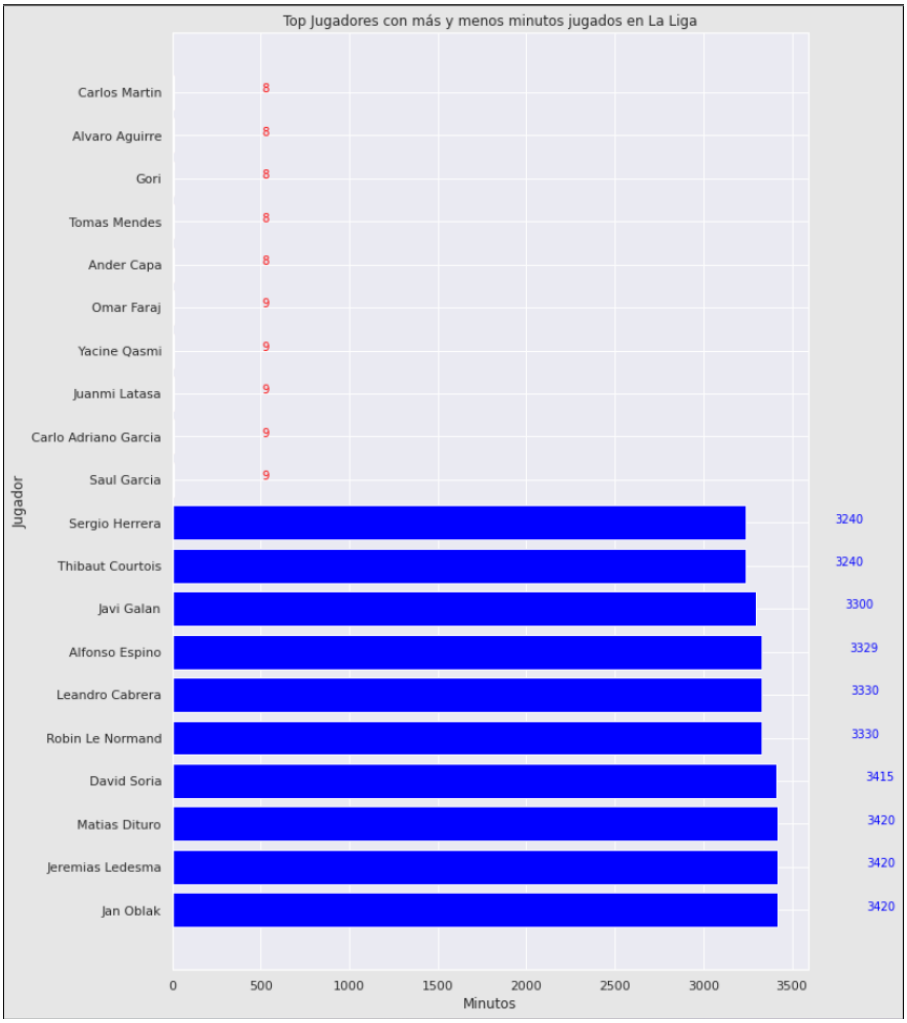


6.2.3 Distribución de jugadores por minutos

En esta sección debido a la gran cantidad de datos, se muestra una comparativa de los 10 jugadores con más minutos jugados y los 10 jugadores con menos minutos jugados durante la temporada.

Se muestra una visualización en forma de barras horizontales que permite ver de la cantidad de minutos que cada jugador ha disputado a lo largo de la temporada. Además, se utiliza un código de colores para diferenciar a los jugadores con más minutos (en azul) de los jugadores con menos minutos (en rojo).

Esta información puede ser de gran utilidad para conocer la importancia y el rendimiento de los jugadores en sus respectivos equipos.



Jugadores con más minutos :

- 3420 minutos

  Jan Oblak (Portero Atlético Madrid), Jeremías Ledesma(Portero Cádiz CF),

  Matias Dituro (Portero Celta Vigo)

- 3415 minutos

  David Soria (Portero Getafe CF)

- 3330 minutos

  Robin Le Normand (Defensa Real Sociedad), Leandro Cabrera (Defensa Espanyol)

- 3329 minutos

  Alfonso Espino (Defensa Cádiz CF)

- 3300 minutos

  Javi Galán (Defensa Celta vigo)

- 3240 minutos

  Thibaut Courtois (Portero Real Madrid) , Sergio Herrera (Portero Osasuna)

Jugadores con menos minutos :

- 8 minutos

  Carlos Martín (Delantero Atlético Madrid), Álvaro Aguirre (Delantero Rayo Vallecano), Gori (Mediocampista Espanyol), Tomas Mendes (Mediocampista Deportivo Alavés), Ander Capa (Defensa Athletic Bilbao)

- 9 minutos

  Omar Faraj (Delantero Levante), Yacine Qasmi(Delantero Rayo Vallecano),

  Juanmi Latasa (Delantero Real Madrid), Carlo Adriano García (Mediocampista Villarreal), Saul Garcia (Defensa Deportivo Alavés)

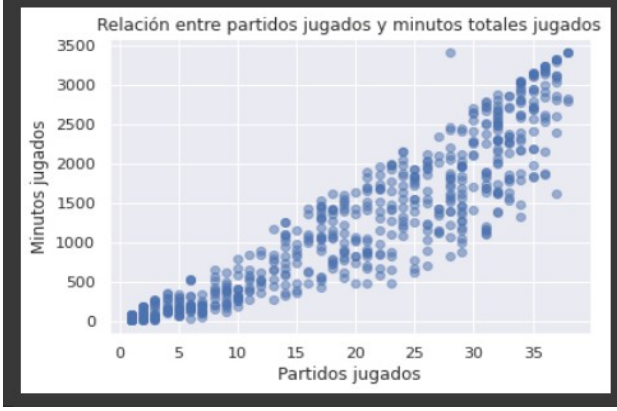
Si nos paramos un momento a comparar los datos de apariciones con minutos jugados vemos que muchos jugadores coinciden, tanto en los 10 con más apariciones / minutos y los 10 con menos apariciones / minutos. Así que podemos decir que hay una relación entre los dos campos.

6.2.4 RELACIÓN ENTRE PARTIDOS JUGADOS(APARICIONES) Y MINUTOS TOTALES JUGADOS

En esta sección mostramos una gráfica de dispersión donde podemos ver cómo se relaciona el número de partidos jugados (apariciones) con los minutos totales jugados para cada jugador. Podríamos encontrar patrones interesantes, como que los jugadores que juegan más partidos no siempre son los que acumulan más minutos totales, o viceversa.

Normalmente es más probable que los jugadores que juegan más minutos también hayan aparecido en más partidos. Pero esto no siempre es cierto, ya que hay jugadores que pueden tener un gran número de minutos en menos partidos debido a lesiones, suspensión, rotación del equipo, entre otros factores.

```
# Seleccionar los datos relevantes
relacionpartidosminutos = laliga[['Apariciones', 'Minutos']]

# Crear la gráfica de dispersión
fig, ax = plt.subplots()
ax.scatter(relacionpartidosminutos['Apariciones'], relacionpartidosminutos['Minutos'], alpha=0.5)

# Añadir etiquetas y título
ax.set_xlabel('Partidos jugados')
ax.set_ylabel('Minutos jugados')
ax.set_title('Relación entre partidos jugados y minutos totales jugados')

plt.show()
```



6.2.5 Distribución de jugadores por goles (top 10)

En esta sección debido a la gran cantidad de datos, se muestra solo los 10 goleadores más importantes de la competición, esto permite comparar el rendimiento goleador de los jugadores y una visualización más clara de los jugadores que han destacado por su capacidad goleadora a lo largo de la temporada. Encontramos con:

- 27 goles

  Karim Benzema (Delantero Real Madrid)
- 18 goles

  Iago Aspas (Delantero Celta Vigo)
- 17 goles

  Vinicius Junior (Delantero Real Madrid), Raúl De Tomás (Delantero Espanyol)
- 16 goles

  Enes Unal (Delantero Getafe CF), Juanmi (Delantero Real Betis Balompie)
- 14 goles

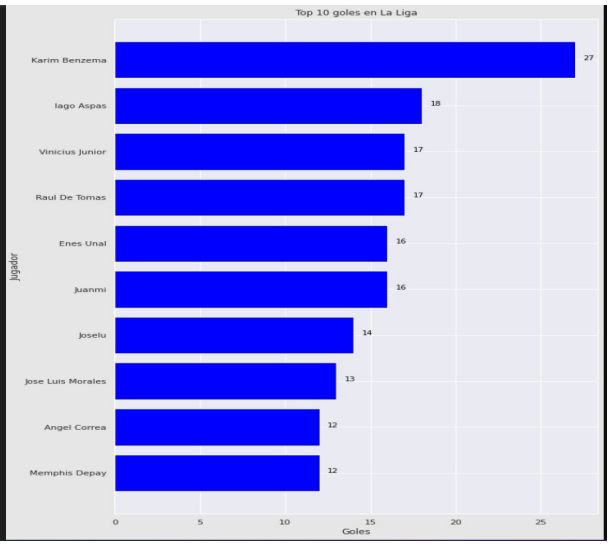
  Joselu (Delantero Deportivo Alavés)
- 13 goles

  José Luis Morales (Delantero Levante)
- 12 goles

  Ángel Correa (Delantero Atlético Madrid), Memphis Depay (Delantero Barcelona)

Podemos apreciar que los máximos goleadores de la competición son Delanteros.

```
# Seleccionar los 10 jugadores con más goles
top_goles = laliga.sort_values('Goles', ascending=False).head(10)

# Ordenar los jugadores por goles
top_goles = top_goles.iloc[::-1]

# Crear el gráfico de barras horizontal
fig, ax = plt.subplots(figsize=(10, 15))
ax.barh(top_goles['Nombre'], top_goles['Goles'], color='blue')


# Añadir etiquetas y título
ax.set_xlabel('Goles')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 goles en La Liga')

# Mostrar el número exacto de goles en la gráfica
for i, v in enumerate(top_goles['Goles']):
    ax.text(v + 0.5, i, str(v), color='black')
plt.show()
```



6.2.6 Distribución de jugadores por Tiros totales (top 10)

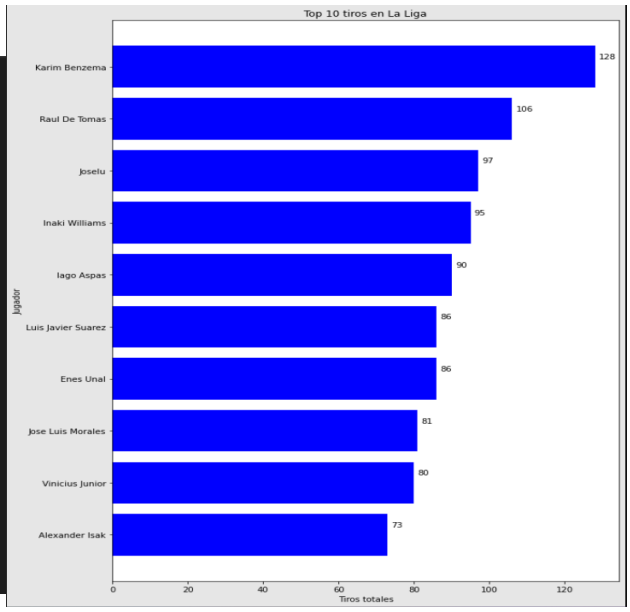
En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más tiros en la competición.

Se puede ver quiénes son los jugadores que han tenido más oportunidades de marcar goles. Además, vemos el número exacto de tiros que han realizado, lo que permite compararlos y ver cuánto se diferencian en términos de oportunidades de gol.

