PRÁCTICA GRUPAL MINERÍA DE DATOS

Estudio sobre el rendimiento de los pilotos en la Fórmula1

Gonzalo Román Sánchez 100473601@alumnos.uc3m.es
Héctor Benito Martínez 100473670@alumnos.uc3m.es
Miguel Mancebo Arnanz 100473607@alumnos.uc3m.es

17/12/2024

MINERÍA DE DATOS

UNIVERSIDAD CARLOS III DE MADRID

Índice del trabajo:

1. Introducción

La estadística y el análisis de datos aplicado en el deporte hoy en día están en auge y a pesar de que en muchos deportes es un ámbito que se sigue rechazando por miedo a romper con los métodos tradicionales, muchos otros lo adoptan para el análisis de estrategias, incorporación de nuevos perfiles o prevención de lesiones.

Entre las disciplinas que son menos reacias a incorporar este tipo de herramientas han destacado históricamente el baloncesto y la Fórmula 1. En el baloncesto resultan esenciales las estadísticas para la valoración de jugadores, partidos y resultados globales; y aspectos como la inteligencia artificial y la estadística espacial están a la orden del día. Por otro lado, en la Fórmula 1 se utilizan modelos estadísticos para estudiar el rendimiento de los coches en simulaciones y el análisis de estrategias.

Este último deporte se ha visto envuelto en una mejora continua de las herramientas de análisis aplicadas a él, desarrollando nuevos modelos de aprendizaje automático e inteligencia artificial aplicados a todo tipo de ámbitos dentro de la categoría reina del automovilismo.

En este proyecto se busca abordar un análisis detallado de aspectos relevantes en la Fórmula 1. Se utilizarán modelos de inteligencia artificial, como varios tipos de redes neuronales, modelos de análisis del lenguaje natural y modelos de aprendizaje automático.

Por tanto se llevará a cabo un análisis detallado de datos relacionados con la competición, resultados históricos, sus circuitos y varios aspectos más que vienen recogidos en nuestra base de datos.

2. Terminología básica Fórmula 1

En esta página introductoria se explicarán conceptos básicos sobre la fórmula 1 para poder entender correctamente los contenidos de nuestra base de datos y de nuestros análisis:

La Fórmula 1 (F1) es la competición de automovilismo más prestigiosa y rápida del mundo, donde los mejores pilotos del planeta compiten en autos monoplaza diseñados con tecnología punta. Cada temporada se compone de una serie de carreras llamadas "Grandes Premios" (GP) que se celebran en diferentes países. Los pilotos y los equipos (escuderías) compiten por dos campeonatos: el Campeonato de Pilotos, que premia al mejor conductor, y el Campeonato de Constructores, que reconoce al equipo con mejor desempeño general. Los equipos, como Ferrari, Red Bull y Mercedes, tienen 2 pilotos cada uno y fabrican sus propios autos siguiendo estrictas normativas técnicas. Estos vehículos son extremadamente veloces, alcanzando más de 350 km/h, gracias a su diseño aerodinámico.

El fin de semana de carrera se divide en tres fases: pruebas libres (viernes), clasificación (sábado) y la carrera principal (domingo). La clasificación se desarrolla en tres rondas (Q1, Q2 y Q3), donde los pilotos más lentos son eliminados hasta que se define la "pole position" para la carrera. La carrera se disputa a lo largo de unos 305 km o 2 horas, con al menos una parada obligatoria para cambiar neumáticos. Los 10 primeros en cruzar la meta reciben puntos, con 25 puntos para el ganador, 18 para el segundo, 15 para el tercero y así sucesivamente. La estrategia de paradas en boxes es clave, ya que se utilizan distintos tipos de neumáticos (blandos, medios y duros), cada uno con distinta durabilidad y velocidad. Al final de la temporada, el piloto con más puntos se corona Campeón del Mundo, y el equipo con mejor rendimiento gana el Campeonato de Constructores.

Además de la velocidad y la estrategia, la seguridad es fundamental en la Fórmula 1. Los pilotos deben respetar los límites de la pista y evitar choques intencionales, ya que los comisarios pueden aplicar sanciones por infracciones La precisión es vital, ya que milésimas de segundo pueden definir al ganador. Esta combinación de ingeniería, estrategia, habilidad y velocidad hace de la F1 uno de los deportes más emocionantes e innovadores del mundo.

3. Base de datos y Análisis Exploratorio (EDA)

Para nuestro proyecto hemos escogido una base de datos con tablas relacionadas que nos ayudan a profundizar en nuestro análisis. La base de datos cuenta con 14 tablas que cuentan con información de diversos aspectos. Además de la información contenida en esta base de datos se ha extraído gran cantidad de información de páginas web y a través de APIs. La información recogida a través de estos métodos será indicada y se explicará su extracción para los análisis en los que se utilicen. Cabe destacar que todas nuestras tablas incluyen información de la competición desde 1950 hasta junio de 2024.

La base de datos está extraida del siguiente enlace:

https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020

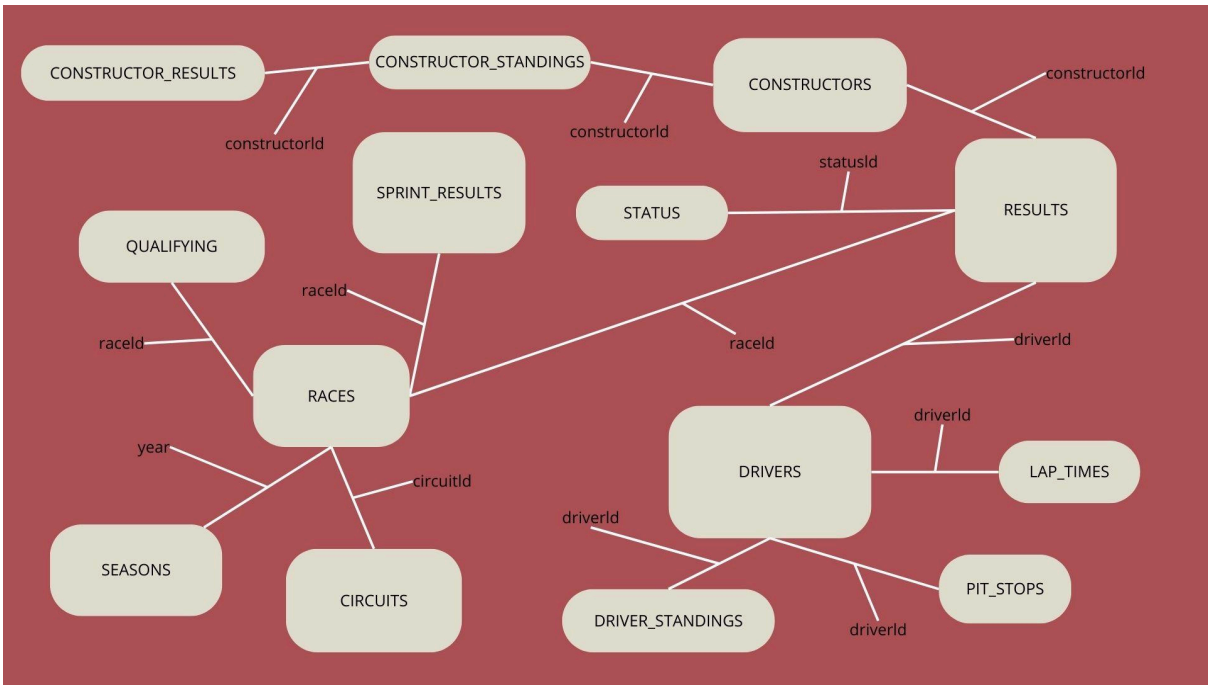
Previo a la elaboración de los algoritmos relacionados con las preguntas que trataremos de responder es necesario realizar un primer análisis exploratorio de los datos (EDA), con el objetivo de poder entender la base de datos antes de tratar con los mismos, así como identificar datos faltantes o constantes, patrones, relaciones entre variables o detectar valores atípicos antes de realizar análisis más complejos.

A continuación, se detallarán los contenidos de nuestras tablas:

1. **Circuits**: Recoge información sobre todos los circuitos que han tenido presencia en la competición en toda su historia. Entre todo el contenido de esta tabla se incluye el nombre de los circuitos, información sobre su localización y los enlaces al artículo de Wikipedia sobre ellos.
2. **Constructor_results**: Contiene información sobre los puntos conseguidos por cada constructor en cada carrera.
3. **Constructor_standings**: Contiene información sobre resultados de constructores, ordenados por posición e incluyendo victorias.
4. **Constructors**: Contiene información sobre los constructores, su país de procedencia y su enlace de Wikipedia.
5. **Driver_Standings**: Contiene los puntos ganados por carrera de los pilotos, entre otra información como su posición e incluyendo victorias.
6. **Drivers**: Contiene información sobre todos los pilotos que han pasado por la competición, como el nombre, su procedencia y su enlace a Wikipedia.
7. **Lap_times**: Contiene información sobre los tiempos de todas las vueltas realizadas por todos los pilotos en cada carrera.
8. **Pit_stops**: Contiene información sobre todas las paradas en boxes realizadas por cada piloto en todas las carreras, como el constructor que la realiza y el tiempo.
9. **Qualifying**: Contiene información sobre la clasificación de cada gran premio. Entre esta información está los tiempos de cada piloto en las 3 rondas.
10. **Races**: Contiene información sobre cada carrera como la fecha, el circuito o el enlace a Wikipedia del gran premio.
11. **Results**: Contiene información sobre los resultados en carreras de cada piloto, con información sobre el tiempo obtenido, las vueltas realizadas o el estado en el que acabó la carrera.
12. **Seasons**: Contiene el año de la temporada y su enlace a Wikipedia.

13. **Sprints_results**: Es similar a la tabla de results pero con resultados sobre los pilotos en las carreras al Sprint.
14. **StatusID**: Es un diccionario con identificador de estado y lo que significa cada uno.

Un gran número de estas tablas se utilizarán para la realización de nuestro proyecto, utilizando para varios análisis una combinación de ellas. A continuación se muestra el modelo de la relación existente en nuestra base de datos:



Haciendo referencia ahora a un primer breve análisis exploratorio de datos, en búsqueda de valores faltantes, constantes o cualquier tipo de atributo mal codificado; simplemente mencionar que no nos hemos encontrado ningún tipo de inconveniente.

Es cierto que hay datos faltantes, pero se corresponden con la realidad de la carrera, es decir, es completamente lógico que, para los pilotos que no se hayan clasificado a la Q2, su tiempo en dicha etapa, sea un valor faltante. Siguiendo esta lógica, no nos hemos encontrado con que haya valores faltantes en ningún elemento para el que sí que esperaríamos un valor real. Por tanto, el dataset, que además está continuamente siendo actualizado y depurado, no contiene deficiencias en el sentido de valores faltantes.

4. **¿Qué circuitos son parecidos en cuanto al ámbito geográfico y el diseño del circuito, para la preparación de coches según los circuitos en los que se obtuvieron mejores resultados?**

La pregunta que se pretende responder con este análisis la hemos llevado a cabo con Análisis de Componentes Principales (PCA), que es un método de disminución de dimensionalidad frecuentemente empleado en campos de minería de datos. Es utilizado para simplificar grupos de datos complejos al disminuir la cantidad de variables, manteniendo al mismo tiempo la mayor cantidad de información que se pueda.

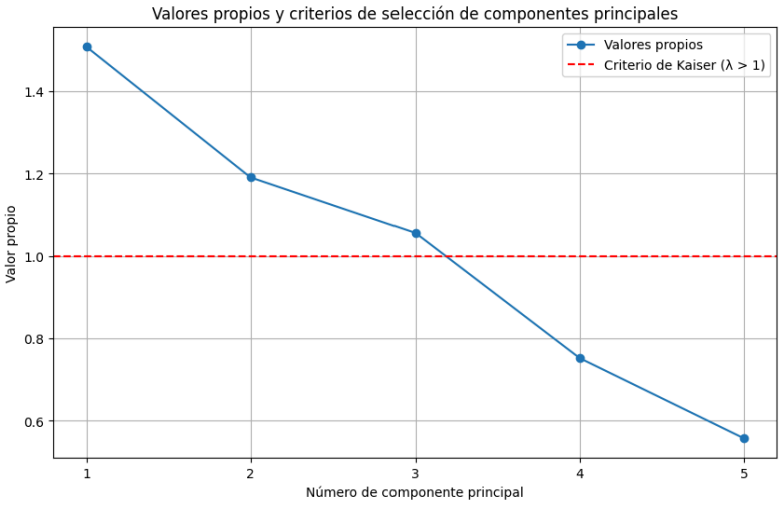
Nuestro análisis tiene como objetivo reducir la dimensionalidad de las variables que hacen referencia a los circuitos, para estudiar de una forma más sencilla en dos dimensiones cuales son los circuitos más parecidos.

Esto se hace desde el punto de vista de los mecánicos del equipo, ya que esto les servirá para preparar el coche antes de las carreras, así si se observa que dos circuitos son muy parecidos, podrán usar las mismas configuraciones que se hayan seleccionado en uno de ellos cuando el equipo ganó la carrera.

Para ello se han utilizado las variables geográficas de los circuitos para poder ver la climatología y otros aspectos, pero como se observó que esto no nos aporta suficiente información, se ha hecho un web scraping en las páginas de la wikipedia para todos los circuitos que están presentes en nuestra base de datos, obteniendo las variables número de curvas del circuito y la longitud del circuito.

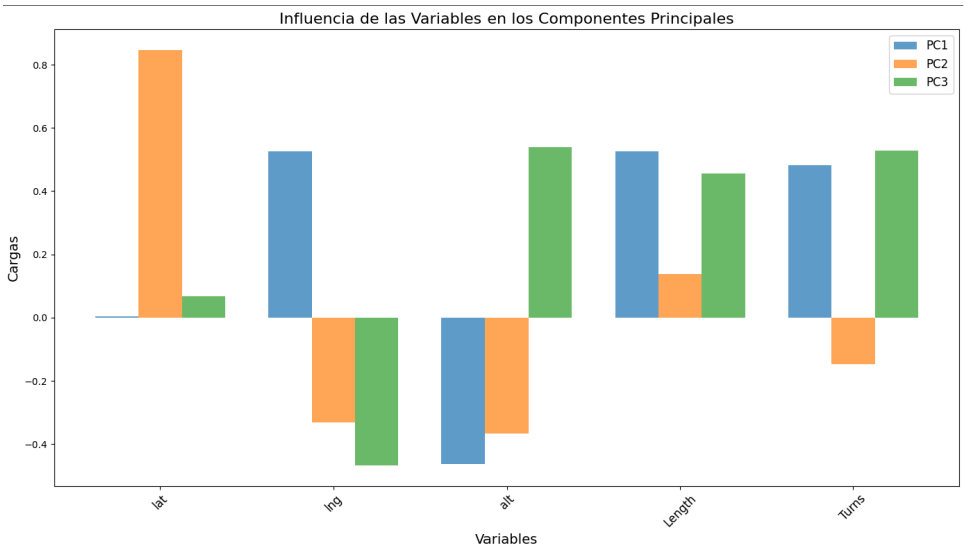
Como algunas de estas variables no se encontraron en wikipedia, se ha hecho una imputación con los valores de la media en la variable “longitud del circuito” y con la moda en la variable “turns”.

Lo primero que se realizó fue el web scraping y la limpieza de datos, después se utilizó el criterio de kaiser para observar cuantos componentes principales necesitaríamos para explicar una gran cantidad de información.



En nuestro análisis vamos a utilizar tres componentes principales, ya que con ellos somos capaces de explicar la mayor cantidad de información.

Con ello, procedemos a realizar la reducción de dimensionalidad y los gráficos para ver qué variables son las explicadas en cada componente principal y gráficos uno a uno para observar qué circuitos tienen similitudes.



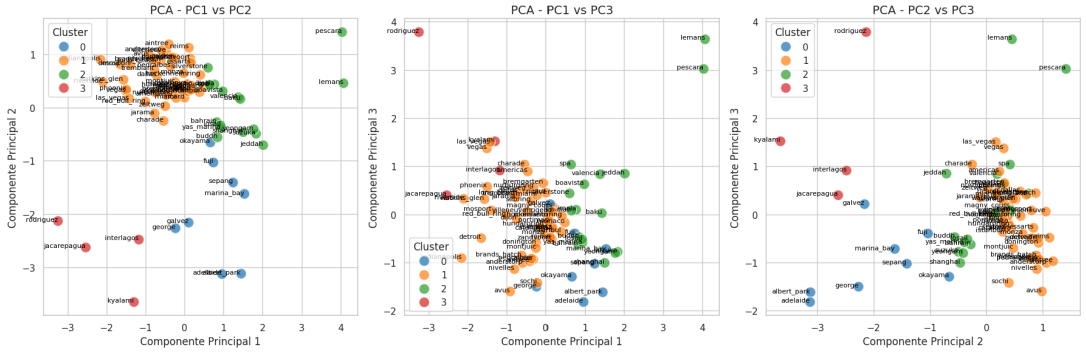
Como podemos observar en la imagen, el componente principal 1 (azul) parece que representa los factores relacionados con la longitud del circuito, el número de curvas y la longitud geográfica, teniendo en cuenta de forma negativa la latitud, por lo que parece representar la complejidad del circuito, con un matiz relacionado con la ubicación geográfica y la altitud.

El componente principal 2 (naranja) representa factores relacionados con las coordenadas geográficas, por lo que puede representar las características geográficas y topográficas de los circuitos de Fórmula 1.

Por último el componente principal 3 (verde), parece representar valores como la altitud, la longitud del circuito y el número de curvas, por lo que podemos decir que representa características topográficas y el diseño del circuito.

Una vez tenemos conocimiento de lo que representa cada componente principal se han hecho los gráficos de componentes principales uno a uno con un clustering, para ver qué circuitos son iguales.

El clustering es un método de aprendizaje no supervisado basado en el aprendizaje automático. La meta principal consiste en agrupar la información en grupos de forma que los objetos pertenecientes a un mismo grupo sean más parecidos entre ellos que los de otros grupos.



Los diagramas ilustran la forma en que los circuitos se agrupan o se separan basándose en los tres componentes fundamentales calculados anteriormente. La cercanía o distancia entre puntos señala la semejanzas o discrepancias entre los circuitos.

Como podemos observar existe una gran cantidad de circuitos con características similares, esto se debe a que los ya que los circuitos con color amarillo son los que se encuentran en su mayoría en Europa y Norteamérica por lo que tiene sentido que tengan características muy parecidas.