También ayuda a entender la relación entre la cantidad de tiros y los goles marcados, lo que puede ser importante para evaluar la efectividad de los jugadores. Vemos con:

- 128 tiros : Karim Benzema (Delantero Real Madrid)
- 106 tiros: Raúl De Tomás (Delantero Espanyol)
- 97 tiros : Joselu (Delantero Deportivo Alavés)
- 95 tiros : Iñaki Williams (Delantero Athletic Bilbao)
- 90 tiros: Iago Aspas (Delantero Celta Vigo)
- 86 tiros : Enes Unal (Delantero Getafe CF), Luis Javier Suarez (Delantero Granada)
- 81 tiros: José Luis Morales (Delantero Levante)
- 80 tiros : Vinicius Junior (Delantero Real Madrid),
- 73 tiros : Alexander Isak (Delantero Real Sociedad)

```python
# Seleccionar los 10 jugadores con más tiros
top_tiros = laliga.sort_values('Tiros Totales', ascending=False).head(10)

# Ordenar los jugadores por número de tiros
top_tiros = top_tiros.iloc[::-1]

# Crear el gráfico de barras horizontal
fig, ax = plt.subplots(figsize=(10, 15))
ax.barh(top_tiros['Nombre'], top_tiros['Tiros Totales'], color='blue')

# Añadir etiquetas y título
ax.set_xlabel('Tiros totales')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 tiros en La Liga')

# Añadir las etiquetas con los valores exactos de los tiros

for i, v in enumerate(top_tiros['Tiros Totales']):
    ax.text(v + 1, i + .15, str(v), color='black')

plt.show()
```

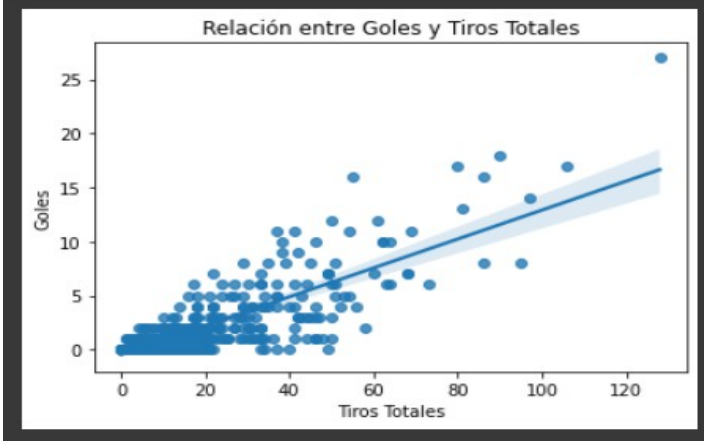


6.2.7 Relación entre goles y tiros totales

En esta sección podemos ver una gráfica de dispersión donde vemos cómo se relaciona el número de tiros torales y los goles de cada jugador durante la temporada.

Esta relación es una medida importante del rendimiento de un jugador. Si un jugador tiene un alto número de tiros a portería pero pocos goles, esto puede indicar que tiene problemas para convertir las oportunidades de gol que crea. Por otro lado, si un jugador tiene un alto número de goles con pocos tiros a portería, puede ser que sea muy efectivo y tenga una gran habilidad para marcar goles. En la gráfica, la línea de regresión indica la relación. Si la línea se inclina hacia arriba, esto indica que hay una relación positiva entre los dos,a medida que aumentan los tiros a portería, también aumentan los goles.

```
# Seleccionar los datos relevantes y eliminar los jugadores con cero tiros totales
relaciongoltiro = laliga[['Nombre', 'Goles', 'Tiros Totales']]
relaciongoltiro = relaciongoltiro[relaciongoltiro['Tiros Totales'] > 0]

# Crear la gráfica de dispersión con línea de regresión
sns.regplot(x='Tiros Totales', y='Goles', data=laliga)

# Añadir etiquetas y título
plt.xlabel('Tiros Totales')
plt.ylabel('Goles')
plt.title('Relación entre Goles y Tiros Totales')

plt.show()
```



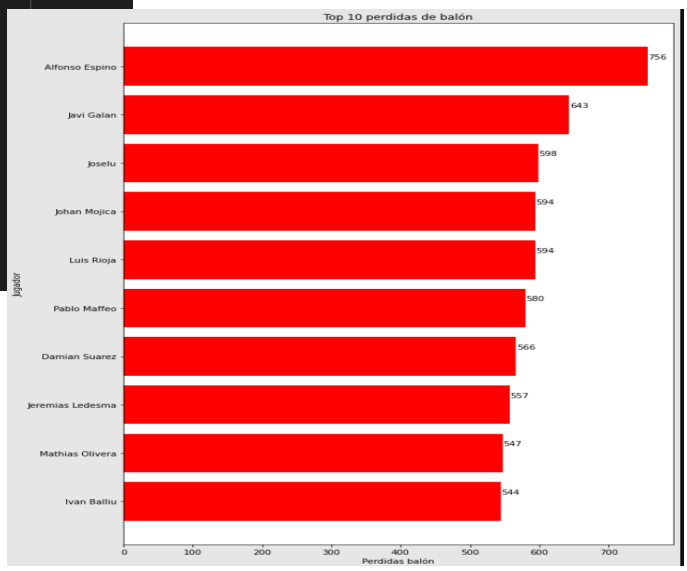
6.2.8 Distribución de jugadores según perdidas de balón (top 10)

En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más perdidas de balón en la competición.

Esta visualización permite comparar fácilmente la cantidad de perdidas de balón de cada jugador y determinar quiénes son los jugadores que tienen un mayor número de pérdidas de posesión en la competición. Podemos visualizar con :

- 756 perdidas : Alfonso Espino (Defensa Cádiz CF)
- 643 perdidas : Javi Galán (Defensa Celta Vigo)
- 598 perdidas : Joselu (Delantero Deportivo Alavés)
- 594 perdidas : Johan Mojica (Defensa Elche CF), Luis Rioja (Mediocampista Deportivo Alavés)
- 580 perdidas : Pablo Maffeo (Defensa RCD Mallorca)
- 566 perdidas : Damian Suárez (Defensa Getafe CF)
- 557 perdidas : Jeremías Ledesma (Portero Cádiz CF)
- 547 perdidas : Mathias Olivera (Defensa Getafe CF)
- 544 perdidas : Ivan Balliu (Defensa Rayo Vallecano)

```
# Seleccionar los 10 jugadores con más perdidas de balón
topmasperdidasbalon = laliga.sort_values('Perdida Posesion', ascending=False).head(10)

# Ordenar los jugadores por número de perdidas
topmasperdidasbalon = topmasperdidasbalon.iloc[::-1]

# Crear el gráfico de barras horizontal
fig, ax = plt.subplots(figsize=(10, 15))
ax.barh(topmasperdidasbalon['Nombre'], topmasperdidasbalon['Perdida Posesion'], color='red')

# Añadir etiquetas y título
ax.set_xlabel('Perdidas balón')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 perdidas de balón')

# Añadir las etiquetas con los valores exactos de los tiros

for i, v in enumerate(topmasperdidasbalon['Perdida Posesion']):
    ax.text(v + 1, i + .15, str(v), color='black')

plt.show()
```



6.2.9 Distribución de jugadores según asistencias de gol (top 10)

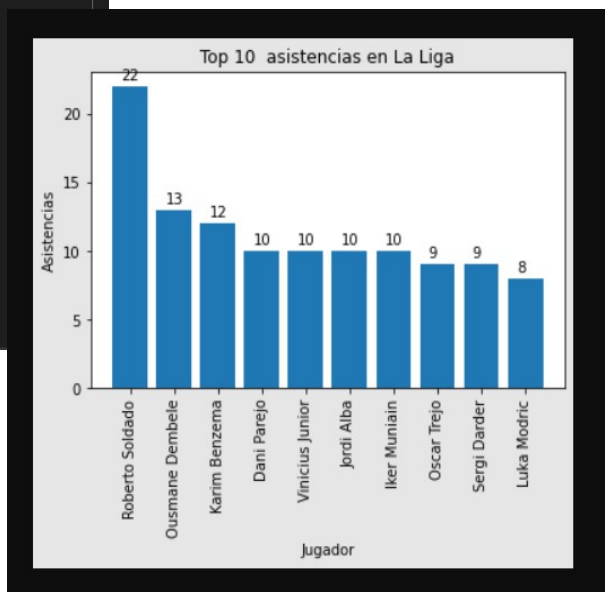
En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más asistencia de gol en la competición.

Esta visualización permite comparar fácilmente la cantidad de asistencias de gol de cada jugador y determinar quiénes son los jugadores que han proporcionado un mayor número de asistencias en la competición.

Las asistencias son pases que llevan directamente a un gol, por lo que los jugadores con más asistencias son aquellos que contribuyen más directamente a los goles del equipo. Encontramos con :

- 22 asistencias : Roberto Soldado (Delantero Levante)
- 13 asistencias : Ousmane Dembele (Delantero Barcelona)
- 12 asistencias : Karim Benzema (Delantero Real Madrid)
- 10 asistencias : Dani Parejo (Mediocampista Villarreal), Vinicius Junior (Delantero Real Madrid), Jordi Alba (Defensa Barcelona), Iker Muniain (Mediocampista Athletic Bilbao)
- 9 asistencias : Oscar Trejo (Mediocampista Rayo Vallecano), Sergi Darder (Mediocampista Espanyol), Luka Modric (Mediocampista Real Madrid)

```
# Seleccionar los datos relevantes y ordenarlos por número de asistencias
asistencias = laliga[['Nombre', 'Asistencias']]
asistencias = asistencias.sort_values(by='Asistencias', ascending=False).head(10)

# Crear la gráfica de barras
fig, ax = plt.subplots()
ax.bar(asistencias['Nombre'], asistencias['Asistencias'])

# girar las etiquetas
for label in ax.get_xticklabels():
    label.set_rotation(90)

# Añadir el número exacto en la etiqueta de cada barra
for i, v in enumerate(asistencias['Asistencias']):
    ax.text(i-0.2, v+0.5, str(v), fontsize=10, color='black')

# Añadir etiquetas y título
ax.set_xlabel('Jugador')
ax.set_ylabel('Asistencias')
ax.set_title('Top 10  asistencias en La Liga')

plt.show()
```



6.2.10 Distribución de jugadores según número de pases totales (top 10)

En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más pases en la competición.

Esta visualización permite comparar fácilmente la cantidad de pases totales de cada jugador y determinar quiénes son los jugadores que han realizado un mayor número de pases en la competición.

Es útil para analizar la capacidad de un jugador para mantener la posesión del balón y para la creación de juego. Vemos con:

2769 pases : Sergio Busquets (Mediocampista Barcelona)

2392 pases : Jordi Alba (Defensa Barcelona)

2181 pases : Diego Carlos (Defensa Sevilla)

2153 pases : Robin Le Normand (Defensa Real Sociedad)

2150 pases : Toni Kross (Mediocampista Real Madrid)

2099 pases : Jules Kounde (Defensa Sevilla)

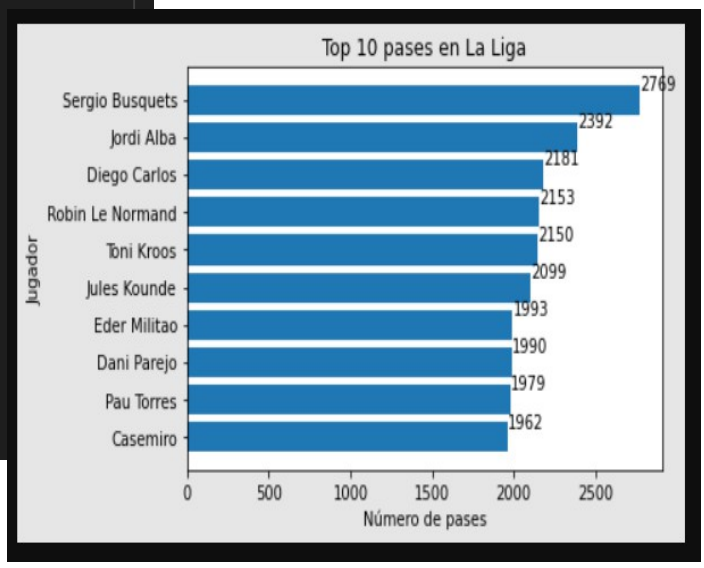
1993 pases : Eder Militao (Defensa Real Madrid)

1990 pases : Dani Parejo (Mediocampista Villarreal)

1979 pases : Pau Torres (Defensa Villarreal)

1962 pases : Casemiro (Mediocampista Real Madrid)

```python
# Seleccionar los datos relevantes y ordenarlos por número de pases totales
totalpases = laliga[['Nombre', 'Pases Totales']]
totalpases = totalpases.sort_values(by='Pases Totales', ascending=False).head(10)

# ordenar

totalpases = totalpases.iloc[::-1]


# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(totalpases['Nombre'], totalpases['Pases Totales'])

# Añadir etiquetas y título
ax.set_xlabel('Número de pases')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 pases en La Liga')

# Añadir el número exacto de pases en la gráfica
for i, v in enumerate(totalpases['Pases Totales']):
    ax.text(v + 3, i + .25, str(v), color='black')

plt.show()
```



6.2.11 DISTRIBUCIÓN DE JUGADORES SEGÚN NÚMERO DE PASES CLAVES (TOP 10)

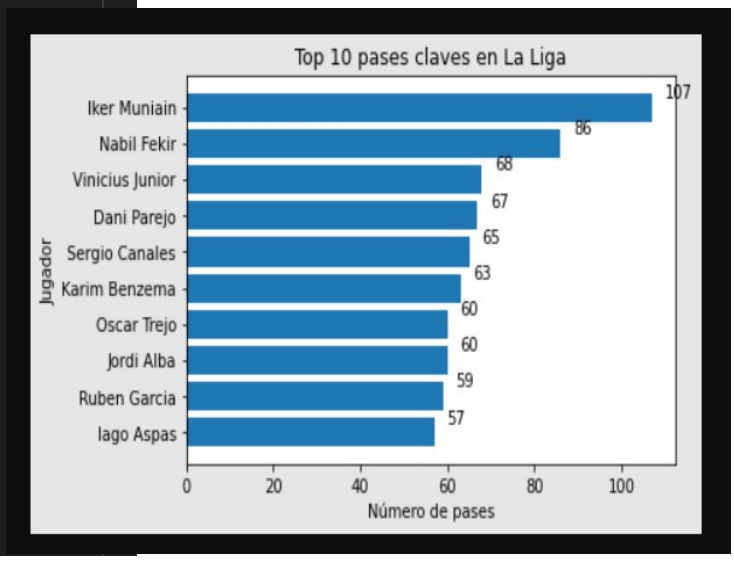
En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más pases claves en la competición.

Esta visualización permite comparar fácilmente la cantidad de pases claves de cada jugador y determinar quiénes son los jugadores que han proporcionado un mayor número de pases claves en la competición.

Estos pases son aquellos que preceden a un intento de tiro a puerta, y son un indicador importante de la capacidad de un jugador para crear oportunidades de gol para su equipo. Encontramos con:

- 107 pases claves : Iker Muniain (Mediocampista Athletic Bilbao)
- 86 pases claves : Nabil Fekir (Mediocampista Real Betis Balompie)
- 68 pases claves : Vinicius Junior (Delantero Real Madrid)
- 67 pases claves : Dani Parejo (Mediocampista Villarreal)
- 65 pases claves : Sergio Canales (Mediocampista Real Betis Balompie)
- 63 pases claves : Karim Benzema (Delantero Real Madrid)
- 60 pases claves : Oscar Trejo (Mediocampista Rayo Vallecano), Jordi Alba (Defensa Barcelona)
- 59 pases claves : Rubén García (Mediocampista Osasuna)
- 57 pases claves : Iago Aspas (Delantero Celta Vigo)

```
# Seleccionar los datos relevantes y ordenarlos por número de pases claves
pasesclaves = laliga[['Nombre', 'Pases Claves']]
pasesclaves = pasesclaves.sort_values(by='Pases Claves', ascending=False).head(10)

# ordenar

pasesclaves = pasesclaves.iloc[::-1]

# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(pasesclaves['Nombre'], pasesclaves['Pases Claves'])

# Añadir etiquetas y título
ax.set_xlabel('Número de pases')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 pases claves en La Liga')

# Añadir el número exacto de pases en la gráfica
for i, v in enumerate(pasesclaves['Pases Claves']):
    ax.text(v + 3, i + .25, str(v), color='black')

plt.show()
```

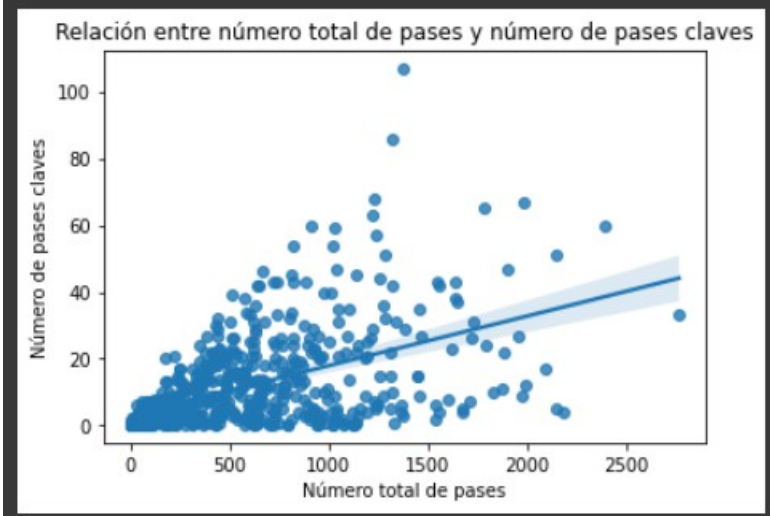


6.2.12 Relación entre total de pases y pases claves

Tras realizar la visualización de datos de pases totales y pases claves vemos que hay una relación entre ambos campos, mostraremos esta relación con una gráfica de dispersión.

Podremos ver que es positiva, a medida que aumenta el número total de pases también tiende a aumentar el número de pases claves.

```
# Seleccionar los datos relevantes
datos_pases = laliga[['Pases Totales', 'Pases Claves']]

# Crear la gráfica de dispersión
sns.regplot(x="Pases Totales", y="Pases Claves", data=datos_pases)

# Añadir etiquetas y título
plt.xlabel('Número total de pases')
plt.ylabel('Número de pases claves')
plt.title('Relación entre número total de pases y número de pases claves')

plt.show()
```



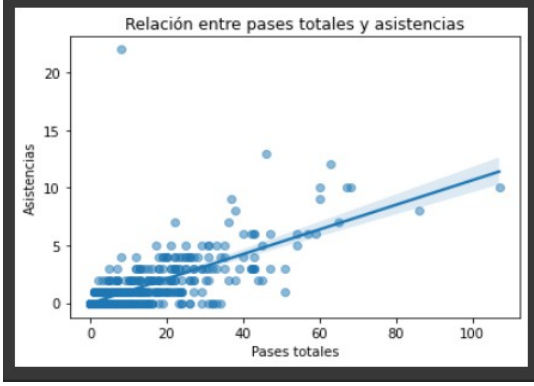
6.2.13 Relación entre pases claves y asistencias

Al igual que lo anterior, también encontramos una relación entre los pases claves y las asistencias, veremos como se relacionan a través de una gráfica de dispersión.

La relación entre los pases claves y las asistencias puede ayudar a medir la efectividad de un jugador en la creación y conversión de oportunidades de gol para su equipo.

Vemos que existe una correlación positiva, lo que significa que a medida que aumenta el número de pases claves, es más probable que un jugador dé una asistencia. Sin embargo,hay varios factores que pueden influir en el número de asistencias de un jugador, como la calidad del rematador, la estrategia del equipo, la posición del jugador en el campo.

```
# Seleccionar los datos relevantes
relacion_asist_claves = laliga[['Asistencias', 'Pases Claves']]

# Crear la gráfica de dispersión
sns.regplot(x='Pases Claves', y='Asistencias', data=relacion_asist_claves, scatter_kws={'alpha':0.5})

# Añadir etiquetas y título
plt.xlabel('Pases totales')
plt.ylabel('Asistencias')
plt.title('Relación entre pases totales y asistencias')

plt.show()
```



6.2.14 Distribución de jugadores según número total de entradas (top 10)

En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más entradas en la competición.

Una entrada se refiere a la acción de recuperar el balón de un oponente. Esta estadística puede ser útil para evaluar la capacidad defensiva de un jugador.

Podremos comparar la capacidad defensiva de diferentes jugadores e identificar a los jugadores que tienen un alto número de entradas. Podemos ver con:

- 108 entradas : Javi Galán (Defensa Celta Vigo)
- 107 entradas : Alfonso Espino (Defensa Cádiz CF)
- 102 entradas : Oscar Valentín (Mediocampista Rayo vallecano)
- 97 entradas : Guido Rodríguez (Mediocampista Real Betis Balompie)
- 93 entradas : Sergio Busquets (Mediocampista Barcelona)
- 87 entradas : Santi Comesaña (Mediocampista Rayo Vallecano)
- 84 entradas : Pablo Maffeo (Defensa RCD Mallorca)
- 83 entradas : Nacho Vidal (Defensa Osasuna)
- 79 entradas : Casemiro (Mediocampista Real Madrid), Mathias Olivera (Defensa Getafe CF)

```python
# Seleccionar los datos relevantes y ordenarlos por número de entradas
entradas = laliga[['Nombre', 'Entradas']]
entradas = entradas.sort_values(by='Entradas', ascending=False).head(10)

# ordenar

entradas = entradas.iloc[::-1]


# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(entradas['Nombre'], entradas['Entradas'])

# Añadir etiquetas y título
ax.set_xlabel('Número de entradas')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 entradas en La Liga')

# Añadir el número exacto de pases en la gráfica
for i, v in enumerate(entradas['Entradas']):
    ax.text(v + 3, i + .25, str(v), color='black')

plt.show()
```

6.2.15 Distribución de jugadores según número total de entradas (top 10)

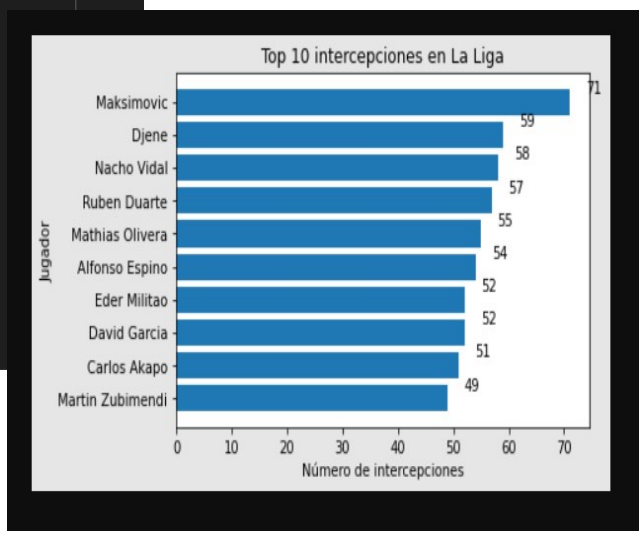
En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más intercepciones en la competición.