Existen también grupos de circuitos más pequeños con características similares, los circuitos de color verde: Lemans, Pescara, Jeddah, Baku, Valencia, son circuitos que se diferencian por su longitud extrema combinado con otras características geográficas similares.

Por otro lado, otro pequeño grupo de circuitos similares son los de color rojo que tienen características diferentes a los demás ya que son circuitos en su mayoría de América del Sur y África, con diferencias de altitud, por lo que tendrá mucha influencia la climatología.

Por último está el grupo de color azul que son circuitos como: Adelaide, Albert Park, Marina Bay, Sepang, George, Okayama, que tienen características similares debido a su ubicación geográfica y diseño del circuito en referencia al número de curvas.

Respondiendo a la pregunta, existirían cuatro tipos de circuitos que tendrían características parecidas, esto es muy interesante para los equipos, ya que podrán preparar el coche en función de los mejores resultados que obtuvieron en cada grupo.

Es importante tener en cuenta que las características geográficas hacen también referencia a la climatología por lo que tener en cuenta estas variables es de vital importancia.

5. **A partir de un entrenamiento en el circuito de la carrera días anteriores, ¿será el piloto capaz de terminar en podio?**

En este análisis, emplearemos un Perceptrón Multicapa es una clase de red neuronal artificial que se utiliza en el aprendizaje supervisado, y se compone de múltiples capas de neuronas. Compone una arquitectura básica pero esta es fundamental en el ámbito del deep learning.

En nuestro caso hemos decidido usarlo para predecir si un piloto a partir de un entrenamiento en ese circuito, será capaz de quedar en podio en la carrera para la que se está preparando.

Para ello se han utilizado las variables “posición de salida en la parrilla”,”nacionalidad del conductor”, “años de experiencia del piloto”, “circuito de referencia”,”nombre del equipo”,”milisegundos que se tardaron en terminar la carrera entera”, “velocidad máxima”.

Con todo ello hemos creado una red neuronal multicapa para que aprenda los patrones de los conductores que han quedado en la posición de podio y así poder predecir.

Lo primero que se ha realizado ha sido unir todas las tablas necesarias para el análisis y la selección de variables importantes para el análisis, además se ha creado una nueva variable llamada podio, que será la variable respuesta con la que el perceptrón realizará el estudio.

A continuación se ha creado la red neuronal multicapa y la hemos entrenado para que realice las predicciones, obteniendo estos resultados:

Rendimiento del modelo (¿Terminará en el podio?):

	precision	recall	f1-score	support
0	0.95	0.95	0.95	4286
1	0.64	0.62	0.63	590
accuracy			0.91	4876
macro avg	0.80	0.79	0.79	4876
weighted avg	0.91	0.91	0.91	4876

Como podemos observar en los resultados, la predicción de “no termina en el podio” tiene una precisión del 95% lo que quiere decir que de las veces que el modelo predijo que un piloto no termina en el podio, acertó un 95%. Además hay 4286 ejemplos de pilotos que no terminaron en el podio en el conjunto de prueba, por lo que se cuentan con muchos datos para el entrenamiento.

Por otro lado la predicción de “termina en el podio” tiene resultados algo peores, cuenta con un 64% de precisión es decir que de las veces que el modelo predijo que el piloto acababa en podio acertaba el 64% de las veces. Esto se puede deber a que contamos con menos datos de pilotos que han acabado en podio como podemos ver en “support”.

Aún así podemos decir que es un buen modelo de predicción, ya que cuenta con un accuracy del 0.91, lo que nos indica que el modelo predice correctamente el resultado en el 91% de los casos, lo cual es muy beneficioso para los equipos que lo usen, ya que sabrán que tienen que mejorar en diferentes aspectos para poder quedar en el podio.

Respondiendo a la pregunta principal la capacidad de predicción de si el piloto acabará en podio es buena, pero se debe tener en cuenta que este resultado depende de muchos otros factores, es por ello que aunque nuestro modelo predice “podio si” solo aceptará el 64% de las veces.

Al contrario que si nos predice “podio no”, en la que existirá una pequeña posibilidad del 9% de que ante los pronósticos negativos, los rompa y consiga el equipo quedar en podio.

Todo el detalle del código para realizar este análisis y el anteior se encuentra en ANEXO 1

6. ¿Es importante tener rápidas paradas en boxes durante la carrera?

Uno de los tantos elementos que conforman una carrera de Fórmula 1 son las paradas en boxes, también llamadas pit stops. Concretamente, se usa este término para aludir a la "parada que hace un vehículo durante la competición para repostar, cambiar los neumáticos, hacer reparaciones o ajustes mecánicos". Son paradas que duran generalmente menos de 1 minuto, ya que el resto de pilotos siguen en la carrera mientras el que ha entrado a repostar está parado, por lo que tratar de ser rápido y manejar una estrategia de número de paradas eficientes puede resultar determinante para el resultado de una carrera.

A raíz de esto, y entendiendo que todos los pilotos parten de una situación similar en términos de gasto de combustible o meteorología (relevante para el tipo de neumático) es esperable que estas paradas sean muy determinantes, es entonces cuando la surge la pregunta siguiente: **¿es extremadamente importante tener rápidas paradas en boxes durante la carrera o, por el contrario, un buen trabajo previo (en términos de buena clasificación en la parrilla de salida) puede ser más relevante a la hora de obtener mejores resultados en los pilotos?**

Para responder a esta pregunta hemos utilizado principalmente la tabla pit_stops, por lo que cada fila representa una única parada. Por tanto, el input del modelo será siempre una única parada, es decir, predecimos su posición final (categorizada) en base a esa parada.

No obstante, se ha recogido la siguiente información adicional para preparar el modelo: **raceId** (para ver en qué carrera se hace la parada), **driverId** (para ver qué piloto hace la parada), **num_stops** (lo hemos creado para ver cuantas paradas en total hizo un mismo piloto en una misma carrera, este dato se repetirá lógicamente para todas las paradas de la misma carrera del piloto), **grid** (posición en la parrilla de salida del piloto que realiza la parada en esa carrera), **milliseconds** (duración en milisegundos de la parada) y **percentil_parada**. La variable percentil_parada tiene muchísima importancia ya que indica para cada parada lo "rápida" o "lenta" que ha sido con respecto al total de paradas de la carrera. Es decir, agrupamos todas las paradas por su raceId y calculamos su distribución. Seguidamente, a cada parada le asignamos su valor del percentil, es decir, para raceID=1092, la parada con percentil_parada=0 fue la parada más rápida de la carrera 1092, la parada con percentil_parada=0.5 fue la parada mediana y la parada con percentil_parada=1 fue la parada más lenta de la carrera. Es importante recordar que estos valores se agrupan por carrera, ya que tiene sentido calcular cuánto de rápida o lenta es la parada comparada únicamente con las paradas de la misma carrera.

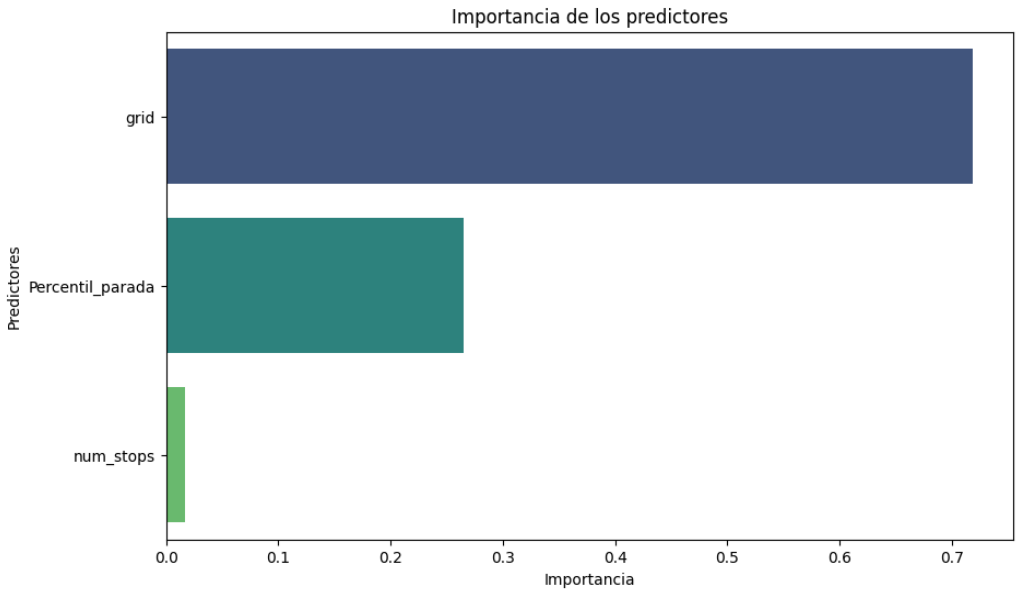
Por último, la variable objetivo es **position_category**, que se define agrupando positionOrder (de la tabla results) en base a: Podio (posiciones 1 a 3), Top 10 sin podio (4 a 10), Sin puntos (11 a 25). De cara a introducir las variables al modelo se han considerado únicamente grid, percentil_parada y num_stops como predictoras, ya que el resto no aportan información adicional; y como variable objetivo position_category.

El modelo aplicado a estos datos ha sido un Random Forest, ya que nos parece interesante explorar diferentes árboles de decisión (500 en nuestro caso), para tener un modelo muy robusto, como máxima profundidad hemos determinado 3, ya que tras probar más profundidad y ver que el accuracy no mejora notablemente, hemos preferido una mejor interpretación. Mencionar también que, al tener clases desbalanceadas, hay muchas más paradas con pilotos que han finalizado sin puntos de las paradas con pilotos que acaban en podio; por ello, hemos aplicado un balance con SMOTE, para hacer sobremuestreo en las clases minoritarias.

Con todo lo anterior, los resultados han sido los siguientes:

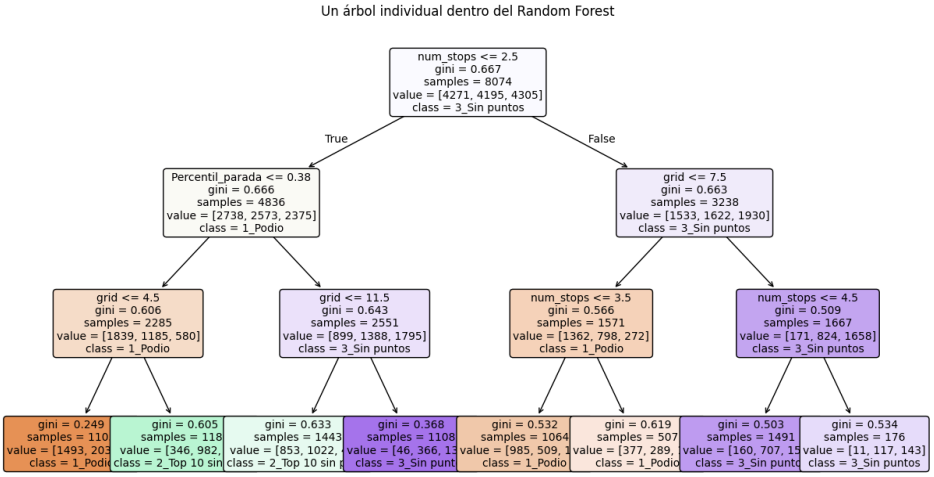
```
Classification Report:
                    precision    recall  f1-score   support

          1_Podio       0.51      0.81      0.63       307
2_Top 10 sin podio       0.54      0.46      0.50       740
      3_Sin puntos       0.76      0.71      0.74      1051

          accuracy                           0.64      2098
         macro avg       0.60      0.66      0.62      2098
      weighted avg       0.65      0.64      0.64      2098


Confusion Matrix:
[[250  43  14]
 [179 338 223]
 [ 60 241 750]]
```

El modelo presenta un accuracy del 64%, consideramos que está razonablemente bien, ya que predecir la categoría en la que va a acabar un piloto únicamente con lo buena que ha sido la parada, el número de paradas totales y su parrilla de salida, es muy complejo, debido a que, como hemos visto en análisis anteriores, son muchas las variables que componen la probabilidad de victoria, o en este caso, de podio o puntos. Por último, se presentan dos gráficos que nos van a ayudar a responder a la pregunta inicial.



El primero de ellos nos muestra la importancia que han tenido los predictores porcentualmente en la construcción de los árboles de random forest:

- **grid: 0.72**. Esto indica que la importancia de la parrilla de salida en predecir la categoría final ha sido de más del 70%!! Volviendo a la pregunta inicial, está información concluye que, claramente, un buen trabajo previo de cara a conseguir una excelente posición en la parrilla de salida es mucho más determinante que la estrategia de paradas.
- **percentil_parada: 0.27**: En segundo lugar, tenemos que hacer paradas rápidas ha tenido una importancia del 27%, lo cual parece razonable ya que su importancia no es extrema pero tampoco irrelevante
- **num_stops: 0.01**. Este dato ha sido muy sorprendente porque carece de sentido que el número de paradas total no tenga importancia, cuando se pierden unos 30 segundos cada vez que haces una parada.



Este segundo gráfico es uno de los 500 árboles, seleccionado aleatoriamente, que ha realizado el algoritmo Random Forest y de aquí extraemos varias conclusiones: por si fuera poco sorprendente que num_stops tuviera un importancia en RF del 1%, más aún si viendo un primer árbol, es la primera rama de decisión del árbol, no obstante,este solo es uno de los 500 árboles realizados. Por otro lado, es bueno que las 3 variables introducidas al modelo tengan presencia en la construcción del árbol. De nuevo, como hemos mencionado antes, tomar este ejemplo con precaución, sólo es una de las 500 simulaciones realizadas.

Finalmente, y contestando a la pregunta inicial, nuestra respuesta a la misma como conclusión sería que **el hecho de no tener buenas paradas en boxes o un número más alto del recomendado para cada carrera, se puede muy claramente compensar si ha habido un buen trabajo previo en el fin de semana, consiguiendo una privilegiada posición en la parrilla de salida.**

El detalle del código ejecutado para responder esta pregunta se encuentra en ANEXO 2.

7. ¿Qué características influyen más en el entretenimiento de las carreras?

Es común que en una temporada haya carreras que nos levantan de nuestros asientos en cada momento y otras en la que tenemos que esforzarnos para no quedarnos dormidos. En este apartado buscaremos analizar qué características son las más influyentes en hacer que una carrera sea entretenida o no sea nada especial.

Para realizar este análisis se buscará extraer el sentimiento generalizado de las carreras de las últimas 2 temporadas. Con información y métricas sobre estas carreras se construirá un modelo Random Forest del que extraeremos la importancia de las características.

Para evaluar los sentimientos de las carreras se ha recurrido a las publicaciones de Reddit del subreddit oficial de la Fórmula 1. En este subreddit se publica una discusión post carrera de todas las carreras que se disputan, publicaciones de las cuales hemos extraído 250 a través de la API Praw para poder hacer un análisis de sentimientos con procesamiento del lenguaje natural (NLP). Una vez extraidos los comentarios se han analizado con VADER para extraer los sentimientos generales de cada carrera.

Una vez que tenemos los sentimientos por carrera, pasamos a realizar web scrapping para extraer información de esta que podría ser relevante para nuestro análisis. También se extraen características interesantes de forma manual. Cabe destacar que hay algunas características que serían interesantes incluir en este análisis, como el número de accidentes en una carrera, pero que no están disponibles para extraer.

Con la información de las carreras y los sentimientos generales de cada una, podemos pasar a definir un modelo random forest que evaluará las características en base a la ganancia de información que obtenga para poder clasificar las carreras entre entretenidas o neutrales. Cabe destacar que conocemos las limitaciones de este método, pues al ser el scrapping de comentarios un proceso tedioso y con altos requisitos computacionales no hemos obtenido toda la información de carreras que nos hubiera gustado para definir un buen modelo, dando lugar a entrenar el árbol con 50 datos aproximadamente. Aún así mostraremos los resultados del análisis de características.

Todo el código empleado para el scrapping de comentarios se encuentra en ANEXO 3

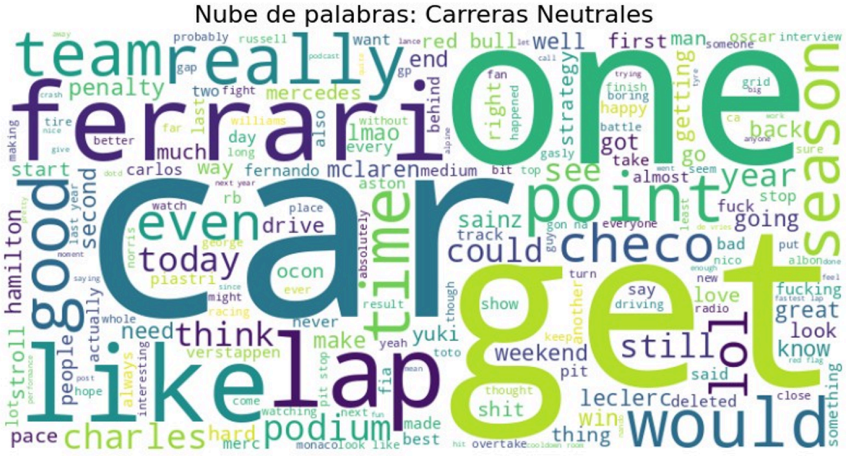
Todo el código empleado para el análisis de sentimientos se encuentra en ANEXO 4

Todo el código empleado para la construcción del modelo Random Forest y el análisis de influencias de las características se encuentra en ANEXO 5.

Resultados Relevantes:

Como parte del análisis de sentimientos hemos considerado interesante generar una nube de palabras para las carreras entretenidas y las carreras neutrales, con el fin de ver si las opiniones sobre estos tipos siguen alguna tendencia y tienen temas en común. A continuación observamos la nube de palabras para las carreras entretenidas:

A pesar de que no muestre una estructura muy clara se puede observar cierta tendencia a tratar temas resultadistas. La discusión sobre los puntos, victorias o resultados de podio puede ser un indicador de que la carrera ha tenido resultados inesperados o resultados dignos de comentar. Esto hace que la carrera haya sido más atractiva para el espectador.



Sin embargo, en esta gráfica no aparece ninguna tendencia significativa, siendo protagonistas de la nube palabras planas y sin mucha relación entre ellas. Esto nos puede indicar que la carrera no ha tenido un tema concreto que haya dado que hablar dando lugar a una carrera sin entretenimiento.