Son importante en el fútbol ya que mide la capacidad del jugador para recuperar el balón interceptando un pase del equipo contrario.

útil para evaluar la capacidad de los jugadores para defender y recuperar el balón, e identificar a los jugadores más efectivos en este aspecto del juego. Encontramos con :

- 71 intercepciones : Maksimovic (Mediocampista Getafe CF)
- 59 intercepciones : Djene (Defensa Getafe CF)
- 58 intercepciones : Nacho Vidal (Defensa Osasuna)
- 57 intercepciones : Rubén Duarte (Defensa Deportivo Alavés)
- 55 intercepciones : Mathias Olivera (Defensa Getafe CF)
- 54 intercepciones : Alfonso Espino (Defensa Cádiz CF)
- 52 intercepciones : Eder Militao (Defensa Real Madrid) , David García (Defensa Osasuna)
- 51 intercepciones : Carlos Akapo (Defensa Cádiz CF)
- 49 intercepciones : Martín Zubimendi (Mediocampista Real Sociedad)

```python
# Seleccionar los datos relevantes y ordenarlos por número de intercepciones
intercepcion = laliga[['Nombre', 'Intercepciones']]
intercepcion = intercepcion.sort_values(by='Intercepciones', ascending=False).head(10)

# ordenar
intercepcion = intercepcion.iloc[::-1]


# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(intercepcion['Nombre'], intercepcion['Intercepciones'])

# Añadir etiquetas y título
ax.set_xlabel('Número de intercepciones')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 intercepciones en La Liga')

# Añadir el número exacto de pases en la gráfica
for i, v in enumerate(intercepcion['Intercepciones']):
    ax.text(v + 3, i + .25, str(v), color='black')

plt.show()
```



6.2.16 Distribución de jugadores según número de duelos ganados (top 10)

En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más duelos ganados en la competición.

Son acciones en las que un jugador gana una disputa física o técnica contra un oponente para recuperar la posesión del balón. Esto puede incluir acciones como ganar un balón aéreo, ganar una carrera, entradas ...

Puede ser útil para entrenadores y directores técnicos para tomar decisiones en cuanto a la selección de jugadores para un partido o para reforzar las áreas débiles del equipo en términos de duelos ganados. Podemos visualizar con :

- 372 duelos ganados: Joselu (Delantero Deportivo Alavés)
- 293 duelos ganados : Javi Galán (Defensa Celta Vigo)
- 275 duelos ganados : Mikel Merino (Mediocampista Real Sociedad), Enes Unal (Delantero Getafe CF)
- 272 duelos ganados : Lucas Torro (Mediocampista Osasuna)
- 263 duelos ganados : Mathias Olivera (Defensa Getafe CF)
- 251 duelos ganados : Pablo Maffeo (Defensa RCD Mallorca)
- 247 duelos ganados : Santi Comesaña (Mediocampista Rayo Vallecano)
- 243 duelos ganados : Nabil Fekir (Mediocampista Real Betis Balompie)
- 238 duelos ganados : Sergio Busquets (Mediocampista Barcelona)

```python
# Seleccionar los datos relevantes y ordenarlos por número de duelos ganados
duelos = laliga[['Nombre', 'Total Duelos Ganados']]
duelos = duelos.sort_values(by='Total Duelos Ganados', ascending=False).head(10)

# ordenar
duelos = duelos.iloc[::-1]


# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(duelos['Nombre'], duelos['Total Duelos Ganados'])

# Añadir etiquetas y título
ax.set_xlabel('Número de duelos ganados')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 duelos ganados en La Liga')

# Añadir el número exacto de pases en la gráfica
for i, v in enumerate(duelos['Total Duelos Ganados']):
    ax.text(v + 3, i + .25, str(v), color='black')

plt.show()
```

6.2.17. DISTRIBUCIÓN DE JUGADORES SEGÚN FALTAS COMETIDAS (TOP 10)

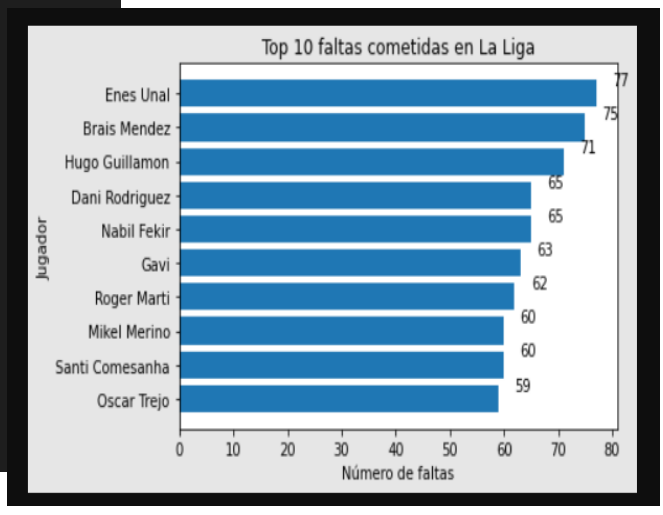
En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores con más faltas cometidas en la competición.

Las faltas pueden influir en el resultado de un partido. La cantidad de faltas que un jugador comete puede tener un impacto directo en el juego, ya que puede dar lugar a tarjetas amarillas o rojas, faltas peligrosas y pérdida de posesión del balón.

Esta información puede ser útil para los entrenadores y los equipos al evaluar el rendimiento de un jugador y trabajar en áreas en las que necesitan mejorar. Por ejemplo, si un jugador tiene una cantidad alta de faltas cometidas, es posible que el equipo necesite trabajar en su técnica de defensa para reducir la cantidad de faltas que comete y evitar consecuencias negativas. Resultados encontrados :

- 77 faltas : Enes Unal (Delantero Getafe CF)
- 75 faltas : Brais Mendez (Mediocampista Celta Vivo)
- 71 faltas : Hugo Guillamon (Defensa Valencia)
- 65 faltas : Dani Rodríguez (Mediocampista RCD Mallorca), Nabil Fekir (Mediocampista Real Betis Balompie)
- 63 faltas : Gavi (Mediocampista Barcelona)
- 62 faltas : Roger Martí (Delantero Levante)
- 60 faltas : Mikel Merino (Mediocampista Real Sociedad), Santi Comesaña (Mediocampista Rayo Vallecano)
- 59 faltas : Oscar Trejo (Mediocampista Rayo Vallecano)

```python
# Seleccionar los datos relevantes y ordenarlos por número de faltas cometidas
faltas = laliga[['Nombre', 'Faltas']]
faltas = faltas.sort_values(by='Faltas', ascending=False).head(10)

# ordenar

faltas = faltas.iloc[::-1]


# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(faltas['Nombre'], faltas['Faltas'])

# Añadir etiquetas y título
ax.set_xlabel('Número de faltas')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 faltas cometidas en La Liga')

# Añadir el número exacto de pases en la gráfica
for i, v in enumerate(faltas['Faltas']):
    ax.text(v + 3, i + .25, str(v), color='black')

plt.show()
```



6.2.18 DISTRIBUCIÓN DE JUGADORES SEGÚN VECES AMONESTADOS (TOP 10)

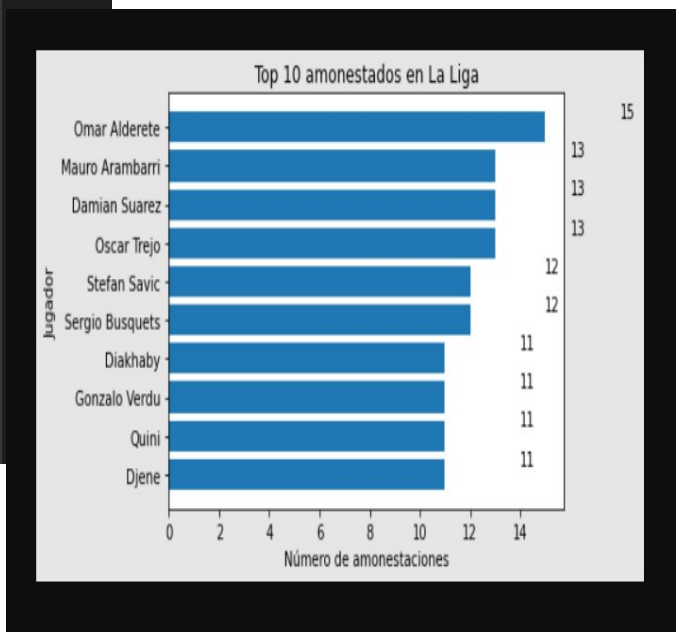
En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores más amonestados en la competición.

Las tarjetas amarillas son una medida disciplinaria que los árbitros pueden mostrar a los jugadores por una variedad de infracciones, como conducta antideportiva, simulación, protestas excesivas y reiteradas faltas.

Útil para comprender la disciplina y el comportamiento de los jugadores en el campo de juego. Puede mostrar si algunos jugadores son propensos a recibir tarjetas amarillas con mayor frecuencia que otros, lo que podría ser un indicador de un problema de disciplina o control emocional. Nos encontramos con :

- 15 amonestaciones : Omar Alderete (Defensa Valencia)
- 13 amonestaciones : Mauro Arambarri (Mediocampista Getafe CF), Damian Suarez (Defensa Getafe CF), Oscar Trejo (Mediocampista Rayo Vallecano)
- 12 amonestaciones : Stefan Savic (Defensa Atlético Madrid), Sergio Busquets (Mediocampista Barcelona)
- 11 amonestaciones : Diakhaby (Defensa Valencia), Gonzalo Verdú (Mediocampista Elche CF), Quini (Defensa Granada), Djene (Defensa Getafe CF)

```
# Seleccionar los datos relevantes y ordenarlos por número de tarjetas amarillas.
tarjetaamarilla = laliga[['Nombre', 'Tarjeta Amarilla']]
tarjetaamarilla = tarjetaamarilla.sort_values(by='Tarjeta Amarilla', ascending=False).head(10)

# ordenar
tarjetaamarilla = tarjetaamarilla.iloc[::-1]


# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(tarjetaamarilla['Nombre'], tarjetaamarilla['Tarjeta Amarilla'])

# Añadir etiquetas y título
ax.set_xlabel('Número de amonestaciones')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 amonestados en La Liga')

# Añadir el número exacto de pases en la gráfica
for i, v in enumerate(tarjetaamarilla['Tarjeta Amarilla']):
    ax.text(v + 3, i + .25, str(v), color='black')

plt.show()
```



6.2.19. DISTRIBUCIÓN DE JUGADORES SEGÚN VECES EXPULSADOS (TOP 10)

En esta sección debido a la gran cantidad de datos, se muestra solo los 10 jugadores más expulsados en la competición.

Es una sanción disciplinaria que se aplica cuando un jugador comete una falta grave o acumula dos tarjetas amarillas durante un mismo partido.

Muestra la disciplina y el comportamiento de los jugadores durante la temporada. Los jugadores que han sido expulsados con frecuencia pueden ser considerados un riesgo para el equipo, ya que su ausencia en un partido puede afectar el desempeño del equipo. Por otro lado, los jugadores que han sido expulsados pocas veces pueden ser considerados jugadores disciplinados y confiables.

Todos los jugadores mostrados tienen 2 expulsiones a lo largo de la temporada.

Jose Luis Gaya (Defensa Valencia), Hugo Mallo (Defensa Celta Vigo),

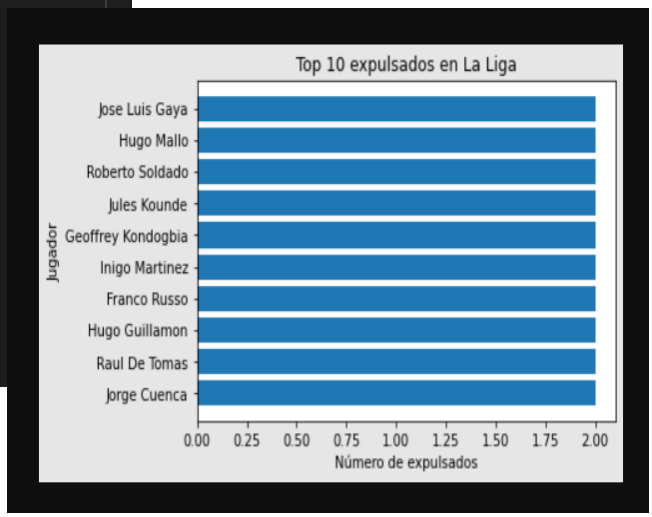
Roberto Soldado (Delantero Levante), Jules Kounde (Defensa Sevilla),

Kondogbia(Mediocampista Atlético Madrid), Iñigo Martínez (Defensa Athletic Bilbao),

Franco Russo (Defensa RCD Mallorca), Hugo Guillamón (Defensa Valencia),

Raúl De Tomás (Delantero Espanyol), Jorge Cuenca (Defensa Getafe CF)

```python
# Seleccionar los datos relevantes y ordenarlos por número de tarjetas rojas.
tarjetaroja = laliga[['Nombre', 'Tarjeta Roja']]
tarjetaroja = tarjetaroja.sort_values(by='Tarjeta Roja', ascending=False).head(10)

# Crear la gráfica de barras horizontal
fig, ax = plt.subplots()
ax.barh(tarjetaroja['Nombre'], tarjetaroja['Tarjeta Roja'])

# Añadir etiquetas y título
ax.set_xlabel('Número de expulsados')
ax.set_ylabel('Jugador')
ax.set_title('Top 10 expulsados en La Liga')

plt.show()
```



6.2.20. RELACIÓN ENTRE FALTAS Y AMONESTACIONES

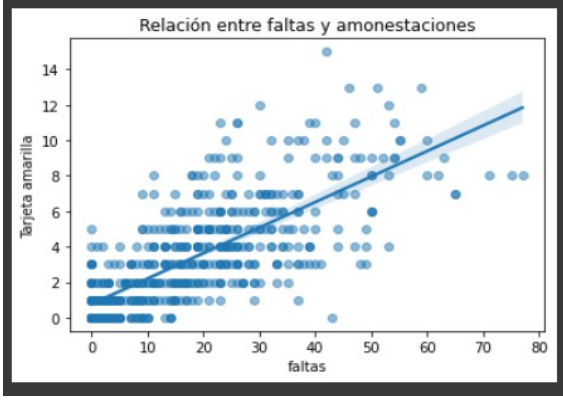
En esta gráfica de dispersión podemos ver cómo se relaciona el número de faltas y tarjetas amarillas durante la temporada.

Las tarjetas amarillas se muestran por infracciones cometidas por los jugadores en el campo, como faltas, juego brusco, conducta antideportiva, etc. Si un jugador comete varias faltas durante un partido, es más probable que reciba una tarjeta amarilla.

No todas las faltas son tarjeta amarilla, ya que los árbitros deciden cuándo mostrarla. Además, una falta grave puede resultar en una tarjeta roja directa, lo que llevaría a la expulsión del jugador.

Existe relación entre las faltas y las tarjetas amarillas, pero no siempre es directa y puede haber otros factores en juego, como la decisión del árbitro y la gravedad de la falta.

```
# Seleccionar los datos relevantes
relacion_faltas_amarillas = laliga[['Faltas', 'Tarjeta Amarilla']]

# Crear la gráfica de dispersión
sns.regplot(x='Faltas', y='Tarjeta Amarilla', data=relacion_faltas_amarillas, scatter_kws={'alpha':0.5})

# Añadir etiquetas y título
plt.xlabel('faltas')
plt.ylabel('Tarjeta amarilla')
plt.title('Relación entre faltas y amonestaciones')

plt.show()
```



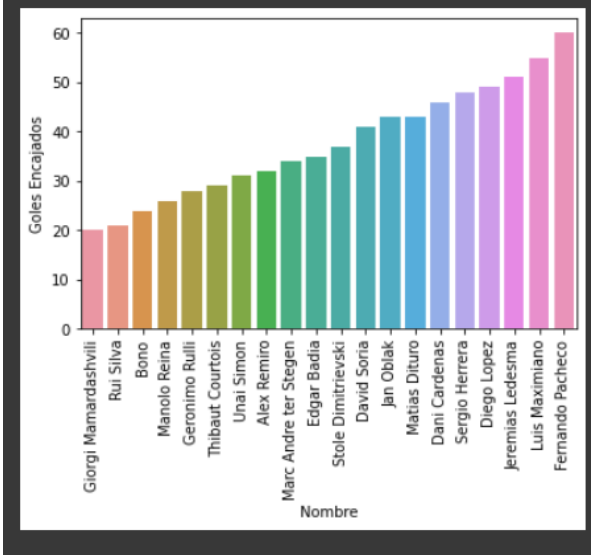
6.2.21. Distribución de Porteros según goles encajados (Top 20)

En esta sección debido a la gran cantidad de datos, se muestra solo los 20 porteros con más apariciones y menos goles encajados en la competición.

Esta información es importante para evaluar la eficacia de los porteros en su papel de proteger la portería del equipo. Además, puede ser útil para comparar la actuación de diferentes porteros y para determinar si un portero necesita ser reemplazado o si es necesario reforzar la defensa del equipo.

También puede ayudar a identificar a los porteros que han tenido un buen desempeño al encajar pocos goles y, por lo tanto, son valiosos para el equipo.

```
# Filtrar los porteros con más partidos jugados
porteros = laliga[laliga["Posicion"] == "Portero"]
top_porteros_apariciones = porteros.sort_values(by=["Apariciones"], ascending=False).head(20)

# Ordenar los porteros por goles encajados
top_porteros_menos_goles = top_porteros_apariciones.sort_values(by=["Goles Encajados"])

plt.xticks(rotation=90)

# Visualizar en una gráfica
sns.barplot(x="Nombre", y="Goles Encajados", data=top_porteros_menos_goles)

plt.show()
```



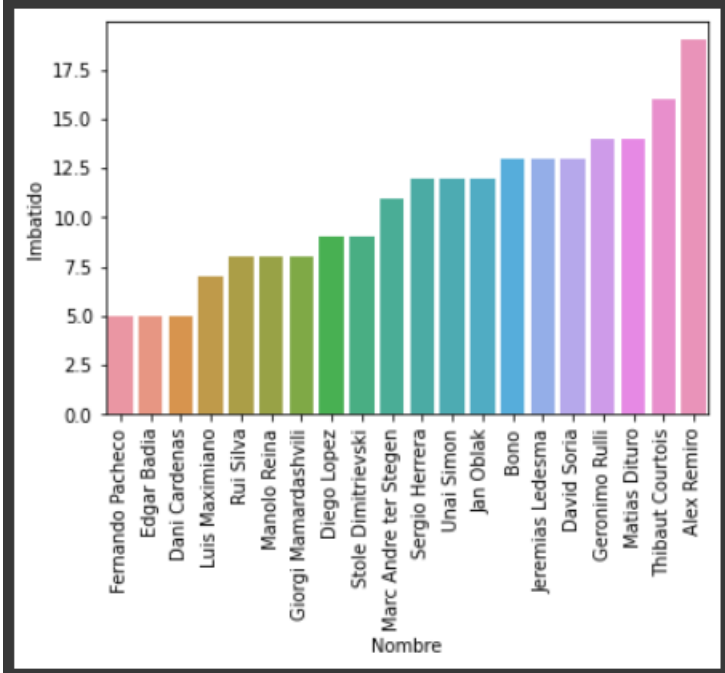
6.2.22. Distribución de Porteros según imbatibildad (Top 20)

En esta sección debido a la gran cantidad de datos, se muestra solo los 20 porteros con más apariciones y más veces imbatido en la competición.

Muestra la cantidad de partidos en los que un portero ha mantenido su portería a cero.Es importante para medir la eficacia y el rendimiento de un portero.

Un portero que mantiene su portería a cero en muchos partidos indica que es capaz de proteger su área de manera efectiva y contribuye a la victoria de su equipo. Además, la cantidad de partidos en los que un portero mantiene su portería a cero puede ser un factor importante en la decisión de un entrenador al elegir qué portero jugará en un partido determinado.

```
# Ordenar los porteros por imbatibilidad
top_porteros_imbatidos = top_porteros_apariciones.sort_values(by=["Imbatido"])

plt.xticks(rotation=90)

# Visualizar en una gráfica
sns.barplot(x="Nombre", y="Imbatido", data=top_porteros_imbatidos)

plt.show()
```



6.2.23. Distribución de jugadores según rendimiento (Top 10)

Esta sección que vamos a visualizar contiene gran importancia ya que relaciona todo el conjunto de datos y lo resume en una valoración del jugador a lo largo de la temporada para saber si tuvo buena actuación.

Debido a la gran cantidad de datos se muestra solo los 10 jugadores con mayor rendimiento pero con un número de apariciones igual o mayor a 19 partidos que se corresponden a la mitad de partidos de la competición. Encontramos con :

- 7,73 : Karim Benzema (Delantero Real Madrid)
- 7,53 : Ousmane Dembele (Delantero Barcelona)
- 7,44 : Toni Kroos (Mediocampista Real Madrid)
- 7,41 : Dani Parejo (Mediocampista Villarreal)
- 7,35 : Vinicius Junior (Delantero Real Madrid)
- 7,34 : Casemiro (Mediocampista Real Madrid), Iago Aspas (Delantero Celta Vigo)
- 7,32 : Iker Muniain (Mediocampista Athletic Bilbao)
- 7,30 : Jordi Alba (Defensa Barcelona)
- 7,26 : Memphis Depay (Delantero Barcelona)

```
topaapariciones = laliga[laliga['Apariciones'] >= 19]

# Ordenar los jugadores por rendimiento en orden descendente
top_apariciones = topaapariciones.sort_values(by='Rendimiento', ascending=False)

# Seleccionar los 10 jugadores con mayor rendimiento
aparicion_top10 = topaapariciones.head(10)

#ordenar
aparicion_top10 = aparicion_top10.iloc[::-1]

# Crear un gráfico de barras para visualizar el rendimiento de los 10 mejores jugadores
plt.barh(aparicion_top10['Nombre'], aparicion_top10['Rendimiento'])

for i, rendimiento in enumerate(aparicion_top10['Rendimiento']):
    plt.text(rendimiento + 0.1, i, str(round(rendimiento, 2)))

# Añadir títulos y etiquetas de los ejes
plt.title('Los 10 mejores jugadores según su rendimiento')
plt.xlabel('Jugador')
plt.ylabel('Rendimiento (19 o más apariciones)')

# Mostrar el gráfico
plt.show()
```

6.3. VISUALIZACIÓN DE CORRELACIONES

En estadística, la correlación se refiere a una medida de la relación entre dos variables. Por ejemplo anteriormente hemos visualizado :