A continuación vamos a observar la influencia que tienen las características en la clasificación:

	Característica	Importancia
1	Course Length (m)	0.238619
5	adelantamientos	0.236518
2	Laps	0.140671
3	turns	0.138579
4	duracion	0.138473
0	weather	0.107140

Observamos que dos características se han diferenciado del resto. La primera es la longitud en metros del trazado. Esta importancia puede tener su origen en las ocurrencias de un trazado. Al ser el trazado más largo puede tener más puntos de interés en el que puedan ocurrir eventos que atraigan al espectador. Uno de estos eventos son sin duda los adelantamientos, que suponen un gran atractivo en todas las carreras. Cuantos más adelantamientos una carrera será más entretenida, pues ocasionan cambios de posición, duelos entre pilotos y pueden dar lugar a accidentes.

El bajo y sorprendente resultado de la climatología se debe a la baja representación que han tenido las carreras con lluvia estas últimas 2 temporadas. Esto sumado a la tendencia de los últimos años a cancelar grandes premios con clima adverso hace que no suponga una característica relevante en estos años.

En resumen, cuantos más eventos ocurran en una carrera, más interesante será. Carreras con pocos puntos de interés o pocos adelantamientos no serán atractivas para el público. Esto ha ocasionado debate en la competición en los últimos años, pues a menudo se han propuesto cambios en los circuitos para tratar de evitar trazados que cumplen las características de tener pocos puntos de interés y de adelantamientos.

8. **¿Cual es la probabilidad de que los equipos actuales se encuentren en una época de dominancia?**

En la historia de la fórmula 1 hemos visto años en los que un equipo ha sido firme dominador con respecto al resto. El caso más reciente es la dominancia del equipo Mercedes entre los años 2014 y 2020 de la mano de Lewis Hamilton, ganando 7 títulos consecutivos. Este es el caso más reciente pero para nada el único, pues durante toda la historia de la competición hemos visto situaciones parecidas en diversas ocasiones, como con Michael Schumacher y Ferrari desde el año 2000 al 2004 o con el equipo Williams de 1992 a 1997. Estas épocas se caracterizan por un aplastante dominio sobre el resto de pilotos y equipos.

En los últimos años el equipo Red Bull y Max Verstappen han conseguido resultados extraordinarios, en muchos casos dando lugar a esta sensación de dominio que ya ha sido experimentado anteriormente. Max Verstappen ha ganado los últimos 4 campeonatos, aunque los resultados de este último correspondiente a 2024 no se encuentran disponibles en nuestra base de datos. Debido a esto, es de carácter general plantearse si los resultados de Max Verstappen junto a Red Bull son comparables a las anteriores épocas de dominio.

Para responder esto vamos a tratar de predecir la probabilidad de que los equipos actuales se encuentren en una época de dominancia o no. Construiremos un dataframe con todos los resultados de todos los pilotos en todos los grandes premios de la historia. Para ello juntaremos información de la table results, circuits, races, pit_stops y qualifying.

Dividiremos nuestro dataframe para entrenar una Red Neuronal estilo Perceptrón Multicapa con los datos hasta 2020 y utilizaremos el modelo resultante para predecir los resultados de 2021 en adelante, y con ello predecir la probabilidad que Red Bull se encuentre en una época de dominancia.

En muchas ocasiones observamos valores faltantes en nuestros datos, sin embargo, esto no podemos ignorarlo o eliminarlos pues aportan información muy valiosa. Por ejemplo, si un piloto se ha eliminado en primera ronda de la clasificación, sus tiempos en las otras 2 rondas serán valores faltantes pero constituyen información importante para analizar los resultados de los pilotos. Por ello, estos valores faltantes se imputarán con valores exagerados para que se valoren como si hubieran conseguido un tiempo exageradamente malo.

Añadiremos a nuestro conjunto de test una columna binaria que será la variable respuesta de dominancia o no, seleccionando como dominantes los equipos y épocas mencionadas. Tras esto pasaremos a entrenar la red y realizar las predicciones.

```
Validation Loss: 0.05758117884397507
Validation Accuracy: 0.9779647588729858
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Probabilidad de dominancia para el equipo McLaren en toda la época
(2021 en adelante): 0.3923
Probabilidad de dominancia para el equipo Williams en toda la época
(2021 en adelante): 0.0884
Probabilidad de dominancia para el equipo Ferrari en toda la época
(2021 en adelante): 0.5892
Probabilidad de dominancia para el equipo Red Bull en toda la época
(2021 en adelante): 0.7432
Probabilidad de dominancia para el equipo Sauber en toda la época
(2021 en adelante): 0.1292
Probabilidad de dominancia para el equipo Alfa Romeo en toda la época
(2021 en adelante): 0.0995
Probabilidad de dominancia para el equipo Aston Martin en toda la
época (2021 en adelante): 0.2434
Probabilidad de dominancia para el equipo Mercedes en toda la época
(2021 en adelante): 0.6335
Probabilidad de dominancia para el equipo Haas F1 Team en toda la
época (2021 en adelante): 0.1042
Probabilidad de dominancia para el equipo AlphaTauri en toda la época
(2021 en adelante): 0.1543
Probabilidad de dominancia para el equipo Alpine F1 Team en toda la
época (2021 en adelante): 0.2012
Probabilidad de dominancia para el equipo RB F1 Team en toda la época
(2021 en adelante): 0.2326
```

En esta figura observamos que la red ya ha sido entrenada y ha obtenido excelentes resultados en las predicciones. Vemos que el modelo cuenta con un accuracy cercano al 0,98 y una pérdida inferior a 0,06. Con este modelo hemos realizado las predicciones de cada equipo actual para obtener los resultados que vemos en el gráfico superior.

Se observa que como podíamos observar el equipo Red Bull es el que tiene más probabilidad de estar en una época de dominancia con una probabilidad de 0,74. A este le siguen Mercedes y Ferrari, lo cual tiene mucho sentido pues han sido sus dos principales competidores en los últimos años. El resto de equipos toman valores similares en la zona baja, destacando el equipo McLaren que en estos últimos años ha solido luchar por el 4º puesto de constructores.

En resumen, el planteamiento general de los espectadores de pensar que Red Bull ha estado dominando la categoría reina del automovilismo se ve contrastado al compararse con otras épocas de dominancia. Esta alta probabilidad del equipo nos muestra que Red Bull es sin duda el equipo más dominante de los últimos años. y que está muy cerca de crear un nuevo legado en la historia del deporte

Todo el el detalle del código empleado para la realización del análisis de dominancias se encuentra en ANEXO 6

9. **¿Cuáles son las características de un circuito según su trazado? ¿Cual es el circuito característico de nuestro país?**

En toda la historia de la fórmula 1 se ha corrido en 76 circuitos distintos, muchos son míticos como el circuito de Mónaco y muchos otros son de reciente creación, como el circuito de las vegas. Todos estos circuitos cuentan con sus características y hacen que cada carrera sea única.

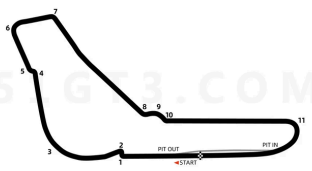
Por ejemplo el circuito de Mónaco siempre se ha caracterizado por contar con pocos adelantamientos en su gran premio, o el circuito de Monza se ha caracterizado por alcanzar velocidades inmensas. Estas características propias de cada circuito suelen venir dadas por la disposición del circuito, es decir, su trazado.

En este análisis se va a tratar de clasificar nuevos circuitos que no han participado nunca en la fórmula 1 a través de únicamente su trazado. Clasificaremos los circuitos en base a 3 aspectos: Velocidad (Rápido o Lento), Seguridad (Seguro o Inseguro) y Adelantamientos (Muchos o Pocos).

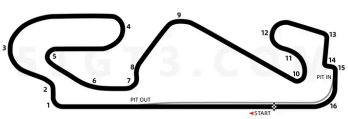
Para comenzar hemos definido estas características de nuestros circuitos presentes en la historia de la F1. Esta información ha sido extraida manualmente pues es información que no se encuentra de fácil acceso en páginas web. Una vez que tenemos nuestros circuitos clasificados vamos a proceder a extraer los trazados de los circuitos a través de Web Scrapping.

Para extraer las imágenes hemos descargado todas las imágenes de una página web de automovilismo y luego seleccionado los circuitos que corresponden a nuestra base de datos. Una vez que ya tenemos todas nuestras imágenes de trazados guardadas vamos a realizar herramientas de Data Augmentation para generar nuevas imágenes y poder entrenar una red neuronal convolucional (CNN) que pueda predecir las características de los circuitos. La labor de Data Augmentation es necesaria debido al bajo número de circuitos con los que contamos en relación a la cantidad de imágenes necesarias para entrenar correctamente una red convolucional.

EJEMPLO FOTOS ORIGINALES:



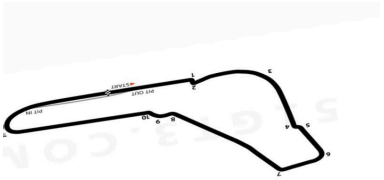
Autódromo Nazionale de Monza



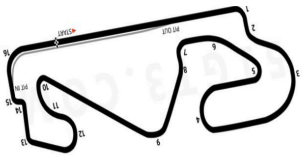
Circuito de Barcelona-Cataluña

Circuito de Mónaco

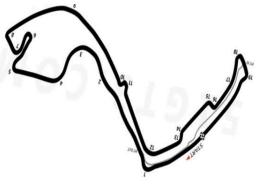
EJEMPLO FOTOS GENERADAS:



Autódromo Nazionale de Monza



Circuito de Barcelona-Cataluña



Circuito de Mónaco

Cuando ya tenemos todas las imágenes generadas, definiremos un bucle que etiquete las imágenes originales y aleatorias según el circuito al que corresponden. Todo el detalle de la carga y generación de imágenes y etiquetas se encuentra recogido en el ANEXO 7.

Una vez que tengamos todas las imágenes guardadas y etiquetadas pasaremos a definir las funciones necesarias para cargar las imágenes junto a las etiquetas en la red neuronal. Utilizamos esta función para cargar las imágenes y dividimos nuestro dataset en entrenamiento y validación. Tras esto ya tenemos todo preparado para definir nuestra red neuronal convolucional.

Para entrenar el mejor modelo posible de nuestra Red Convolucional definiremos una rejilla con distintos hiperparámetros para definir un bucle que pruebe todas las combinaciones posibles de hiperparámetros y escoja el mejor modelo. Evaluaremos nuestra red con el f1-score, no por la coincidencia de su nombre con el tema del proyecto sino por su capacidad de evaluar en datos desbalanceados, pues nuestras características cuentan con más representación de unos valores que de otros. Los hiperparámetros con distintos valores dentro de la rejilla son el número de neuronas de cada capa, el dropout rate, el learning rate, el tamaño del batch y el número de épocas.

Tras casi 24 horas de entrenamiento (1406 min) encontramos el mejor modelo posible con un valor de 0.75 en el f1-score. Este valor es suficiente para poder afirmar que tenemos un buen modelo que sea capaz de generalizar a nuevos circuitos sin temer a riesgos de sobreajuste.

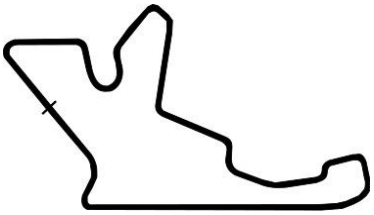
```
                    "num_filters": num_filters,
                    "dropout_rate": dropout_rate,
                    "learning_rate": learning_rate,
                    "batch_size": batch_size,
                    "epochs": epochs
                }


# Guardar el mejor modelo
best_model.save("mejor_modelo_circuitos.h5")
print(f"Mejor modelo guardado con F1 Score: {best_f1}")
print(f"Mejores hiperparámetros: {best_params}")
```

[8] ✓ 1406m 27.4s

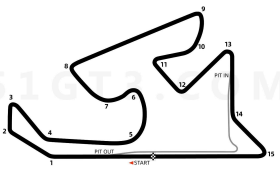
```
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001, batch_size 32, epochs 20
Epochs 20: 100%|██████████| 20/20 [03:08<00:00,  9.44s/it]
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step
F1 Score en validación: 0.7435032699572858
c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001, batch_size 32, epochs 30
Epochs 30: 100%|██████████| 30/30 [04:07<00:00,  8.24s/it]
```

Tras esto podemos utilizar nuestra red neuronal para predecir las características de los nuevos circuitos. Utilizaremos el trazado de 5 circuitos españoles para ver qué características tendrían en el caso de que se incluyeran en la Fórmula 1. Los circuitos que predecimos serán:

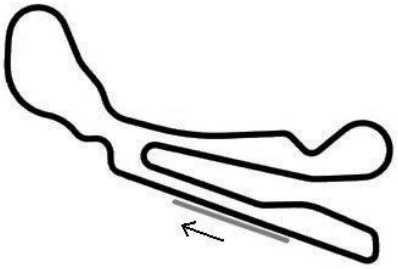
Circuito de Cheste



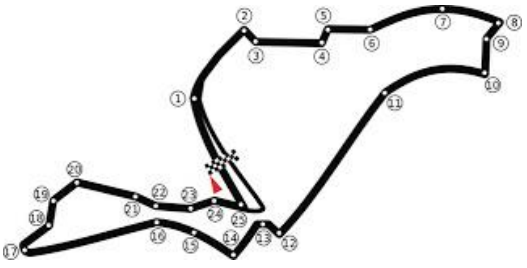
Circuito de Almería



Circuito de Navarra



Circuito de Guadix



Circuito Urbano de Valencia

Hemos obtenido las siguientes predicciones:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Imagen: CHESTE.png
Predicción continua: [[1.0, 0.9999998807907104, 0.9999698996543884]]
Clasificación :  Rápido,  Seguro,  Muchos Adelantamiendos

Imagen: CIRCUITO ALMERIA.png
Predicción continua: [[0.9999918341636658, 0.992418110370636, 0.0011154836975038052]]
Clasificación :  Rápido,  Seguro,  Pocos Adelantamientos

Imagen: CIRCUITO DE NAVARRA.png
Predicción continua: [[1.0, 0.9999995231628418, 0.8977177739143372]]
Clasificación :  Rápido,  Seguro,  Muchos Adelantamiendos

Imagen: CIRCUITO GUADIX.jpg
Predicción continua: [[0.999999463558197, 1.584191522852052e-05, 0.01137122604995966]]
Clasificación :  Rápido,  Inseguro,  Pocos Adelantamientos

Imagen: CIRCUITO VALENCIA.png
Predicción continua: [[0.0001163847555289763, 0.9985663890838623, 4.4724185954692075e-07]]
Clasificación :  Lento,  Seguro,  Pocos Adelantamientos
```

Gracias a nuestro modelo hemos podido ver las características de los trazados de estos circuitos, aunque podemos utilizar el modelo para evaluar más trazados. A continuación evaluaremos nuestro modelo en todos los circuitos españoles para tratar de ver las características generales y la tendencia en nuestro país. Si realizamos las predicciones y vemos la media de las todas las predicciones y calculamos la moda para la clasificación binaria vemos lo siguiente:

```
Media de las predicciones continuas: [0.87536575 0.74794224
0.24814561]
Moda de las predicciones binarias: [np.int64(1), np.int64(1),
np.int64(0)]
```

Vemos que la tendencia de los circuitos españoles está claramente definida. Gracias a nuestro análisis podemos predecir cómo actuarían circuitos humildes en competiciones del más alto nivel. Con esta información se podría intuir el espectáculo y entretenimiento que otorgarían en una competición de alto calibre, ayudando a ver los aspectos en los que se podría mejorar. Además, hemos podido ver la tónica general de los circuitos españoles, que nos ha llevado a poder afirmar que el circuito característico de nuestro país sería un circuito rápido, seguro y con pocos adelantamientos.

Todo el detalle de la definición de la red neuronal convolucional, su entrenamiento y predicciones se encuentra en ANEXO 8

10. Conclusión

En este proyecto hemos visto cómo el mundo del análisis de datos tiene una inmensidad de aplicaciones en el mundo del deporte, un mundo en el que todavía no se han implementado estas tecnologías todo lo que podría. En concreto hemos buscado responder preguntas relacionadas con el mundo del automovilismo y la Fórmula 1, su presente y la influencia histórica.

Hemos podido comprobar la capacidad que tiene la inteligencia artificial de realizar análisis con precisión a través de redes neuronales. Estas herramientas han ido creciendo en estos últimos años y actualmente forman parte de las herramientas utilizadas en los proyectos de más alto nivel de análisis e investigación. También hemos visto cómo aplicar todas las herramientas aprendidas en un proyecto completo de minería de datos, desde la extracción del conjunto de datos a utilizar a la búsqueda de respuestas a nuestras preguntas.

A través de estas técnicas hemos podido extraer información no trivial de nuestro conjunto de datos históricos de la competición, así como desarrollar modelos que no solo nos ayudan a comprender resultados actuales sino que también nos pueden ayudar a encontrar respuestas en el futuro.

En resumen, en este proyecto hemos conseguido descifrar información relevante de la fórmula 1 a través de un proyecto exhaustivo de minería de datos. Pensamos que los resultados obtenidos han sido satisfactorios y que corresponden únicamente a una pequeña parte de todo lo que se puede extraer de un ámbito tan grande como el mundo del automovilismo. Por ello, vemos necesario seguir desarrollando técnicas que se implementen en este mundo para mejorar los resultados de la competición.

ANEXO 1

#IMPLEMENTACIÓN DE LA BASE DE DATOS

```
from google.colab import files

# Sube los archivos CSV descargados
uploaded = files.upload()

# Guarda los nombres de los archivos cargados
import pandas as pd

# Leer cada archivo CSV y almacenarlo en un DataFrame
drivers = pd.read_csv('drivers.csv')  # Cambia el nombre según los
archivos
results = pd.read_csv('results.csv')  # Cambia el nombre según los
archivos

# Verifica los datos cargados
print(drivers.head())
print(results.head())
```

```
<IPython.core.display.HTML object>

Saving circuits.csv to circuits.csv
Saving constructor_results.csv to constructor_results.csv
Saving constructor_standings.csv to constructor_standings.csv
Saving constructors.csv to constructors.csv
Saving driver_standings.csv to driver_standings.csv
Saving drivers.csv to drivers.csv
Saving lap_times.csv to lap_times.csv
Saving pit_stops.csv to pit_stops.csv
Saving qualifying.csv to qualifying.csv
Saving races.csv to races.csv
Saving results.csv to results.csv
Saving seasons.csv to seasons.csv
Saving sprint_results.csv to sprint_results.csv
Saving status.csv to status.csv
   driverId   driverRef number code  forename     surname         dob
\
0         1    hamilton     44  HAM     Lewis    Hamilton  1985-01-07