- Relación entre partidos jugados y minutos totales jugados. (Normalmente es más probable que los jugadores que juegan más minutos hayan aparecido en más partidos, aunque no siempre es cierto)
- Relación entre goles y tiros totales.(a medida que aumentan los tiros a portería, también aumentan los goles)
- Relación entre número total de pases y número de pases claves(a medida que aumenta el número total de pases, también tiende a aumentar el número de pases claves)
- Relación entre pases claves y número de asistencias(a medida que aumenta el número de pases claves, es más probable que un jugador dé una asistencia)
- Relación entre faltas y amonestaciones(jugador comete varias faltas durante un partido, es más probable que reciba una tarjeta amarilla)

Antes de nada podemos eliminar la columna Id ya que no nos mostrará información relevante.

```
# eliminar columna id

laliga = laliga.drop(columns=['Id'])
```

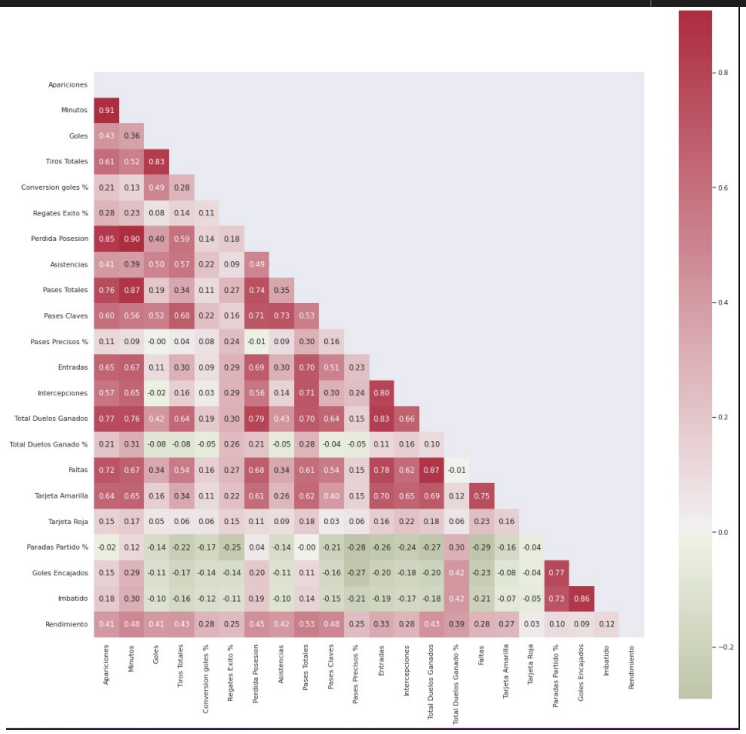
A continuación pasamos a ver la La matriz de correlaciones que es una tabla que muestra la relación entre todas las variables en un conjunto de datos.

```
# matriz de correlaciones

corr_matrix = laliga.corr()
corr_matrix
```

	Apariciones	Minutos	Goles	Tiros Totales	Conversion goles %	Regates Exito %	Perdida Posesion	Asistencias	Pases Totales	Pases Claves	...	Intercepciones	Total Duelos Ganados	Total Duelos Ganado %	Faltas	Tarjeta Amarilla	Tarjeta Roja	Paradas Partido %
Apariciones	1.000000	0.908153	0.427626	0.607284	0.207698	0.280043	0.846375	0.411168	0.759439	0.596450	...	0.572421	0.768393	0.206956	0.720405	0.641637	0.147306	-0.023921
Minutos	0.908153	1.000000	0.363276	0.523476	0.131915	0.225647	0.898836	0.385358	0.865063	0.557433	...	0.645199	0.759219	0.305811	0.666939	0.653474	0.173870	0.120878
Goles	0.427626	0.363276	1.000000	0.830493	0.494535	0.076852	0.396708	0.500353	0.194963	0.524962	...	-0.018613	0.419091	-0.076842	0.336501	0.161742	0.051105	-0.142932
Tiros Totales	0.607284	0.523476	0.830493	1.000000	0.280354	0.139650	0.586517	0.574394	0.343140	0.679792	...	0.160558	0.635504	-0.082868	0.542723	0.335753	0.056076	-0.220606
Conversion goles %	0.207698	0.131915	0.494535	0.280354	1.000000	0.111870	0.139916	0.218921	0.107151	0.223636	...	0.030743	0.188319	-0.045416	0.158798	0.110270	0.057966	-0.172180
Regates Exito %	0.280043	0.225647	0.076852	0.139650	0.111870	1.000000	0.180199	0.094923	0.268345	0.157518	...	0.286874	0.296213	0.255556	0.266144	0.222420	0.145218	-0.253991
Perdida Posesion	0.846375	0.898836	0.396708	0.586517	0.139916	0.180199	1.000000	0.492157	0.739255	0.711190	...	0.561840	0.794541	0.209782	0.681674	0.610430	0.111802	0.038536
Asistencias	0.411168	0.385358	0.500353	0.574394	0.218921	0.094923	0.492157	1.000000	0.346354	0.725290	...	0.135444	0.426934	-0.047361	0.341288	0.263085	0.090854	-0.141047
Pases Totales	0.759439	0.865063	0.194963	0.343140	0.107151	0.268345	0.739255	0.346354	1.000000	0.526151	...	0.710200	0.697385	0.278173	0.606152	0.623276	0.184225	-0.004059
Pases Claves	0.596450	0.557433	0.524962	0.679792	0.223636	0.157518	0.711190	0.725290	0.526151	1.000000	...	0.295741	0.640382	-0.035065	0.539810	0.398177	0.031026	-0.207894
Pases Precisos %	0.114609	0.090017	-0.004487	0.035915	0.078768	0.235519	-0.009473	0.093888	0.298473	0.158071	...	0.237309	0.152268	-0.047083	0.151570	0.145980	0.058792	-0.284769
Entradas	0.646697	0.665902	0.106552	0.303626	0.090012	0.289756	0.689202	0.303587	0.702991	0.505135	...	0.795016	0.827816	0.112860	0.775824	0.695352	0.164175	-0.262926
Intercepciones	0.572421	0.645199	-0.018613	0.160558	0.030743	0.286874	0.561840	0.135444	0.710200	0.295741	...	1.000000	0.664173	0.159560	0.617387	0.648620	0.219546	-0.235065
Total Duelos Ganados	0.768393	0.759219	0.419091	0.635504	0.188319	0.296213	0.794541	0.426934	0.697385	0.640382	...	0.664173	1.000000	0.103697	0.866800	0.693073	0.178430	-0.271448
Total Duelos Ganado %	0.206956	0.305811	-0.076842	-0.082868	-0.045416	0.255556	0.209782	-0.047361	0.278173	-0.035065	...	0.159560	0.103697	1.000000	-0.005134	0.115942	0.062225	0.296124
Faltas	0.720405	0.666939	0.336501	0.542723	0.158798	0.266144	0.681674	0.341288	0.606152	0.539810	...	0.617387	0.866800	-0.005134	1.000000	0.754602	0.230258	-0.290843

Para mostrar esas correlaciones de una manera más visual mostramos un mapa de calor donde aparece cada campo y el porcentaje de relación con otros campos de nuestro conjunto de datos.

```
#crear una mascara
my_mask = np.triu(np.ones_like(corr_matrix,dtype=bool))

#crear una paleta personalizada
my_cmap = sns.diverging_palette(100, 7, s=75, l=40, n=5, center="light", as_cmap=True)

#creamos mapas de calor

sns.set(rc={"figure.figsize": (20, 20)})
sns.heatmap(corr_matrix, mask=my_mask, cmap=my_cmap, center=0, fmt='.2f', annot=True, square=True)
plt.show()
```



Otras relaciones que podemos ver a parte de las nombradas :

Por ejemplo los minutos están relacionados con la mayoría de los campos ya que cuanto más minutos juegues más probabilidad hay de realizar la acción

(Tiros totales 0,52 – Perdida posesión 0,90 – Pases totales 0,87 – Total duelos ganados 0,76 …)

Otro ejemplo la relación de pases totales con perdida de posesión 0.74 ya que cuanto más pases hagas más probabilidad hay de que salga un pase mal y dar la posesión al rival

7. PREPARACIÓN DE LOS DATOS PARA LOS ALGORITMOS DE MACHINE LEARNING

En la preparación de datos para algoritmos de Machine Learning se realizan una serie de tareas para asegurarse de que los datos estén listos para ser utilizados en los modelos de aprendizaje automático. Estas tareas son :

- Selección de campos a utilizar.
- cambiar la variable categórica de posición a numérica.
- Mostrar de nuevo como quedarón las correlaciones.

La preparación de los datos es importante para obtener un buen rendimiento del modelo de aprendizaje automático.

7.1. SELECCIONAR CAMPOS

Tras todo el estudio realizado y lo comentado anteriormente llegamos a la conclusión de que tenemos campos que podemos eliminar a la hora de entrenar nuestro modelo como:

- Id (identificador borrada anteriormente)
- Equipo (Club al que pertenece el jugador)
- Nombre (Nombre del jugador)
- Apariciones (Partidos en los que ha participado ya que minutos puede tener más relevancia)
- Tarjeta Amarilla (Veces amonestado tomamos como referencia las faltas)
- Tarjeta Roja (Veces expulsado tomamos como referencia las faltas)

```
# Eliminar las columnas que no son necesarias para la predicción del rendimiento
laliga.drop(['Equipo', 'Nombre', 'Apariciones', 'Tarjeta Amarilla', 'Tarjeta Roja'], axis=1, inplace=True)

# Imprimir las primeras filas del dataset para verificar los cambios
print(laliga.head())
```

7.2. VARIABLES CATEGÓRICAS

Se transforman las variables categóricas en variables numéricas para que los algoritmos de aprendizaje automático puedan trabajar con ellas.

En nuestro caso transformaremos el campo posición ya que es importante utilizarlo en el modelo. Hay que tener en cuenta la posición del jugador ya que un defensa no meterá el mismo número de goles que un delantero. El nuevo valor será el siguiente :

- Portero – 1 Defensa – 2 Mediocampista – 3 Delantero – 4

```
# Convertir a valores númericos Posición

laliga['Posicion'].replace(['Portero', 'Defensa', 'Mediocampista', 'Delantero'],[1., 2., 3., 4.], inplace=True)
```

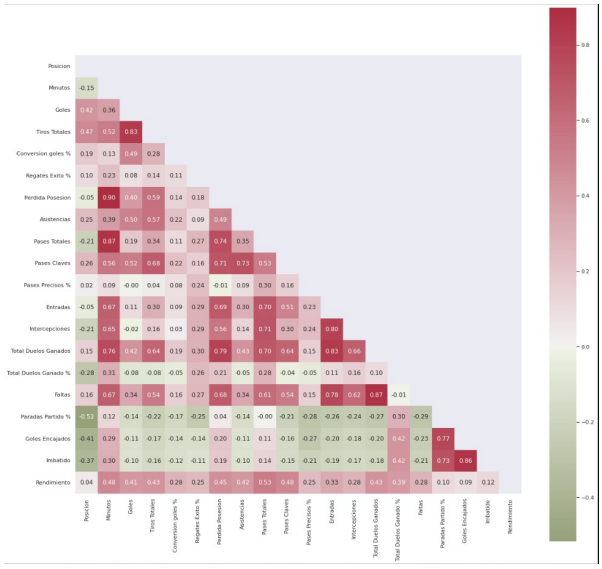
7.3. NUEVAS CORRELACIONES

Tras la limpieza de datos realizadas podemos mostrar una nueva tabla de correlaciones y el mapa de calor .

```
# matriz de correlaciones

corr_matrix1 = laliga.corr()
corr_matrix1
```

	Posicion	Minutos	Goles	Tiros Totales	Conversion goles %	Regates Exito %	Perdida Posesion	Asistencias	Pases Totales	Pases Claves	Pases Precisos %	Entradas	Intercepciones	Total Duelos Ganados	Total Duelos Ganado %
Posicion	1.000000	-0.145379	0.419256	0.472898	0.187178	0.098770	-0.048488	0.245635	-0.209649	0.257459	0.020246	-0.054948	-0.205656	0.151363	-0.281774
Minutos	-0.145379	1.000000	0.363276	0.523476	0.131915	0.225647	0.898836	0.385358	0.865063	0.557433	0.090017	0.665902	0.645199	0.759219	0.305811
Goles	0.419256	0.363276	1.000000	0.830493	0.494535	0.076852	0.396708	0.500353	0.194963	0.524962	-0.004487	0.106552	-0.018613	0.419091	-0.076842
Tiros Totales	0.472898	0.523476	0.830493	1.000000	0.280354	0.139650	0.586517	0.574394	0.343140	0.679792	0.035915	0.303626	0.160558	0.635504	-0.082868
Conversion goles %	0.187178	0.131915	0.494535	0.280354	1.000000	0.111870	0.139916	0.218921	0.107151	0.223636	0.078768	0.090012	0.030743	0.188319	-0.045416
Regates Exito %	0.098770	0.225647	0.076852	0.139650	0.111870	1.000000	0.180199	0.094923	0.268345	0.157518	0.235519	0.289756	0.286874	0.296213	0.255556
Perdida Posesion	-0.048488	0.898836	0.396708	0.586517	0.139916	0.180199	1.000000	0.492157	0.739255	0.711190	-0.009473	0.689202	0.561840	0.794541	0.209782
Asistencias	0.245635	0.385358	0.500353	0.574394	0.218921	0.094923	0.492157	1.000000	0.346354	0.725290	0.093888	0.303587	0.135444	0.426934	-0.047361
Pases Totales	-0.209649	0.865063	0.194963	0.343140	0.107151	0.268345	0.739255	0.346354	1.000000	0.526151	0.298473	0.702991	0.710200	0.697385	0.278173
Pases Claves	0.257459	0.557433	0.524962	0.679792	0.223636	0.157518	0.711190	0.725290	0.526151	1.000000	0.158071	0.505135	0.295741	0.640382	-0.035065
Pases Precisos %	0.020246	0.090017	-0.004487	0.035915	0.078768	0.235519	-0.009473	0.093888	0.298473	0.158071	1.000000	0.226637	0.237309	0.152268	-0.047083
Entradas	-0.054948	0.665902	0.106552	0.303626	0.090012	0.289756	0.689202	0.303587	0.702991	0.505135	0.226637	1.000000	0.795016	0.827816	0.112860

```
#crear una mascara
my_mask = np.triu(np.ones_like(corr_matrix1,dtype=bool))

#crear una paleta personalizada
my_cmap = sns.diverging_palette(100, 7, s=75, l=40, n=5, center="light", as_cmap=True)

#creamos mapas de calor

sns.set(rc={"figure.figsize": (20, 20)})
sns.heatmap(corr_matrix1, mask=my_mask, cmap=my_cmap, center=0, fmt='.2f', annot=True, square=True)
plt.show()
```



8. ENTRENAMIENTO DEL MODELO Y COMPROBACIÓN DEL RENDIMIENTO

Tras el estudio,visualización y preparación de nuestros datos ahora se dividirán los datos en conjuntos de entrenamiento y prueba. Luego utilizaremos nuestros datos de entrenamiento y prueba para realizar modelos de predicción con algoritmos de aprendizaje automático seleccionados y se evaluará el rendimiento que han tenido cada uno de ellos seleccionando el más adecuado.

8.1. ELEGIR CAMPOS DEL MODELO

Elegimos los campos que utilizaremos en los diferentes modelos (Posición,minutos,goles,conversión de goles %, regates éxito%, perdida posesión, asistencias, pases claves, pases precisos %, entradas, intercepciones, total duelos ganados %, faltas, paradas partido %, goles encajados, rendimiento) y lo guardaremos en una variable llamada datosmodelo.

También dividimos el conjunto de datos entrenamiento y prueba, donde utilizamos el 20 % de datos como prueba y para evaluar el modelo y el 80 % restante se utiliza como entrenamiento.

Ademas establecemos una semilla con random_state= 42 es un parámetro utilizado para la división de los datos, cada vez que se ejecute el código se obtiene la misma división de los datos y es útil para asegurarse de que los resultados obtienen los mismos resultados cada vez que se ejecuta el código utilizando mismos datos de entrada y parámetros y también obtenga resultados similares)

```
from sklearn.model_selection import train_test_split

datosmodelo = laliga[['Posicion', 'Minutos', 'Goles', 'Conversion goles %', 'Regates Exito %', 'Perdida Posesion',
        'Asistencias', 'Pases Claves', 'Pases Precisos %', 'Entradas', 'Intercepciones',
        'Total Duelos Ganado %', 'Faltas', 'Paradas Partido %', 'Goles Encajados', 'Rendimiento']]

# Divide los datos en conjunto de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(datosmodelo.drop('Rendimiento', axis=1), datosmodelo['Rendimiento'],
                                                    test_size=0.2, random_state=42)
```

Podemos mostrar los datos utilizados tanto en entrenamiento como prueba

```
# mostrar x_train

print(X_train)
```

```
# mostrar y_train

print(y_train)
```

```
# mostrar X_test

print(X_test)
```

```
# mostrar y_test

print(y_test)
```

8.2. ENTRENAR MODELO DE REGRESIÓN LINEAL

El primer modelo que vamos a probar es el de regresión lineal, es utilizado en el aprendizaje automático y estadísticas para predecir una variable numérica continua a partir de una o varias variables independientes, su objetivo es encontrar una linea recta que ajuste los datos para poder hacer predicciones precisas.

A continuación implementamos el modelo de regresión lineal utilizando la biblioteca Scikit-learn (sklearn) para entrenar y ajustar nuestro modelo, lo introducimos en una variable llamada reg y llamamos a esa variable para entrenar con fit.

Una vez ajustado el modelo ha sido ajustado se puede utilizar para hacer predicciones sobre nuevos datos.

```
from sklearn.linear_model import LinearRegression

# Entrena un modelo de regresión lineal
reg = LinearRegression()
reg.fit(X_train, y_train)
```

8.2.1 VALOR DE PREDICCIONES

A continuación vamos a hacer unas predicciones guardándolas en la variable prediccioneslineal

```
prediccioneslineal = reg.predict(X_test)
prediccioneslineal
```

```
array([6.60822555, 6.18869729, 6.87378318, 6.87461189, 6.59747538,
       6.74941264, 6.21993535, 6.77779018, 6.92457472, 6.6386055 ,
       6.64854835, 6.89856107, 6.80941927, 6.73357865, 6.85089346,
       6.99942024, 6.78743562, 6.66611422, 6.87789578, 6.83809307,
       6.90972573, 6.78195991, 6.77514311, 7.06360868, 6.49494863,
       6.64022008, 6.31168299, 6.7548297 , 6.61315262, 6.83457739,
       6.90762361, 6.93285862, 7.11185911, 7.06097123, 6.47931533,
       6.92096392, 6.94813943, 6.68645385, 6.89532167, 6.6747393 ,
       6.73537761, 6.71883597, 6.81005174, 6.71677797, 6.51769604,
       6.73637125, 6.74104761, 6.66671324, 7.13825981, 6.5841736 ,
       6.72373395, 6.68004146, 7.43804881, 6.63064507, 6.81743979,
       6.6550034 , 6.56122558, 6.63204462, 6.93229983, 6.7704358 ,
       6.72627412, 6.8202884 , 6.71191909, 7.02127139, 6.81246141,
       6.6792309 , 6.90009082, 6.99214239, 6.76306844, 6.85493262,
       6.46684838, 6.81985412, 6.91047257, 6.71841871, 6.90831326,
       6.68982132, 6.92324238, 6.88620498, 6.72924384, 7.0810713,
       7.15990959, 6.8880546 , 7.01293808, 6.60051375, 6.79368966,
       6.94897517, 6.97628785, 6.7175893 , 6.65621197, 6.42277453,
       6.41791604, 6.65770159, 6.69180682, 6.70774884, 6.81224612,
       6.94875279, 6.812004  , 6.62549988, 6.67522501, 6.58702147,
       6.94743981, 6.8354379 , 6.3893283 , 7.4839232 , 6.70852949,
       6.68986211, 6.66955129, 6.73349938, 6.52325573, 6.64298925,
       6.84528221, 6.9056172 , 7.10263401, 6.8598332 , 6.72435098,
       7.00946875, 6.74383781, 6.68837947, 6.85464491])
```

8.2.2. PORCENTAJE DE ERROR Y COEFICIENTE DE DETERMINACIÓN

Ahora evaluamos la calidad del modelo de regresión lineal y se determina su capacidad para predecir valores de rendimiento.

El porcentaje de error y el coeficiente de determinación son dos medidas de evaluación comunes utilizadas en la regresión lineal para evaluar el rendimiento del modelo. El porcentaje de error indica qué tan cerca está una predicción de su valor real. Cuanto menor sea el porcentaje de error, más precisa será la predicción

El coeficiente de determinación mide como se ajustan los datos a la línea de regresión. Su valor varía entre 0 y 1. Cercano a 1 indica que el modelo ajusta muy bien los datos, mientras que un valor cercano a 0 indica que el modelo no ajusta bien los datos.

Nuestro modelo tiene :

- Error porcentual : 2.193942
- coeficiente de determinación: 0.6104293697862527

```
from sklearn.metrics import mean_squared_error, r2_score

error = np.sqrt(mean_squared_error(y_test, prediccioneslineal))
error_porcentual = error / np.mean(y_test) * 100

print("Error porcentual : %f" % error_porcentual)

# coeficiente de determinacion 1 es la prediccion perfecta
print("coeficiente de determinacion: ", r2_score(y_test, prediccioneslineal))
```

```
Error porcentual : 2.193942
coeficiente de determinacion:  0.6104293697862527
```

8.2.3. AGREGAR NUEVO JUGADOR Y PREDECIR RESULTADO

Para terminar nuestro modelo de regresión lineal haremos una predicción de prueba con nuevos datos introducidos :

```
nuevo_jugador = pd.DataFrame(np.array([[4, 2300, 24, 20.05, 71.05, 286, 9, 22, 77.20, 11, 7, 52.01, 21, 0, 0]]),
                             columns=['Posicion', 'Minutos', 'Goles', 'Conversion goles %', 'Regates Exito %',
                                      'Perdida Posesion', 'Asistencias', 'Pases Claves', 'Pases Precisos %',
                                      'Entradas', 'Intercepciones', 'Total Duelos Ganado %', 'Faltas',
                                      'Paradas Partido %', 'Goles Encajados'])
nuevo_jugador
```

```
reg.predict(nuevo_jugador)
```

```
array([7.33583516])
```

8.3. MODELO DE ÁRBOL DE DECISIÓN

El segundo modelo que probamos es el de árbol de decisión , este modelo utiliza un árbol para representar y clasificar los datos de entrada en función de una serie de decisiones basadas en características especificas de los datos.

Cada nodo del árbol representa una pregunta sobre una característica de los datos de entrada, cada respuesta conduce por un camino diferente del árbol y a otra pregunta.

Cada camino del árbol representa una predicción distinta basada en la respuesta.

A continuación implementamos el modelo de árbol de decisión utilizando la biblioteca Scikit-learn (sklearn) para entrenar y ajustar nuestro modelo, lo introducimos en una variable llamada modelotree y llamamos a esa variable para entrenar con fit.