1         2    heidfeld     \N  HEI      Nick    Heidfeld  1977-05-10

2         3     rosberg      6  ROS      Nico     Rosberg  1985-06-27

3         4      alonso     14  ALO  Fernando      Alonso  1981-07-29

4         5  kovalainen     \N  KOV    Heikki  Kovalainen  1981-10-19


  nationality                                              url
```

```
0     British     http://en.wikipedia.org/wiki/Lewis_Hamilton
1      German      http://en.wikipedia.org/wiki/Nick_Heidfeld
2      German       http://en.wikipedia.org/wiki/Nico_Rosberg
3     Spanish    http://en.wikipedia.org/wiki/Fernando_Alonso
4     Finnish  http://en.wikipedia.org/wiki/Heikki_Kovalainen
   resultId  raceId  driverId  constructorId number  grid position  \
0         1      18         1              1     22     1        1
1         2      18         2              2      3     5        2
2         3      18         3              3      7     7        3
3         4      18         4              4      5    11        4
4         5      18         5              1     23     3        5

  positionText  positionOrder  points  laps          time milliseconds
\
0            1              1    10.0    58  1:34:50.616      5690616

1            2              2     8.0    58       +5.478      5696094

2            3              3     6.0    58       +8.163      5698779

3            4              4     5.0    58      +17.181      5707797

4            5              5     4.0    58      +18.014      5708630


  fastestLap rank fastestLapTime fastestLapSpeed  statusId
0         39    2       1:27.452         218.300         1
1         41    3       1:27.739         217.586         1
2         41    5       1:28.090         216.719         1
3         58    7       1:28.603         215.464         1
4         43    1       1:27.418         218.385         1
```

```python
import pandas as pd

# Cargar las tablas CSV en DataFrames
drivers = pd.read_csv('drivers.csv')
constructors = pd.read_csv('constructors.csv')
circuits = pd.read_csv('circuits.csv')
races = pd.read_csv('races.csv')
results = pd.read_csv('results.csv')
pit_stops = pd.read_csv('pit_stops.csv')
lap_times = pd.read_csv('lap_times.csv')
qualifying = pd.read_csv('qualifying.csv')
seasons = pd.read_csv('seasons.csv')
constructor_results = pd.read_csv('constructor_results.csv')
constructor_standings = pd.read_csv('constructor_standings.csv')
driver_standings = pd.read_csv('driver_standings.csv')
status = pd.read_csv('status.csv')

# Imprimir las primeras filas de cada DataFrame para confirmar que
```

```
todo se cargó correctamente
print("Drivers DataFrame:")
print(drivers.head(), "\n")

print("Constructors DataFrame:")
print(constructors.head(), "\n")

print("Circuits DataFrame:")
print(circuits.head(), "\n")

print("Races DataFrame:")
print(races.head(), "\n")

print("Results DataFrame:")
print(results.head(), "\n")

print("Pit Stops DataFrame:")
print(pit_stops.head(), "\n")

print("Lap Times DataFrame:")
print(lap_times.head(), "\n")

print("Qualifying DataFrame:")
print(qualifying.head(), "\n")

print("Seasons DataFrame:")
print(seasons.head(), "\n")

print("Constructor Results DataFrame:")
print(constructor_results.head(), "\n")

print("Constructor Standings DataFrame:")
print(constructor_standings.head(), "\n")

print("Driver Standings DataFrame:")
print(driver_standings.head(), "\n")

print("Status DataFrame:")
print(status.head(), "\n")
```

```
Drivers DataFrame:
   driverId   driverRef number code  forename     surname         dob
\
0         1    hamilton     44  HAM     Lewis    Hamilton  1985-01-07

1         2    heidfeld     \N  HEI      Nick    Heidfeld  1977-05-10

2         3     rosberg      6  ROS      Nico     Rosberg  1985-06-27

3         4      alonso     14  ALO  Fernando      Alonso  1981-07-29
```

```
4         5  kovalainen     \N  KOV    Heikki  Kovalainen  1981-10-19


  nationality                                                 url
0     British     http://en.wikipedia.org/wiki/Lewis_Hamilton
1      German      http://en.wikipedia.org/wiki/Nick_Heidfeld
2      German       http://en.wikipedia.org/wiki/Nico_Rosberg
3     Spanish    http://en.wikipedia.org/wiki/Fernando_Alonso
4     Finnish  http://en.wikipedia.org/wiki/Heikki_Kovalainen

Constructors DataFrame:
   constructorId constructorRef        name nationality  \
0              1        mclaren     McLaren     British
1              2     bmw_sauber  BMW Sauber      German
2              3       williams    Williams     British
3              4        renault     Renault      French
4              5     toro_rosso  Toro Rosso     Italian

                                             url
0               http://en.wikipedia.org/wiki/McLaren
1            http://en.wikipedia.org/wiki/BMW_Sauber
2  http://en.wikipedia.org/wiki/Williams_Grand_Pr...
3  http://en.wikipedia.org/wiki/Renault_in_Formul...
4   http://en.wikipedia.org/wiki/Scuderia_Toro_Rosso

Circuits DataFrame:
   circuitId   circuitRef                            name
location  \
0          1  albert_park  Albert Park Grand Prix Circuit
Melbourne
1          2       sepang    Sepang International Circuit  Kuala
Lumpur
2          3      bahrain   Bahrain International Circuit
Sakhir
3          4    catalunya  Circuit de Barcelona-Catalunya
Montmeló
4          5     istanbul                   Istanbul Park
Istanbul

     country       lat        lng  alt  \
0  Australia -37.84970  144.96800   10
1   Malaysia   2.76083  101.73800   18
2    Bahrain  26.03250   50.51060    7
3      Spain  41.57000    2.26111  109
4     Turkey  40.95170   29.40500  130

                                               url
0  http://en.wikipedia.org/wiki/Melbourne_Grand_P...
1  http://en.wikipedia.org/wiki/Sepang_Internatio...
2  http://en.wikipedia.org/wiki/Bahrain_Internati...
```

```
3  http://en.wikipedia.org/wiki/Circuit_de_Barcel...
4         http://en.wikipedia.org/wiki/Istanbul_Park

Races DataFrame:
   raceId  year  round  circuitId                  name
date  \
0       1  2009      1          1  Australian Grand Prix  2009-03-29

1       2  2009      2          2   Malaysian Grand Prix  2009-04-05

2       3  2009      3         17     Chinese Grand Prix  2009-04-19

3       4  2009      4          3     Bahrain Grand Prix  2009-04-26

4       5  2009      5          4     Spanish Grand Prix  2009-05-10


       time                                                  url
fp1_date  \
0  06:00:00  http://en.wikipedia.org/wiki/2009_Australian_G...       \
N
1  09:00:00  http://en.wikipedia.org/wiki/2009_Malaysian_Gr...       \
N
2  07:00:00  http://en.wikipedia.org/wiki/2009_Chinese_Gran...       \
N
3  12:00:00  http://en.wikipedia.org/wiki/2009_Bahrain_Gran...       \
N
4  12:00:00  http://en.wikipedia.org/wiki/2009_Spanish_Gran...       \
N

  fp1_time fp2_date fp2_time fp3_date fp3_time quali_date
quali_time  \
0       \N       \N       \N       \N       \N         \N         \N

1       \N       \N       \N       \N       \N         \N         \N

2       \N       \N       \N       \N       \N         \N         \N

3       \N       \N       \N       \N       \N         \N         \N

4       \N       \N       \N       \N       \N         \N         \N


  sprint_date sprint_time
0          \N          \N
1          \N          \N
2          \N          \N
3          \N          \N
4          \N          \N

Results DataFrame:
```

```
   resultId  raceId  driverId  constructorId number  grid position  \
0         1      18         1              1     22     1        1
1         2      18         2              2      3     5        2
2         3      18         3              3      7     7        3
3         4      18         4              4      5    11        4
4         5      18         5              1     23     3        5

  positionText  positionOrder  points  laps          time milliseconds
\
0            1              1    10.0    58  1:34:50.616      5690616

1            2              2     8.0    58       +5.478      5696094

2            3              3     6.0    58       +8.163      5698779

3            4              4     5.0    58      +17.181      5707797

4            5              5     4.0    58      +18.014      5708630


  fastestLap rank fastestLapTime fastestLapSpeed  statusId
0         39    2       1:27.452         218.300         1
1         41    3       1:27.739         217.586         1
2         41    5       1:28.090         216.719         1
3         58    7       1:28.603         215.464         1
4         43    1       1:27.418         218.385         1

Pit Stops DataFrame:
   raceId  driverId  stop  lap      time duration  milliseconds
0     841       153     1    1  17:05:23   26.898         26898
1     841        30     1    1  17:05:52   25.021         25021
2     841        17     1   11  17:20:48   23.426         23426
3     841         4     1   12  17:22:34   23.251         23251
4     841        13     1   13  17:24:10   23.842         23842

Lap Times DataFrame:
   raceId  driverId  lap  position      time  milliseconds
0     841        20    1         1  1:38.109         98109
1     841        20    2         1  1:33.006         93006
2     841        20    3         1  1:32.713         92713
3     841        20    4         1  1:32.803         92803
4     841        20    5         1  1:32.342         92342

Qualifying DataFrame:
   qualifyId  raceId  driverId  constructorId  number  position
q1  \
0          1      18         1              1      22         1
1:26.572
1          2      18         9              2       4         2
1:26.103
```

```
2          3      18          5               1      23          3
1:25.664
3          4      18         13               6       2          4
1:25.994
4          5      18          2               2       3          5
1:25.960

          q2         q3
0  1:25.187  1:26.714
1  1:25.315  1:26.869
2  1:25.452  1:27.079
3  1:25.691  1:27.178
4  1:25.518  1:27.236

Seasons DataFrame:
   year                                                url
0  2009  http://en.wikipedia.org/wiki/2009_Formula_One_...
1  2008  http://en.wikipedia.org/wiki/2008_Formula_One_...
2  2007  http://en.wikipedia.org/wiki/2007_Formula_One_...
3  2006  http://en.wikipedia.org/wiki/2006_Formula_One_...
4  2005  http://en.wikipedia.org/wiki/2005_Formula_One_...

Constructor Results DataFrame:
   constructorResultsId  raceId  constructorId  points status
0                     1      18              1    14.0     \N
1                     2      18              2     8.0     \N
2                     3      18              3     9.0     \N
3                     4      18              4     5.0     \N
4                     5      18              5     2.0     \N

Constructor Standings DataFrame:
   constructorStandingsId  raceId  constructorId  points  position  \
0                       1      18              1    14.0         1
1                       2      18              2     8.0         3
2                       3      18              3     9.0         2
3                       4      18              4     5.0         4
4                       5      18              5     2.0         5

  positionText  wins
0            1     1
1            3     0
2            2     0
3            4     0
4            5     0

Driver Standings DataFrame:
   driverStandingsId  raceId  driverId  points  position positionText
wins
0                  1      18         1    10.0         1            1
1
```

```
1                  2      18          2     8.0          2              2
0
2                  3      18          3     6.0          3              3
0
3                  4      18          4     5.0          4              4
0
4                  5      18          5     4.0          5              5
0

Status DataFrame:
   statusId        status
0         1      Finished
1         2  Disqualified
2         3      Accident
3         4     Collision
4         5        Engine
```

#EDA

El objetivo del EDA es observar los datos, para ver si se necesitan limpiar y o hacer transformaciones para más adelante poder realizar los ánalisis. Para ello se han observado, los valores nulos, las distribuciones y los correlogramas de las tablas más importantes.

```
# Función para mostrar un resumen estadístico de un DataFrame
def eda_summary(df, name):
    print(f"--- Resumen de la Tabla: {name} ---")
    print("Primeras 5 filas:")
    print(df.head())
    print("\nInformación del DataFrame:")
    print(df.info())
    print("\nResumen Estadístico:")
    print(df.describe(include='all'))
    print("\nValores Nulos:")
    print(df.isnull().sum())
    print("\n" + "-"*50 + "\n")

# Resumen de las tablas importantes
eda_summary(drivers, "drivers")
eda_summary(constructors, "constructors")
eda_summary(circuits, "circuits")
eda_summary(races, "races")
eda_summary(results, "results")
eda_summary(driver_standings, "driver_standings")
eda_summary(constructor_standings, "constructor_standings")

--- Resumen de la Tabla: drivers ---
Primeras 5 filas:
   driverId   driverRef number code  forename     surname         dob
nationality  \
```

```
0         1    hamilton     44  HAM     Lewis    Hamilton  1985-01-07
British
1         2    heidfeld     \N  HEI      Nick    Heidfeld  1977-05-10
German
2         3     rosberg      6  ROS      Nico     Rosberg  1985-06-27
German
3         4      alonso     14  ALO  Fernando      Alonso  1981-07-29
Spanish
4         5  kovalainen     \N  KOV    Heikki  Kovalainen  1981-10-19
Finnish

                                                  url
0     http://en.wikipedia.org/wiki/Lewis_Hamilton
1      http://en.wikipedia.org/wiki/Nick_Heidfeld
2       http://en.wikipedia.org/wiki/Nico_Rosberg
3    http://en.wikipedia.org/wiki/Fernando_Alonso
4  http://en.wikipedia.org/wiki/Heikki_Kovalainen

Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 859 entries, 0 to 858
Data columns (total 9 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   driverId     859 non-null    int64
 1   driverRef    859 non-null    object
 2   number       859 non-null    object
 3   code         859 non-null    object
 4   forename     859 non-null    object
 5   surname      859 non-null    object
 6   dob          859 non-null    object
 7   nationality  859 non-null    object
 8   url          859 non-null    object
dtypes: int64(1), object(8)
memory usage: 60.5+ KB
None

Resumen Estadístico:
        driverId driverRef number code forename surname         dob
nationality  \
count     859.00       859    859  859      859     859         859
859
unique       NaN       859     47   97      478     800         841
42
top          NaN  hamilton     \N   \N     John  Taylor  1918-10-06
British
freq         NaN         1    802  757       14       5           2
166
mean      430.06       NaN    NaN  NaN      NaN     NaN         NaN
NaN
```

```
std        248.21        NaN    NaN  NaN       NaN     NaN          NaN
NaN
min          1.00        NaN    NaN  NaN       NaN     NaN          NaN
NaN
25%        215.50        NaN    NaN  NaN       NaN     NaN          NaN
NaN
50%        430.00        NaN    NaN  NaN       NaN     NaN          NaN
NaN
75%        644.50        NaN    NaN  NaN       NaN     NaN          NaN
NaN
max        860.00        NaN    NaN  NaN       NaN     NaN          NaN
NaN

                                                    url
count                                               859
unique                                              859
top     http://en.wikipedia.org/wiki/Lewis_Hamilton
freq                                                  1
mean                                                NaN
std                                                 NaN
min                                                 NaN
25%                                                 NaN
50%                                                 NaN
75%                                                 NaN
max                                                 NaN

Valores Nulos:
driverId       0
driverRef      0
number         0
code           0
forename       0
surname        0
dob            0
nationality    0
url            0
dtype: int64

--------------------------------------------------

--- Resumen de la Tabla: constructors ---
Primeras 5 filas:
   constructorId constructorRef        name nationality  \
0              1        mclaren     McLaren     British
1              2     bmw_sauber  BMW Sauber      German
2              3       williams    Williams     British
3              4        renault     Renault      French
4              5     toro_rosso  Toro Rosso     Italian

                                                    url
```

```
0                          http://en.wikipedia.org/wiki/McLaren
1                       http://en.wikipedia.org/wiki/BMW_Sauber
2  http://en.wikipedia.org/wiki/Williams_Grand_Prix_Engineering
3           http://en.wikipedia.org/wiki/Renault_in_Formula_One
4              http://en.wikipedia.org/wiki/Scuderia_Toro_Rosso

Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 212 entries, 0 to 211
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   constructorId   212 non-null    int64
 1   constructorRef  212 non-null    object
 2   name            212 non-null    object
 3   nationality     212 non-null    object
 4   url             212 non-null    object
dtypes: int64(1), object(4)
memory usage: 8.4+ KB
None

Resumen Estadístico:
        constructorId constructorRef     name nationality  \
count          212.00            212      212         212
unique            NaN            212      212          24
top               NaN        mclaren  McLaren     British
freq              NaN              1        1          86
mean           107.55            NaN      NaN         NaN
std             61.95            NaN      NaN         NaN
min              1.00            NaN      NaN         NaN
25%             54.75            NaN      NaN         NaN
50%            107.50            NaN      NaN         NaN
75%            160.25            NaN      NaN         NaN
max            215.00            NaN      NaN         NaN

                                                    url
count                                               212
unique                                              175
top     http://en.wikipedia.org/wiki/Cooper_Car_Company
freq                                                 11
mean                                                NaN
std                                                 NaN
min                                                 NaN
25%                                                 NaN
50%                                                 NaN
75%                                                 NaN
max                                                 NaN

Valores Nulos:
constructorId     0
```

```
constructorRef    0
name              0
nationality       0
url               0
dtype: int64

--------------------------------------------------

--- Resumen de la Tabla: circuits ---
Primeras 5 filas:
   circuitId   circuitRef                                name
location    country    lat    lng  \
0          1  albert_park  Albert Park Grand Prix Circuit
Melbourne  Australia -37.85 144.97
1          2       sepang    Sepang International Circuit  Kuala
Lumpur   Malaysia   2.76 101.74
2          3      bahrain   Bahrain International Circuit
Sakhir    Bahrain  26.03  50.51
3          4    catalunya  Circuit de Barcelona-Catalunya
Montmeló      Spain  41.57   2.26
4          5     istanbul                   Istanbul Park
Istanbul     Turkey  40.95  29.41

   alt                                                      url
Length  Turns
0   10    http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit
5.28  14.00
1   18    http://en.wikipedia.org/wiki/Sepang_International_Circuit
5.54  15.00
2    7   http://en.wikipedia.org/wiki/Bahrain_International_Circuit
5.41  15.00
3  109  http://en.wikipedia.org/wiki/Circuit_de_Barcelona-Catalunya
4.66  14.00
4  130                   http://en.wikipedia.org/wiki/Istanbul_Park
5.34  14.00

Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77 entries, 0 to 76
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   circuitId   77 non-null     int64
 1   circuitRef  77 non-null     object
 2   name        77 non-null     object
 3   location    77 non-null     object
 4   country     77 non-null     object
 5   lat         77 non-null     float64
 6   lng         77 non-null     float64
 7   alt         77 non-null     int64
```

```
 8   url         77 non-null     object
 9   Length      77 non-null     float64
 10  Turns       77 non-null     float64
dtypes: float64(4), int64(2), object(5)
memory usage: 6.7+ KB
None

Resumen Estadístico:
        circuitId   circuitRef                              name
location country    lat     lng  \
count       77.00           77                              77
77      77  77.00   77.00
unique        NaN           77                              77
75      35    NaN     NaN
top           NaN  albert_park  Albert Park Grand Prix Circuit
Barcelona     USA    NaN     NaN
freq          NaN            1                               1
2      11    NaN     NaN
mean        39.88          NaN                             NaN
NaN     NaN  33.44    1.08
std         23.00          NaN                             NaN
NaN     NaN  22.81   65.52
min          1.00          NaN                             NaN
NaN     NaN -37.85 -118.19
25%         20.00          NaN                             NaN
NaN     NaN  32.78   -9.39
50%         40.00          NaN                             NaN
NaN     NaN  40.95    3.93
75%         59.00          NaN                             NaN
NaN     NaN  46.96   19.25
max         80.00          NaN                             NaN
NaN     NaN  57.27  144.97

           alt
url  Length  Turns
count    77.00
77   77.00  77.00
unique     NaN
77     NaN    NaN
top        NaN
http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit     NaN
NaN
freq       NaN
1     NaN    NaN
mean    247.01
NaN    4.99  14.60
std      362.74
NaN    2.88   5.03
min      -7.00
```

```
NaN    1.65   4.00
25%      18.00
NaN    3.85  12.00
50%     129.00
NaN    4.56  14.00
75%     332.00
NaN    5.42  17.00
max    2227.00
NaN   25.80  38.00

Valores Nulos:
circuitId     0
circuitRef    0
name          0
location      0
country       0
lat           0
lng           0
alt           0
url           0
Length        0
Turns         0
dtype: int64

--------------------------------------------------

--- Resumen de la Tabla: races ---
Primeras 5 filas:
   raceId  year  round  circuitId                  name        date
time  \
0       1  2009      1          1  Australian Grand Prix  2009-03-29
06:00:00
1       2  2009      2          2   Malaysian Grand Prix  2009-04-05
09:00:00
2       3  2009      3         17     Chinese Grand Prix  2009-04-19
07:00:00
3       4  2009      4          3     Bahrain Grand Prix  2009-04-26
12:00:00
4       5  2009      5          4     Spanish Grand Prix  2009-05-10
12:00:00

                                                  url fp1_date
fp1_time fp2_date fp2_time  \
0  http://en.wikipedia.org/wiki/2009_Australian_Grand_Prix       \N
\N       \N       \N
1   http://en.wikipedia.org/wiki/2009_Malaysian_Grand_Prix       \N
\N       \N       \N
2     http://en.wikipedia.org/wiki/2009_Chinese_Grand_Prix       \N
\N       \N       \N
3     http://en.wikipedia.org/wiki/2009_Bahrain_Grand_Prix       \N
```

```
\N       \N       \N
4     http://en.wikipedia.org/wiki/2009_Spanish_Grand_Prix       \N
\N       \N       \N

  fp3_date fp3_time quali_date quali_time sprint_date sprint_time
0       \N       \N         \N         \N          \N          \N
1       \N       \N         \N         \N          \N          \N
2       \N       \N         \N         \N          \N          \N
3       \N       \N         \N         \N          \N          \N
4       \N       \N         \N         \N          \N          \N

Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1125 entries, 0 to 1124
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   raceId       1125 non-null   int64
 1   year         1125 non-null   int64
 2   round        1125 non-null   int64
 3   circuitId    1125 non-null   int64
 4   name         1125 non-null   object
 5   date         1125 non-null   object
 6   time         1125 non-null   object
 7   url          1125 non-null   object
 8   fp1_date     1125 non-null   object
 9   fp1_time     1125 non-null   object
 10  fp2_date     1125 non-null   object
 11  fp2_time     1125 non-null   object
 12  fp3_date     1125 non-null   object
 13  fp3_time     1125 non-null   object
 14  quali_date   1125 non-null   object
 15  quali_time   1125 non-null   object
 16  sprint_date  1125 non-null   object
 17  sprint_time  1125 non-null   object
dtypes: int64(4), object(14)
memory usage: 158.3+ KB
None

Resumen Estadístico:
        raceId    year   round  circuitId                name
date  time  \
count  1125.00 1125.00 1125.00    1125.00                1125
1125  1125
unique     NaN     NaN     NaN        NaN                  54
1125    35
top        NaN     NaN     NaN        NaN  Italian Grand Prix  2009-
03-29    \N
freq       NaN     NaN     NaN        NaN                  75
1   731
```

```
mean    565.71 1992.70    8.58      23.89                  NaN
NaN   NaN
std     328.81   20.60    5.16      19.63                  NaN
NaN   NaN
min       1.00 1950.00    1.00       1.00                  NaN
NaN   NaN
25%     282.00 1977.00    4.00       9.00                  NaN
NaN   NaN
50%     563.00 1994.00    8.00      18.00                  NaN
NaN   NaN
75%     845.00 2011.00   13.00      34.00                  NaN
NaN   NaN
max    1144.00 2024.00   24.00      80.00                  NaN
NaN   NaN

                                                          url
fp1_date fp1_time fp2_date  \
count                                                    1125
1125     1125     1125
unique                                                   1125
91       21       91
top
http://en.wikipedia.org/wiki/2009_Australian_Grand_Prix       \N
\N       \N
freq                                                       1
1035     1057     1035
mean                                                      NaN
NaN      NaN      NaN
std                                                       NaN
NaN      NaN      NaN
min                                                       NaN
NaN      NaN      NaN
25%                                                       NaN
NaN      NaN      NaN
50%                                                       NaN
NaN      NaN      NaN
75%                                                       NaN
NaN      NaN      NaN
max                                                       NaN
NaN      NaN      NaN

       fp2_time fp3_date fp3_time quali_date quali_time sprint_date
sprint_time
count      1125     1125     1125       1125       1125        1125
1125
unique       20       73       19         91         16          19
13
top          \N       \N       \N         \N         \N          \N
\N
```

```
freq        1057     1053     1072        1035        1057         1107
1110
mean         NaN      NaN      NaN         NaN         NaN          NaN
NaN
std          NaN      NaN      NaN         NaN         NaN          NaN
NaN
min          NaN      NaN      NaN         NaN         NaN          NaN
NaN
25%          NaN      NaN      NaN         NaN         NaN          NaN
NaN
50%          NaN      NaN      NaN         NaN         NaN          NaN
NaN
75%          NaN      NaN      NaN         NaN         NaN          NaN
NaN
max          NaN      NaN      NaN         NaN         NaN          NaN
NaN

Valores Nulos:
raceId         0
year           0
round          0
circuitId      0
name           0
date           0
time           0
url            0
fp1_date       0
fp1_time       0
fp2_date       0
fp2_time       0
fp3_date       0
fp3_time       0
quali_date     0
quali_time     0
sprint_date    0
sprint_time    0
dtype: int64

--------------------------------------------------

--- Resumen de la Tabla: results ---
Primeras 5 filas:
   resultId  raceId  driverId  constructorId number  grid position
positionText  positionOrder  \
0         1      18         1              1     22     1        1
1             1
1         2      18         2              2      3     5        2
2             2
2         3      18         3              3      7     7        3
3             3
```

```
3         4      18         4               4      5    11         4
4             4
4         5      18         5               1     23     3         5
5             5

   points  laps         time milliseconds fastestLap rank
fastestLapTime fastestLapSpeed  statusId
0   10.00    58  1:34:50.616      5690616         39    2
1:27.452         218.300         1
1    8.00    58       +5.478      5696094         41    3
1:27.739         217.586         1
2    6.00    58       +8.163      5698779         41    5
1:28.090         216.719         1
3    5.00    58      +17.181      5707797         58    7
1:28.603         215.464         1
4    4.00    58      +18.014      5708630         43    1
1:27.418         218.385         1

Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26519 entries, 0 to 26518
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   resultId         26519 non-null  int64
 1   raceId           26519 non-null  int64
 2   driverId         26519 non-null  int64
 3   constructorId    26519 non-null  int64
 4   number           26519 non-null  object
 5   grid             26519 non-null  int64
 6   position         26519 non-null  object
 7   positionText     26519 non-null  object
 8   positionOrder    26519 non-null  int64
 9   points           26519 non-null  float64
 10  laps             26519 non-null  int64
 11  time             26519 non-null  object
 12  milliseconds     26519 non-null  object
 13  fastestLap       26519 non-null  object
 14  rank             26519 non-null  object
 15  fastestLapTime   26519 non-null  object
 16  fastestLapSpeed  26519 non-null  object
 17  statusId         26519 non-null  int64
dtypes: float64(1), int64(8), object(9)
memory usage: 3.6+ MB
None

Resumen Estadístico:
        resultId   raceId  driverId  constructorId number     grid
position positionText  \
count   26519.00 26519.00  26519.00       26519.00  26519 26519.00
```

```
26519         26519
unique          NaN      NaN        NaN             NaN     130       NaN
34            39
top             NaN      NaN        NaN             NaN       4       NaN
\N             R
freq            NaN      NaN        NaN             NaN    1007       NaN
10928          8876
mean       13260.94   546.38     274.36           49.80     NaN     11.15
NaN           NaN
std         7656.81   309.64     279.28           61.09     NaN      7.21
NaN           NaN
min            1.00     1.00       1.00            1.00     NaN      0.00
NaN           NaN
25%         6630.50   298.00      57.00            6.00     NaN      5.00
NaN           NaN
50%        13260.00   527.00     170.00           25.00     NaN     11.00
NaN           NaN
75%        19889.50   803.00     385.00           60.00     NaN     17.00
NaN           NaN
max        26524.00  1132.00     860.00          215.00     NaN     34.00
NaN           NaN

           positionOrder   points     laps   time milliseconds fastestLap
rank fastestLapTime  \
count           26519.00 26519.00 26519.00  26519        26519      26519
26519          26519
unique             NaN      NaN      NaN   7272         7493         81
26            7298
top                NaN      NaN      NaN     \N           \N         \N
\N              \N
freq               NaN      NaN      NaN  18986        18986      18499
18249          18499
mean             12.81     1.96    46.23    NaN          NaN        NaN
NaN            NaN
std                7.68     4.31    29.58    NaN          NaN        NaN
NaN            NaN
min                1.00     0.00     0.00    NaN          NaN        NaN
NaN            NaN
25%                6.00     0.00    23.00    NaN          NaN        NaN
NaN            NaN
50%               12.00     0.00    53.00    NaN          NaN        NaN
NaN            NaN
75%               18.00     2.00    66.00    NaN          NaN        NaN
NaN            NaN
max               39.00    50.00   200.00    NaN          NaN        NaN
NaN            NaN

          fastestLapSpeed  statusId
count               26519  26519.00
```

```
unique             7514       NaN
top                  \N       NaN
freq              18499       NaN
mean                NaN     17.32
std                 NaN     26.08
min                 NaN      1.00
25%                 NaN      1.00
50%                 NaN     10.00
75%                 NaN     14.00
max                 NaN    141.00

Valores Nulos:
resultId            0
raceId              0
driverId            0
constructorId       0
number              0
grid                0
position            0
positionText        0
positionOrder       0
points              0
laps                0
time                0
milliseconds        0
fastestLap          0
rank                0
fastestLapTime      0
fastestLapSpeed     0
statusId            0
dtype: int64

--------------------------------------------------

--- Resumen de la Tabla: driver_standings ---
Primeras 5 filas:
   driverStandingsId  raceId  driverId  points  position positionText
wins
0                  1      18         1   10.00         1            1
1
1                  2      18         2    8.00         2            2
0
2                  3      18         3    6.00         3            3
0
3                  4      18         4    5.00         4            4
0
4                  5      18         5    4.00         5            5
0

Información del DataFrame:
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34595 entries, 0 to 34594
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   driverStandingsId  34595 non-null  int64  
 1   raceId             34595 non-null  int64  
 2   driverId           34595 non-null  int64  
 3   points             34595 non-null  float64
 4   position           34595 non-null  int64  
 5   positionText       34595 non-null  object 
 6   wins               34595 non-null  int64  
dtypes: float64(1), int64(5), object(1)
memory usage: 1.8+ MB
None

Resumen Estadístico:
        driverStandingsId   raceId  driverId   points  position
positionText     wins
count            34595.00 34595.00  34595.00 34595.00  34595.00
34595 34595.00
unique                NaN      NaN       NaN      NaN       NaN
109      NaN
top                   NaN      NaN       NaN      NaN       NaN
1      NaN
freq                  NaN      NaN       NaN      NaN       NaN
1113      NaN
mean              42944.38   580.12    313.46    14.11     19.78
NaN     0.27
std               21859.82   289.29    272.05    37.24     16.33
NaN     1.02
min                   1.00     1.00      1.00     0.00      1.00
NaN     0.00
25%               19767.50   352.00     87.00     0.00      8.00
NaN     0.00
50%               49910.00   600.00    222.00     1.00     16.00
NaN     0.00
75%               59299.50   803.00    517.00    10.00     26.00
NaN     0.00
max               72871.00  1132.00    860.00   575.00    108.00
NaN    19.00

Valores Nulos:
driverStandingsId    0
raceId               0
driverId             0
points               0
position             0
positionText         0
```

```
wins                   0
dtype: int64

------------------------------------------------

--- Resumen de la Tabla: constructor_standings ---
Primeras 5 filas:
   constructorStandingsId  raceId  constructorId  points  position
positionText  wins
0                       1      18              1   14.00         1
1     1
1                       2      18              2    8.00         3
3     0
2                       3      18              3    9.00         2
2     0
3                       4      18              4    5.00         4
4     0
4                       5      18              5    2.00         5
5     0

Información del DataFrame:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13271 entries, 0 to 13270
Data columns (total 7 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   constructorStandingsId  13271 non-null  int64
 1   raceId                  13271 non-null  int64
 2   constructorId           13271 non-null  int64
 3   points                  13271 non-null  float64
 4   position                13271 non-null  int64
 5   positionText            13271 non-null  object
 6   wins                    13271 non-null  int64
dtypes: float64(1), int64(5), object(1)
memory usage: 725.9+ KB
None

Resumen Estadístico:
        constructorStandingsId   raceId  constructorId   points
position positionText     wins
count                 13271.00 13271.00       13271.00 13271.00
13271.00        13271 13271.00
unique                     NaN      NaN            NaN      NaN
NaN           23      NaN
top                        NaN      NaN            NaN      NaN
NaN            1      NaN
freq                       NaN      NaN            NaN      NaN
NaN         1049      NaN
mean                  16874.15   529.95          49.22    35.68
7.24          NaN     0.69
```

```
std                      8835.15   303.68          60.74    80.56
4.36          NaN     1.87
min                         1.00     1.00           1.00     0.00
1.00          NaN     0.00
25%                      8853.50   300.00           6.00     0.00
4.00          NaN     0.00
50%                     20458.00   504.00          25.00     7.00
7.00          NaN     0.00
75%                     24873.50   733.00          58.00    32.00
10.00          NaN     0.00
max                     28852.00  1132.00         215.00   860.00
22.00          NaN    21.00

Valores Nulos:
constructorStandingsId    0
raceId                    0
constructorId             0
points                    0
position                  0
positionText              0
wins                      0
dtype: int64

--------------------------------------------------
```

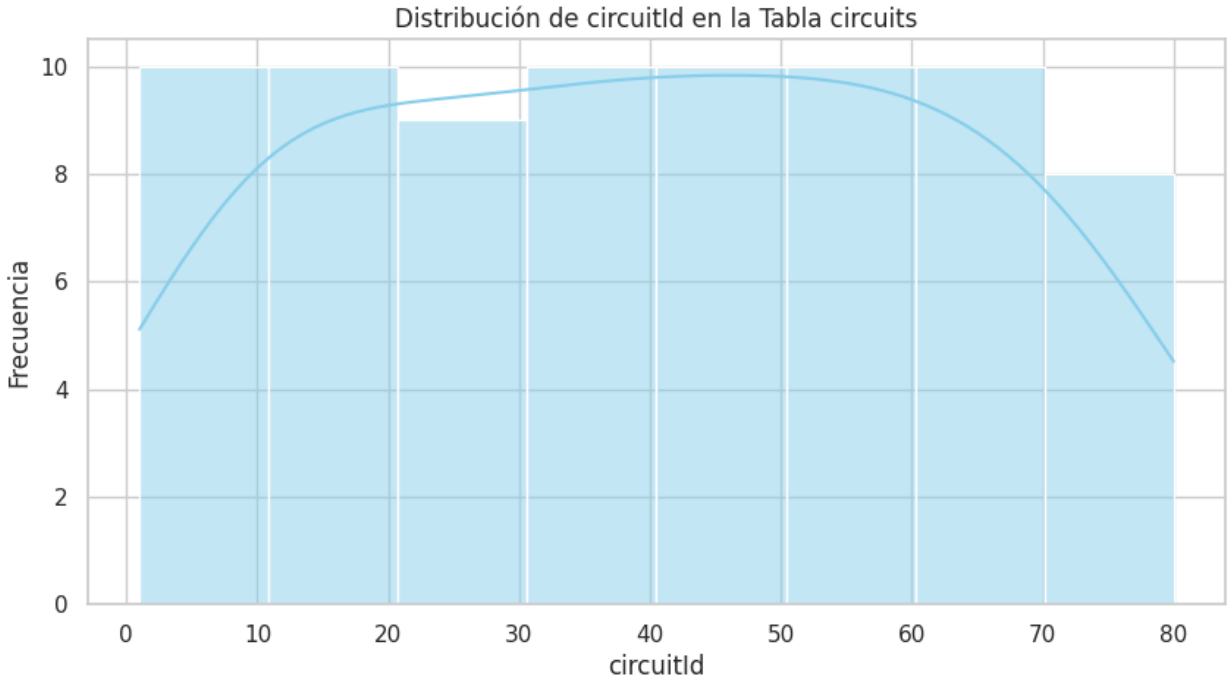
```python
# Función para mostrar valores nulos de cada tabla
def show_null_values(df, name):
    print(f"--- Valores Nulos en la Tabla: {name} ---")
    print(df.isnull().sum())
    print("\n" + "-"*50 + "\n")