Una vez ajustado el modelo ha sido ajustado se puede utilizar para hacer predicciones sobre nuevos datos.

```
from sklearn.tree import DecisionTreeRegressor

modelotree = DecisionTreeRegressor()

# entrenamiento del modelo
modelotree.fit(X_train, y_train)
```

8.3.1. VALOR DE PREDICCIONES

A continuación vamos a hacer unas predicciones guardándolas en la variable prediccionestree

```
prediccionestree = modelotree.predict(X_test)
prediccionestree
```

```
array([6.64, 6.5 , 6.62, 7.02, 6.63, 6.77, 6.5 , 6.84, 6.64, 6.63, 6.75,
       6.99, 6.61, 6.65, 6.98, 7.34, 7.04, 6.9 , 6.89, 6.99, 6.71, 6.91,
       6.85, 6.91, 6.6 , 6.69, 6.2 , 6.89, 6.59, 6.37, 7.34, 6.71, 7.04,
       7.09, 6.34, 6.92, 7.  , 6.65, 6.54, 6.65, 6.95, 6.84, 6.9 , 6.75,
       6.6 , 6.88, 7.14, 6.58, 7.26, 6.64, 6.54, 6.39, 7.26, 6.57, 6.7 ,
       6.7 , 6.6 , 6.68, 6.9 , 6.71, 6.61, 6.88, 6.8 , 6.86, 6.69, 6.69,
       6.9 , 7.02, 6.78, 6.92, 6.3 , 6.73, 7.  , 6.62, 6.93, 6.72, 7.02,
       6.54, 6.66, 6.88, 6.85, 7.08, 7.7 , 6.59, 6.6 , 7.11, 7.15, 6.93,
       6.63, 6.7 , 6.58, 6.56, 6.7 , 7.14, 6.6 , 6.37, 7.09, 6.39, 6.68,
       6.37, 6.81, 6.73, 6.5 , 7.09, 7.13, 6.62, 6.75, 6.85, 6.85, 6.61,
       6.92, 6.84, 6.99, 7.07, 6.6 , 6.9 , 6.88, 6.5 , 6.5 ])
```

8.3.2. AGREGAR NUEVO JUGADOR Y PREDECIR RESULTADO

El jugador será el introducido anteriormente :

```
nuevo_jugador = pd.DataFrame(np.array([[4, 2300, 24, 20.05, 71.05, 286, 9, 22, 77.20, 11, 7, 52.01, 21, 0, 0]]),
                             columns=['Posicion', 'Minutos', 'Goles', 'Conversion goles %', 'Regates Exito %',
                                      'Perdida Posesion', 'Asistencias', 'Pases Claves', 'Pases Precisos %',
                                      'Entradas', 'Intercepciones', 'Total Duelos Ganado %', 'Faltas',
                                      'Paradas Partido %', 'Goles Encajados'])
nuevo_jugador
```

```
modelotree.predict(nuevo_jugador)
```

```
array([7.09])
```

8.3.3. PORCENTAJE DE ERROR Y COEFICIENTE DE DETERMINACIÓN

Nuestro modelo tiene :

- error cuadrático medio: 0.23374193915009575
- coeficiente de determinación: 0.03406158111736135

```
# error
print("error cuadratico medio: ", mean_squared_error(y_test, prediccionestree, squared=False))

# coeficiente de determinacion 1 es la prediccion perfecta
print("coeficiente de determinacion: ", r2_score(y_test, prediccionestree))
```

```
error cuadratico medio:  0.23374193915009575
coeficiente de determinacion:  0.03406158111736135
```

8.4. MODELO DE RANDOM FOREST

El tercer y ultimo modelo utilizado es random forest, se basa en la combinación de varios árboles de decisión, cada uno de los cuales se entrena con una muestra aleatoria de los datos de entrada y un subconjunto aleatorio de las características.

A continuación implementamos el modelo de árbol de decisión utilizando la biblioteca Scikit-learn (sklearn) para entrenar y ajustar nuestro modelo, lo introducimos en una variable llamada modeloforest y llamamos a esa variable para entrenar con fit.

Una vez ajustado el modelo ha sido ajustado se puede utilizar para hacer predicciones sobre nuevos datos. Añadiremos las predicciones en este código para no abrir otro apartado.

```
from sklearn.ensemble import RandomForestRegressor

modeloforest = RandomForestRegressor()

# entrenamiento del modelo
modeloforest.fit(X_train, y_train)

#prediccion
prediccionesforest = modeloforest.predict(X_test)
prediccionesforest
```

```
array([6.5688, 6.3577, 6.8144, 6.8545, 6.6431, 6.9267, 6.4874, 6.7948,
       6.772 , 6.6249, 6.6707, 7.0573, 6.669 , 6.7181, 6.8779, 6.9747,
       6.7411, 6.8607, 6.8286, 6.8475, 6.7754, 6.8828, 6.7997, 7.0871,
       6.4278, 6.688 , 6.2867, 6.9142, 6.6047, 6.8571, 6.889 , 6.8531,
       6.9452, 7.0892, 6.3462, 6.8925, 6.9551, 6.6896, 6.8689, 6.7046,
       6.8676, 6.7291, 6.9375, 6.6476, 6.5437, 6.771 , 6.8248, 6.773 ,
       7.1636, 6.5856, 6.7698, 6.7069, 7.1731, 6.6433, 6.7799, 6.6628,
       6.3688, 6.6404, 6.9247, 6.7558, 6.6597, 6.8187, 6.7704, 6.893 ,
       6.7279, 6.7145, 6.8988, 7.009 , 6.6721, 6.8679, 6.3718, 6.9074,
       6.9991, 6.7183, 7.0244, 6.624 , 7.0428, 6.7622, 6.7348, 6.9705,
       6.7694, 6.9054, 6.8834, 6.5265, 6.7773, 7.002 , 7.0473, 6.8475,
       6.6685, 6.6539, 6.5143, 6.6864, 6.6874, 6.8003, 6.7694, 6.789 ,
       7.0172, 6.5571, 6.6988, 6.6029, 6.9202, 6.8254, 6.4928, 7.2087,
       6.7293, 6.7947, 6.6301, 6.7744, 6.556 , 6.691 , 6.9119, 6.9028,
       6.8227, 7.0356, 6.7208, 6.9371, 6.8568, 6.8078, 6.9471])
```

8.4.1. PORCENTAJE DE ERROR Y COEFICIENTE DE DETERMINACIÓN

Nuestro modelo tiene :

- error cuadrático medio: 0.14212513394005613
- coeficiente de determinación: 0.6428769117877677

```
# error
print("error cuadratico medio: ", mean_squared_error(y_test, prediccionesforest, squared=False))


# coeficiente de determinacion 1 es la prediccion perfecta
print("coeficiente de determinacion: ", r2_score(y_test, prediccionesforest))
```

```
error cuadratico medio:  0.14212513394005613
coeficiente de determinacion:  0.6428769117877677
```

8.4.2. AGREGAR NUEVO JUGADOR Y PREDECIR EL RESULTADO

```
nuevo_jugador = pd.DataFrame(np.array([[4, 2300, 24, 20.05, 71.05, 286, 9, 22, 77.20, 11, 7, 52.01, 21, 0, 0]]),
                             columns=['Posicion', 'Minutos', 'Goles', 'Conversion goles %', 'Regates Exito %',
                                      'Perdida Posesion', 'Asistencias', 'Pases Claves', 'Pases Precisos %',
                                      'Entradas', 'Intercepciones', 'Total Duelos Ganado %', 'Faltas',
                                      'Paradas Partido %', 'Goles Encajados'])
nuevo_jugador
```

```
modeloforest.predict(nuevo_jugador)
```

```
array([7.0447])
```

9. SÍNTESIS DE VOZ

La síntesis de voz es un proceso mediante el cual se utiliza la tecnología para crear una representación sonora de un texto escrito, de manera que se pueda escuchar una versión hablada del mismo.

En este apartado importaremos la librería gTTS para que nuestra predicción pueda ser escuchada por parte del usuario. Utilizamos el modelo de random forest que es el más eficiente de los tres comprobados.

```
!pip install gTTS
```

```
from gtts import gTTS

texto = "El rendimiento de este jugador es: " + str(modeloforest.predict(nuevo_jugador))
tts = gTTS(text=texto, lang='es')
```

```
voz = gTTS(texto, lang='es', slow=False)
```

```
voz.save("prueba.mp3")

from IPython.display import Audio
Audio("prueba.mp3")
```

9. IMPLEMENTAR EN STREAMLIT

A continuación pasamos a crear una aplicación web interactiva que contenga varios modelos de predicción y cada uno realiza una función diferente, tendremos:

- Un modelo para predecir el rendimiento de un jugador.
- Un modelo para predecir los goles de un jugador.
- Un modelo para predecir goles encajados por un portero.

Cada uno de estos modelos tendrá una ventana definida en la página que podemos elegir a través de un menú

Para ello utilizamos streamlit una herramienta de código abierto que permite visualizar datos y crear aplicaciones web interactivas fácilmente.

Tendremos que crear un archivo app.py que contiene la librerías utilizadas, el código de nuestra pagina y el formato que le queremos dar, a parte de enlazar nuestros modelos.pkl que contienen el código python para realizar las predicciones nombradas anteriormente en el sitio adecuado.

Luego todo el contenido de nuestra página será subido a nuestro github:

app.py (contiene el código de nuestra página)

modelos.pkl (Contiene el codigo de nuestros modelos de predicción)

imágenes, requeriments.txt (Contiene las librerías que hemos utilizado)

Tras subir todo correctamente conectamos github con nuestro streamlit para que podamos tener acceso a la página a través de Internet.

10. CONCLUSIÓN

Durante este TFM se han utilizado varios modelos de Machine Learning para predecir el rendimiento de los jugadores de La Liga. Se han utilizado modelos de regresión lineal, árboles de decisión y random forest.

El modelo de regresión lineal ha obtenido un error porcentual del 2.19% y un coeficiente de determinación de 0.61. Esto indica que el modelo puede hacer predicciones con una precisión razonable, pero aún hay margen de mejora.

Por otro lado, el árbol de decisión ha obtenido un error cuadrático medio del 0.23 y un coeficiente de determinación de solo 0.034, lo que indica que este modelo no es adecuado para hacer predicciones precisas en este conjunto de datos.

Finalmente, el modelo de random forest ha obtenido el menor error cuadrático medio de 0.14 y el coeficiente de determinación más alto de 0.64. Esto indica que el modelo de random forest es el más adecuado para hacer predicciones precisas en este conjunto de datos.

En conclusión, el modelo de random forest es el más adecuado para hacer predicciones precisas en este conjunto de datos de La Liga. Sin embargo, siempre hay margen de mejora y se pueden explorar otros modelos y técnicas de procesamiento de datos para obtener mejores resultados.

- 1º Random forest error cuadrático medio 0.14, coeficiente de determinación más alto de 0.64
- 2º Regresión lineal error cuadrático medio 2.19%, coeficiente de determinación de 0.61
- 3º árbol de decisión error cuadrático medio 0.23 y coeficiente de determinación de solo 0.034 (No adecuado para este conjunto de datos)

11. ENLACES

Por ultimo mostraremos diferente enlaces relacionados con el TFM

- Enlace github que contiene todo los archivos necesarios para la realización :
  github_scoutliga

- Enlaces diferentes cuadernos colab :
  - tfm_scoutliga_JaimeSalado
  - prediccion_rendimiento.pkl
  - prediccion_goles.pkl
  - prediccion_goles_encajados.pkl

- Enlace a la pagina streamlit :
  - Descubriendo el talento a través del arte del análisis deportivo(ScoutLiga)