# Mostrar valores nulos de cada tabla importante
show_null_values(drivers, "drivers")
show_null_values(constructors, "constructors")
show_null_values(circuits, "circuits")
show_null_values(races, "races")
show_null_values(results, "results")
show_null_values(driver_standings, "driver_standings")
show_null_values(constructor_standings, "constructor_standings")
```

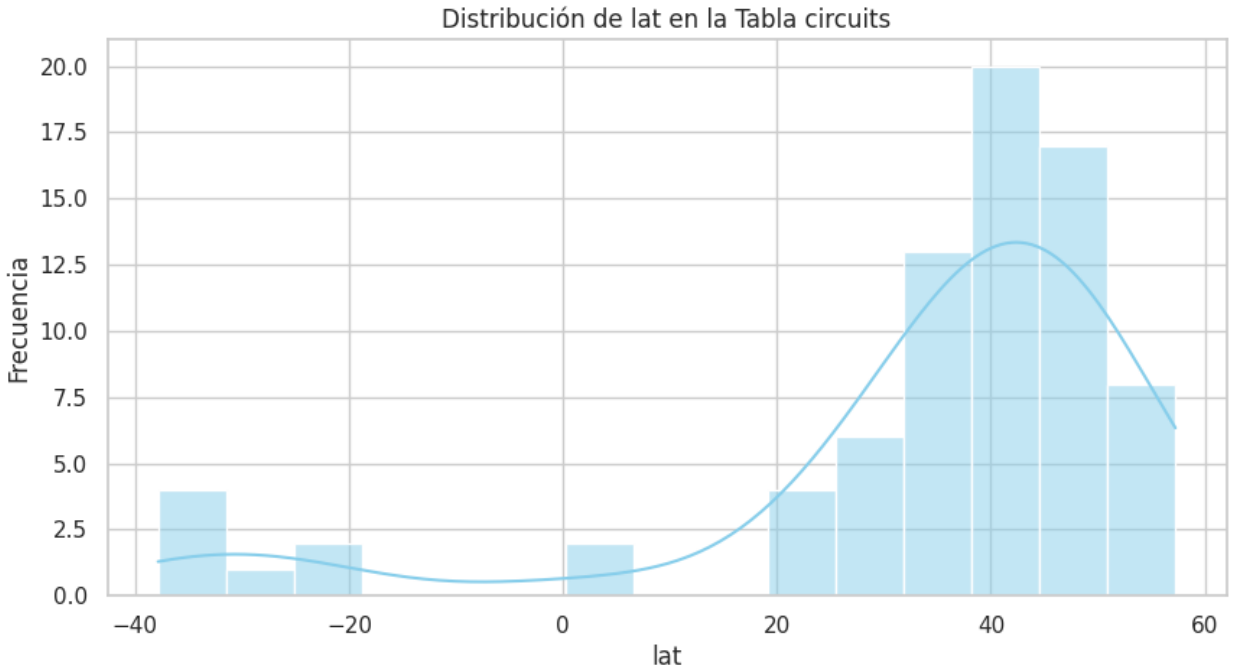
```
--- Valores Nulos en la Tabla: drivers ---
driverId       0
driverRef      0
number         0
code           0
forename       0
surname        0
dob            0
nationality    0
```

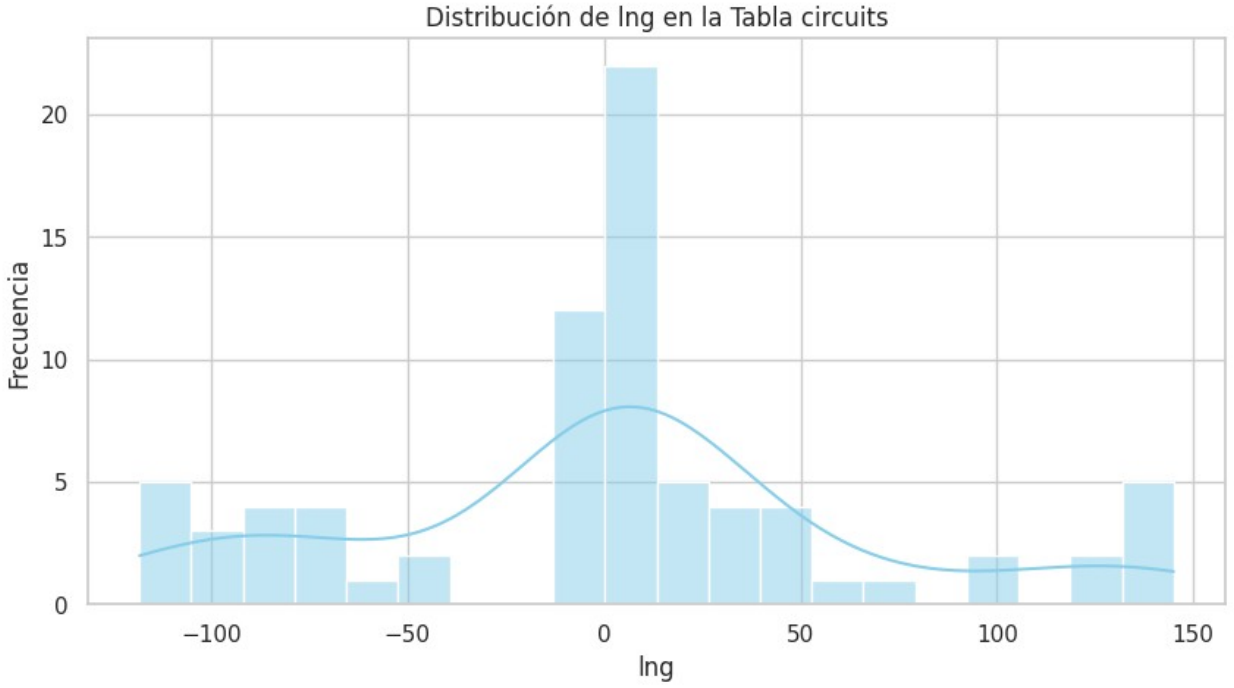
```
url             0
dtype: int64

--------------------------------------------------

--- Valores Nulos en la Tabla: constructors ---
constructorId     0
constructorRef    0
name              0
nationality       0
url               0
dtype: int64

--------------------------------------------------

--- Valores Nulos en la Tabla: circuits ---
circuitId     0
circuitRef    0
name          0
location      0
country       0
lat           0
lng           0
alt           0
url           0
Length        0
Turns         0
dtype: int64

--------------------------------------------------

--- Valores Nulos en la Tabla: races ---
raceId         0
year           0
round          0
circuitId      0
name           0
date           0
time           0
url            0
fp1_date       0
fp1_time       0
fp2_date       0
fp2_time       0
fp3_date       0
fp3_time       0
quali_date     0
quali_time     0
sprint_date    0
sprint_time    0
```

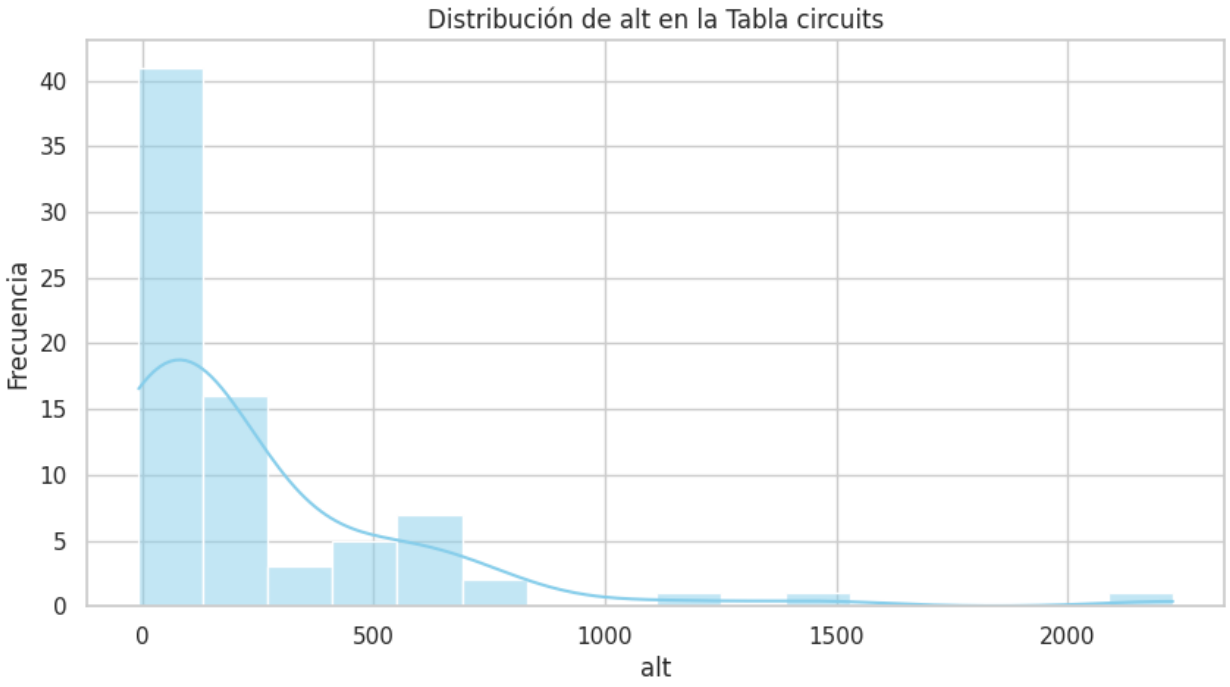
```
dtype: int64

--------------------------------------------------

--- Valores Nulos en la Tabla: results ---
resultId           0
raceId             0
driverId           0
constructorId      0
number             0
grid               0
position           0
positionText       0
positionOrder      0
points             0
laps               0
time               0
milliseconds       0
fastestLap         0
rank               0
fastestLapTime     0
fastestLapSpeed    0
statusId           0
dtype: int64

--------------------------------------------------

--- Valores Nulos en la Tabla: driver_standings ---
driverStandingsId    0
raceId               0
driverId             0
points               0
position             0
positionText         0
wins                 0
dtype: int64

--------------------------------------------------

--- Valores Nulos en la Tabla: constructor_standings ---
constructorStandingsId    0
raceId                    0
constructorId             0
points                    0
position                  0
positionText              0
wins                      0
dtype: int64
```

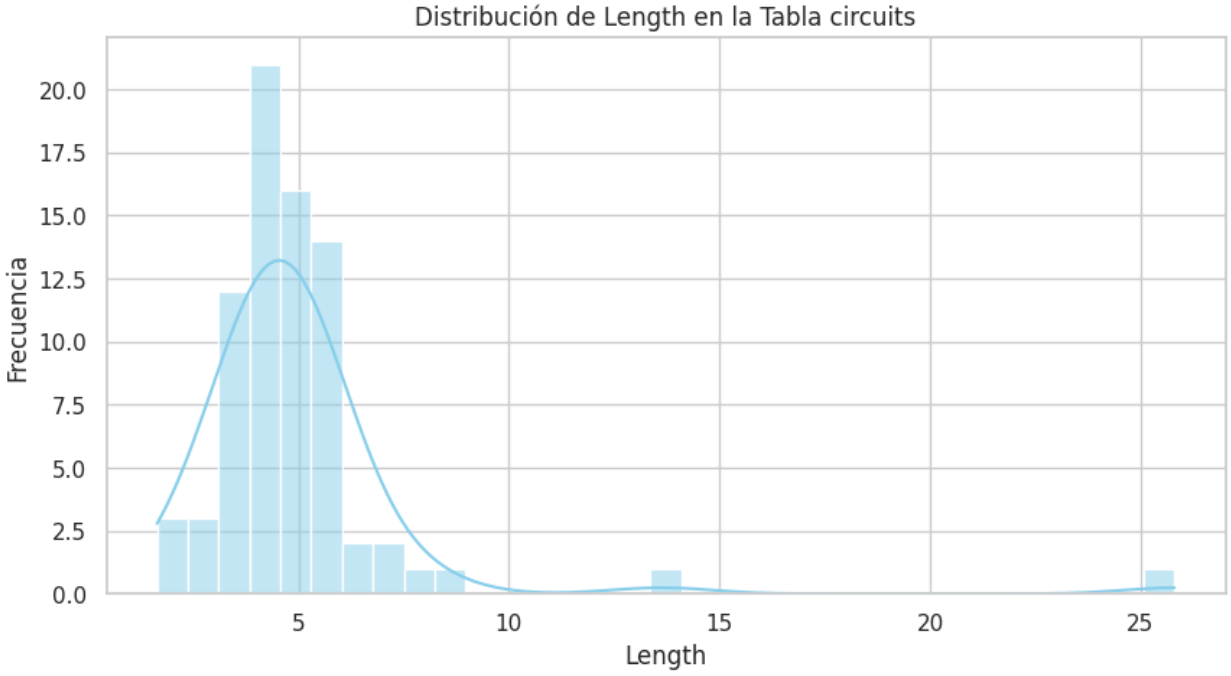
```
------------------------------------------------


import matplotlib.pyplot as plt
import seaborn as sns

# Función para graficar la distribución de variables numéricas de un
DataFrame
def plot_numeric_distributions(df, name):
    numeric_columns = df.select_dtypes(include=['float64',
'int64']).columns
    for col in numeric_columns:
        plt.figure(figsize=(10, 5))
        sns.histplot(df[col].dropna(), kde=True, color='skyblue')
        plt.title(f'Distribución de {col} en la Tabla {name}')
        plt.xlabel(col)
        plt.ylabel('Frecuencia')
        plt.grid(True)
        plt.show()

# Graficar distribuciones para tablas importantes
plot_numeric_distributions(circuits, "circuits")
plot_numeric_distributions(races, "races")
plot_numeric_distributions(results, "results")
plot_numeric_distributions(driver_standings, "driver_standings")
plot_numeric_distributions(constructor_standings,
"constructor_standings")
```

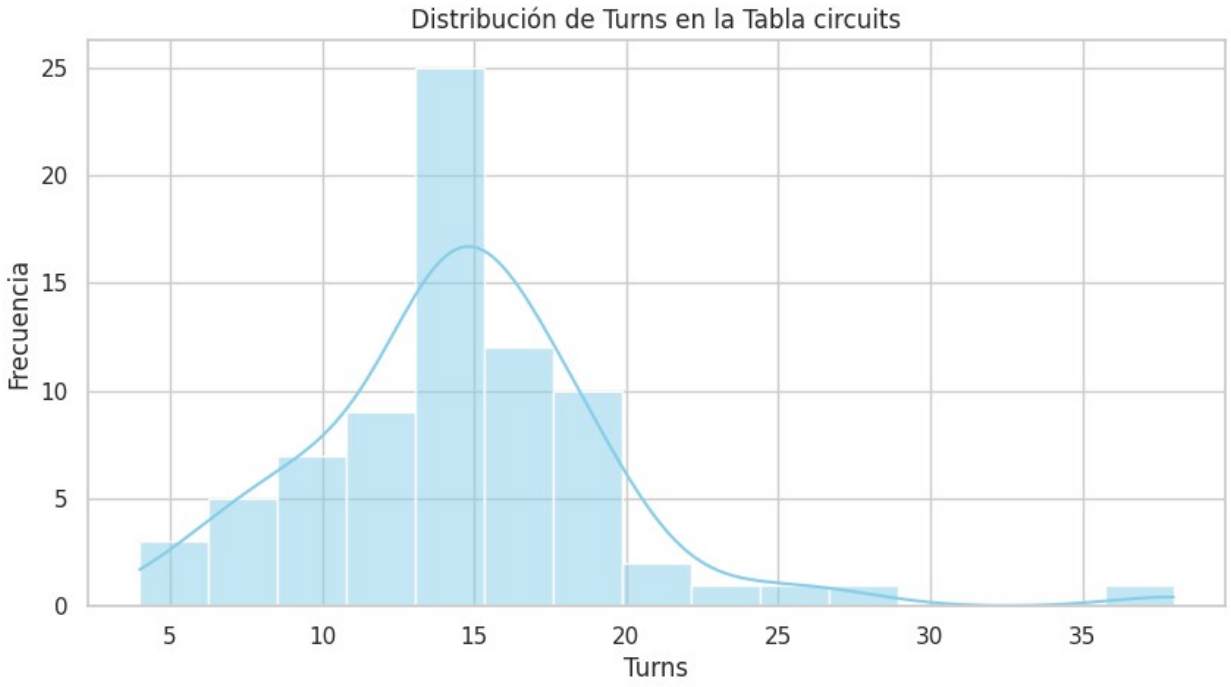


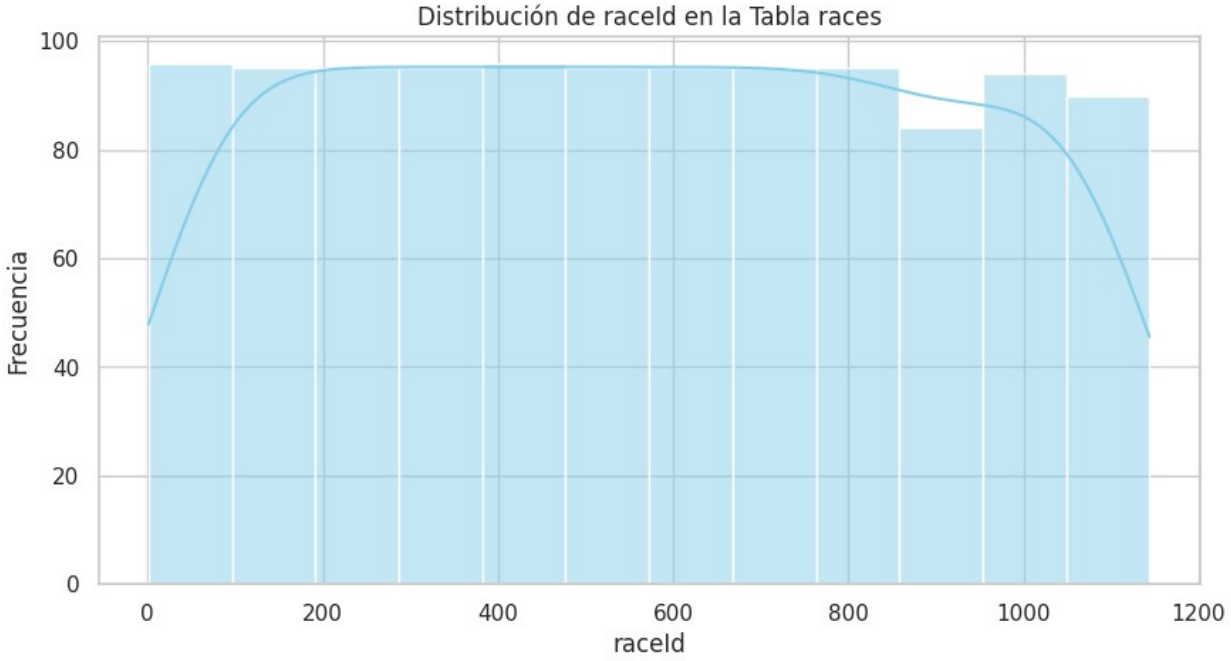

Distribución de lat en la Tabla circuits
Frecuencia
lat
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−40
−20
0
20
40
60

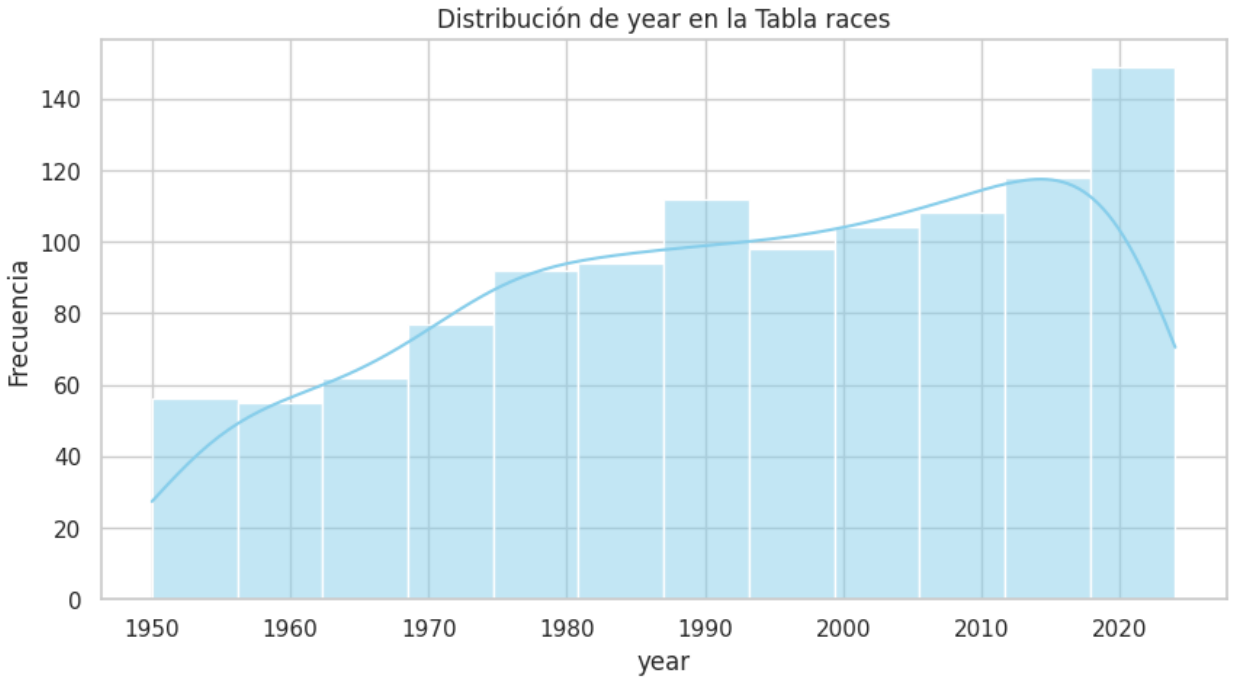

Distribución de lng en la Tabla circuits
Frecuencia
lng
20
15
10
5
0
−100
−50
0
50
100
150

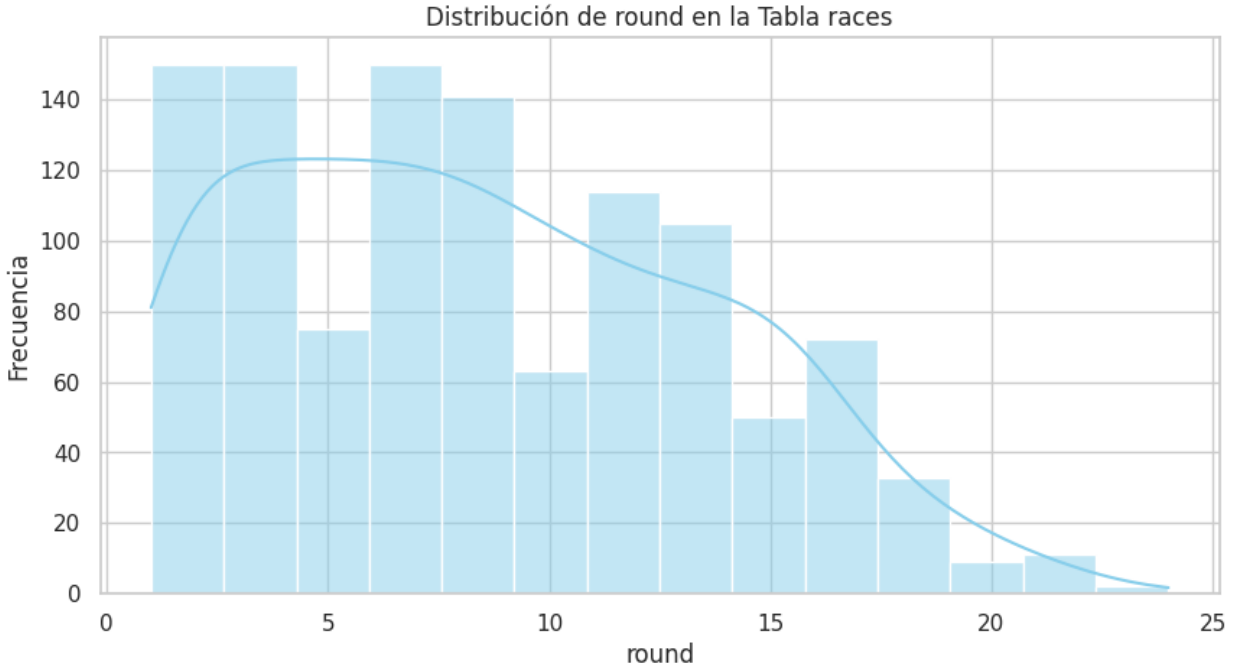

Distribución de alt en la Tabla circuits
Frecuencia
alt
0
5
10
15
20
25
30
35
40
0
500
1000
1500
2000

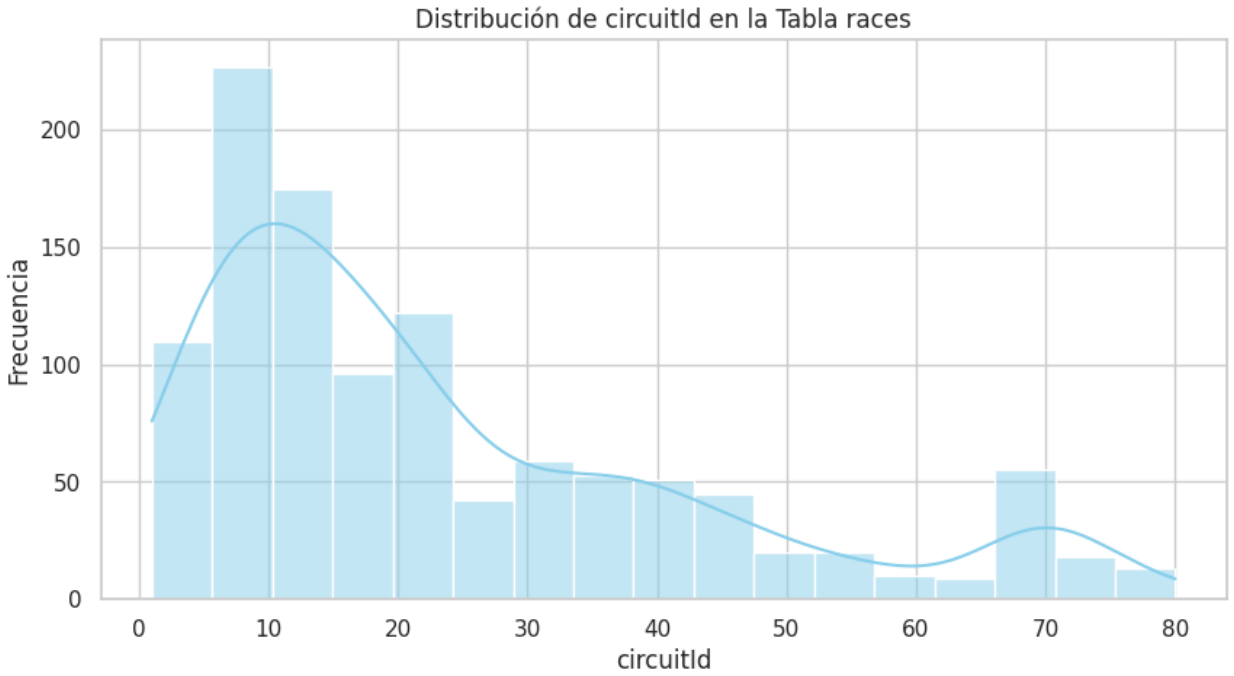

Distribución de Length en la Tabla circuits
Frecuencia
Length
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
5
10
15
20
25

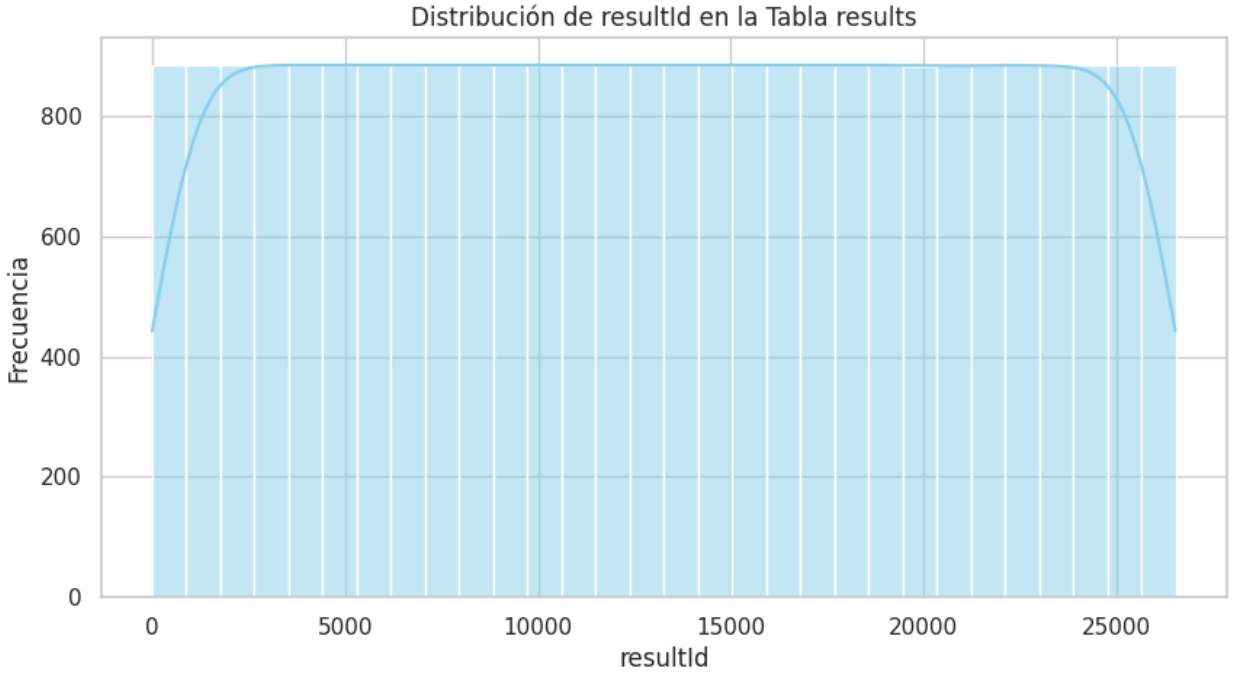

Distribución de Turns en la Tabla circuits
Frecuencia
Turns
0
5
10
15
20
25
30
35

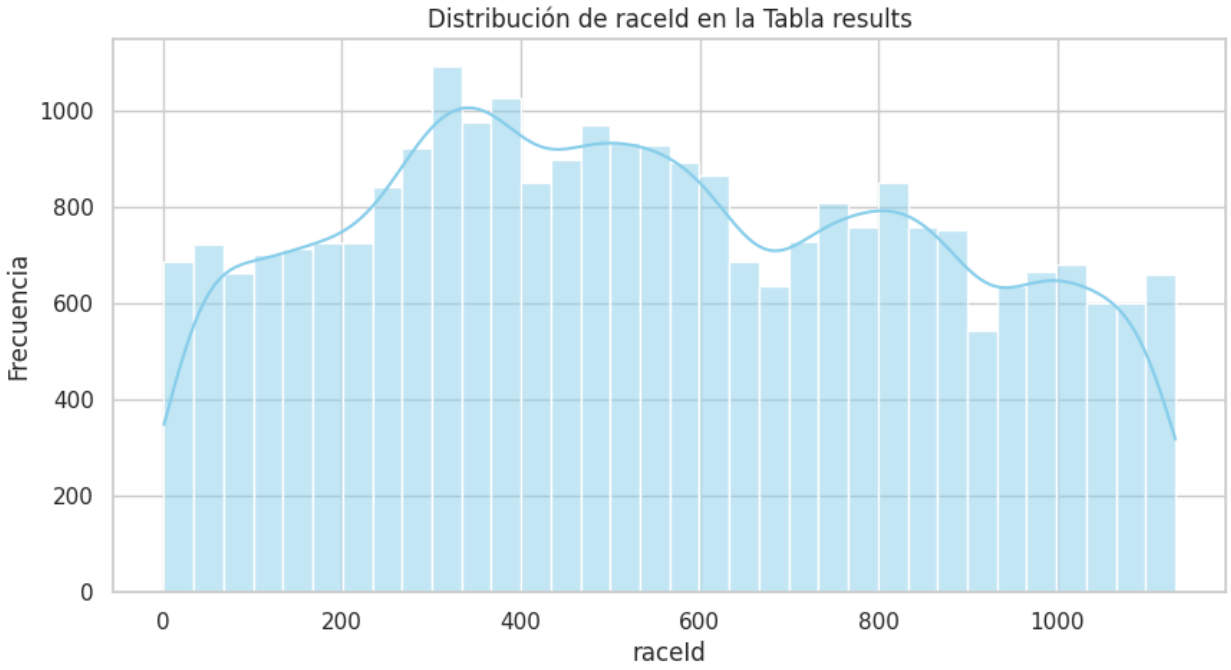

Distribución de raceId en la Tabla races
Frecuencia
raceId
0
20
40
60
80
100
0
200
400
600
800
1000
1200

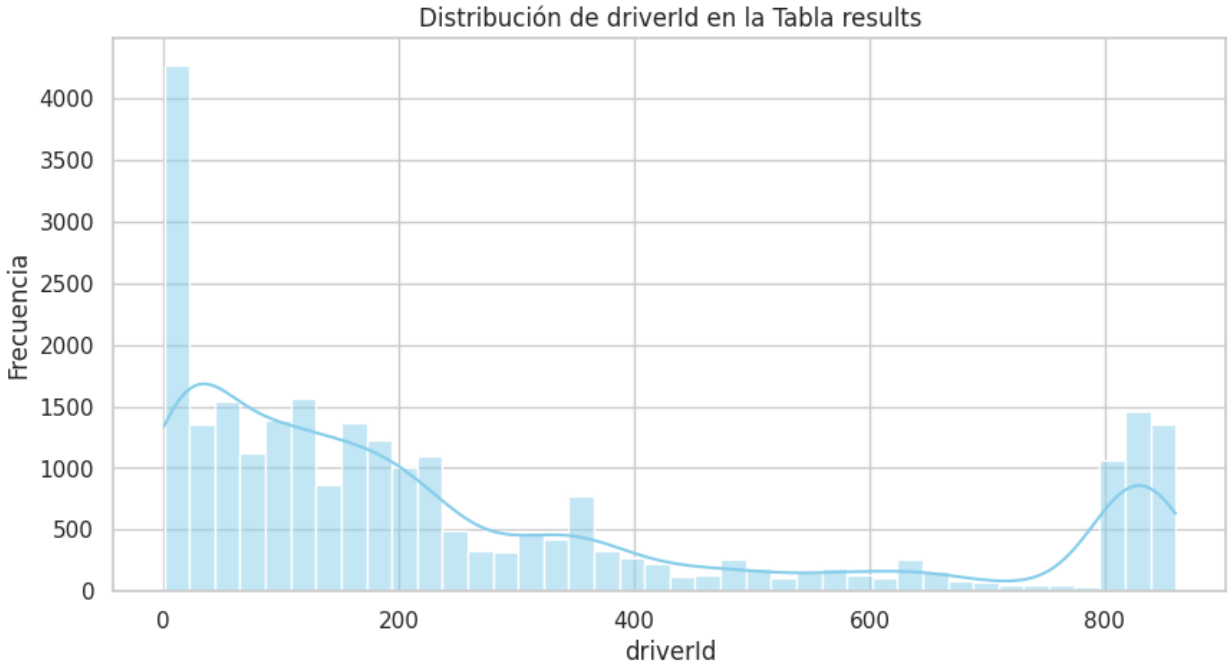

Distribución de year en la Tabla races
Frecuencia
0
20
40
60
80
100
120
140
1950
1960
1970
1980
1990
2000
2010
2020
year

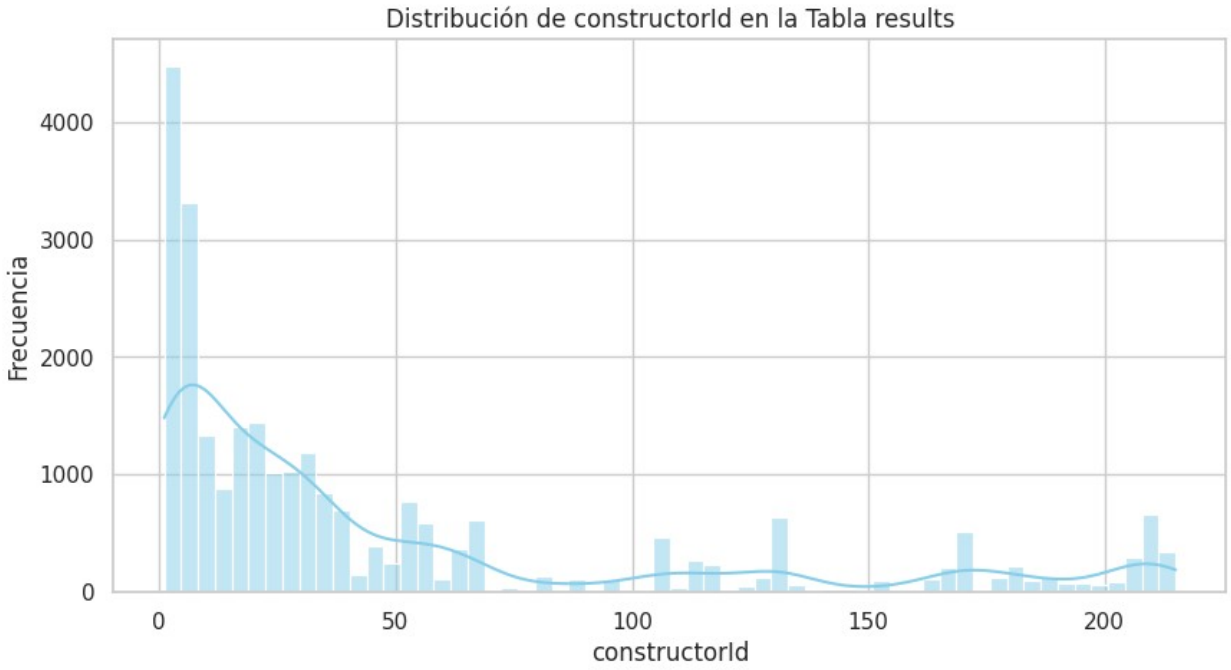

Distribución de round en la Tabla races
Frecuencia
0
20
40
60
80
100
120
140
0
5
10
15
20
25
round

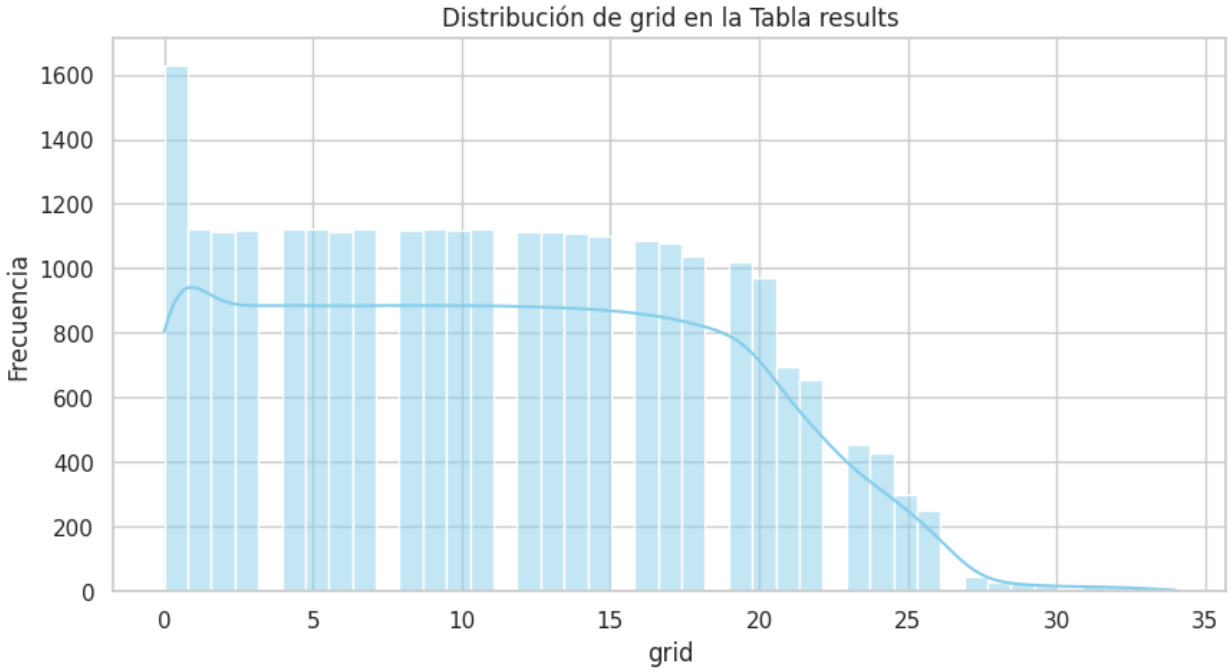

Distribución de circuitId en la Tabla races
Frecuencia
0
50
100
150
200
0
10
20
30
40
50
60
70
80
circuitId

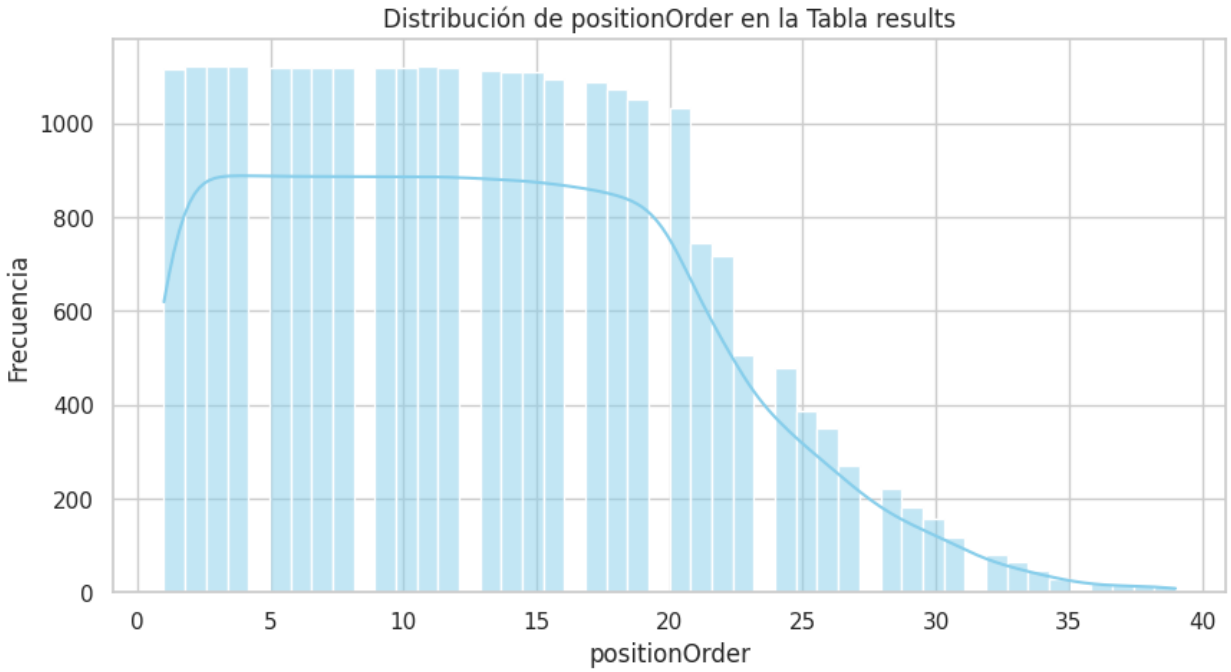

Distribución de resultId en la Tabla results
Frecuencia
0
200
400
600
800
0
5000
10000
15000
20000
25000
resultId

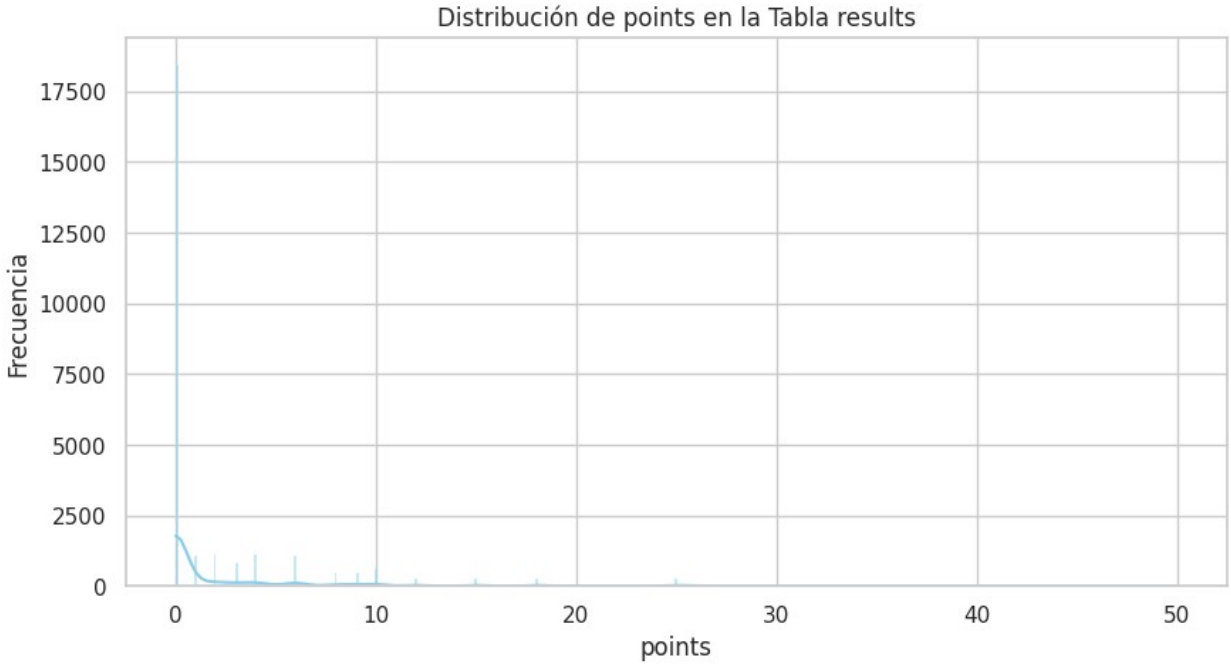

Distribución de raceId en la Tabla results
Frecuencia
0
200
400
600
800
1000
raceId

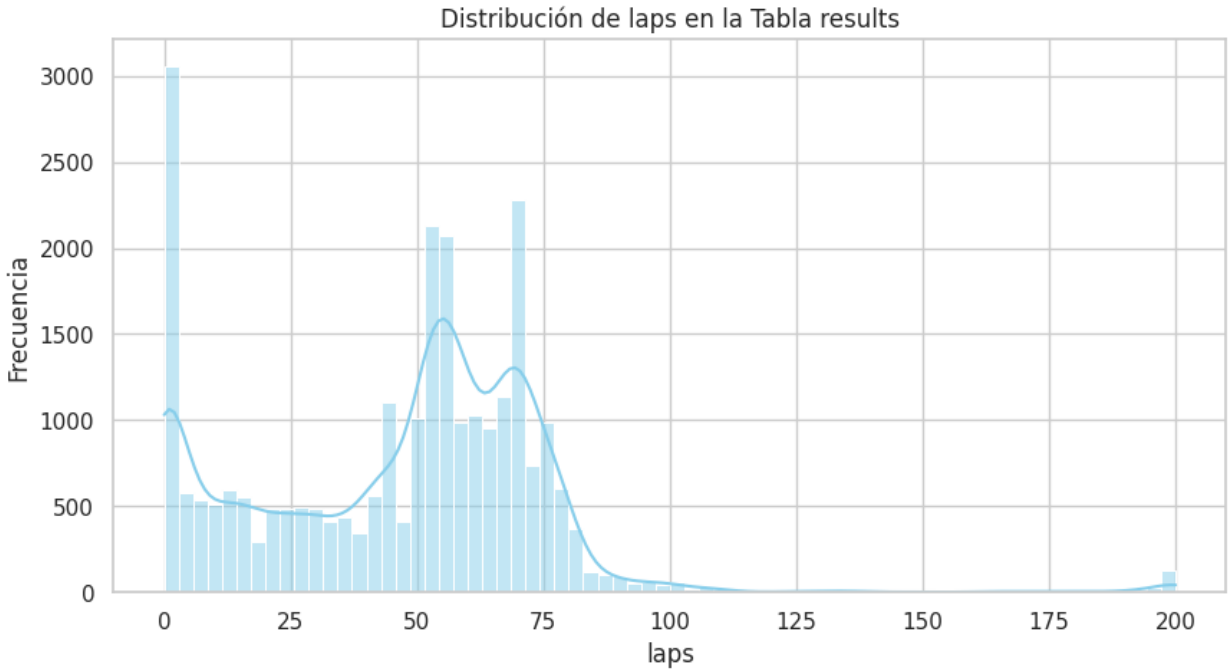

Distribución de driverId en la Tabla results
Frecuencia
0
500
1000
1500
2000
2500
3000
3500
4000
driverId

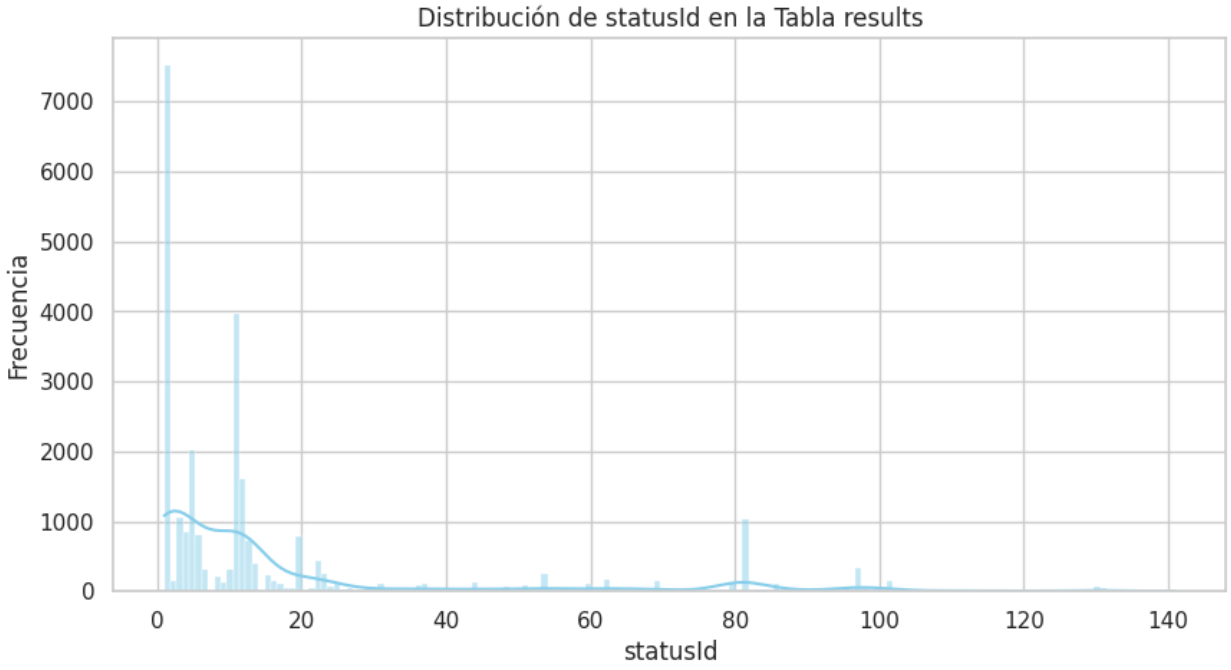

Distribución de constructorId en la Tabla results
Frecuencia
constructorId
0
1000
2000
3000
4000
0
50
100
150
200

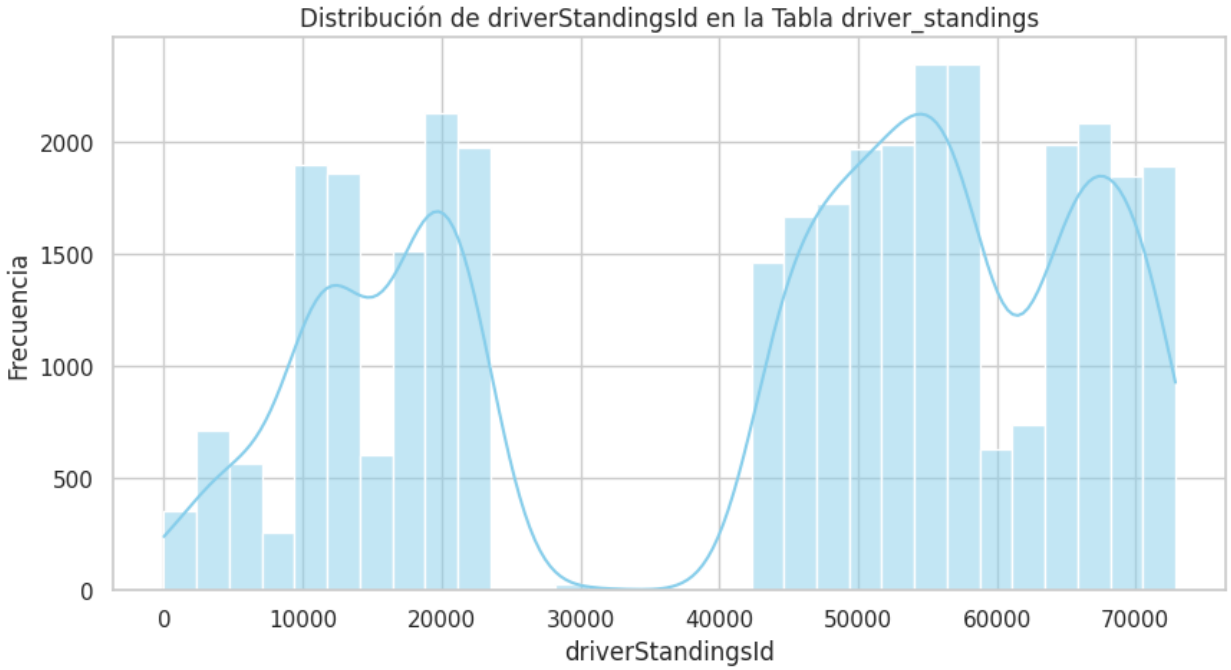

Distribución de grid en la Tabla results
Frecuencia
grid
0
200
400
600
800
1000
1200
1400
1600
0
5
10
15
20
25
30
35

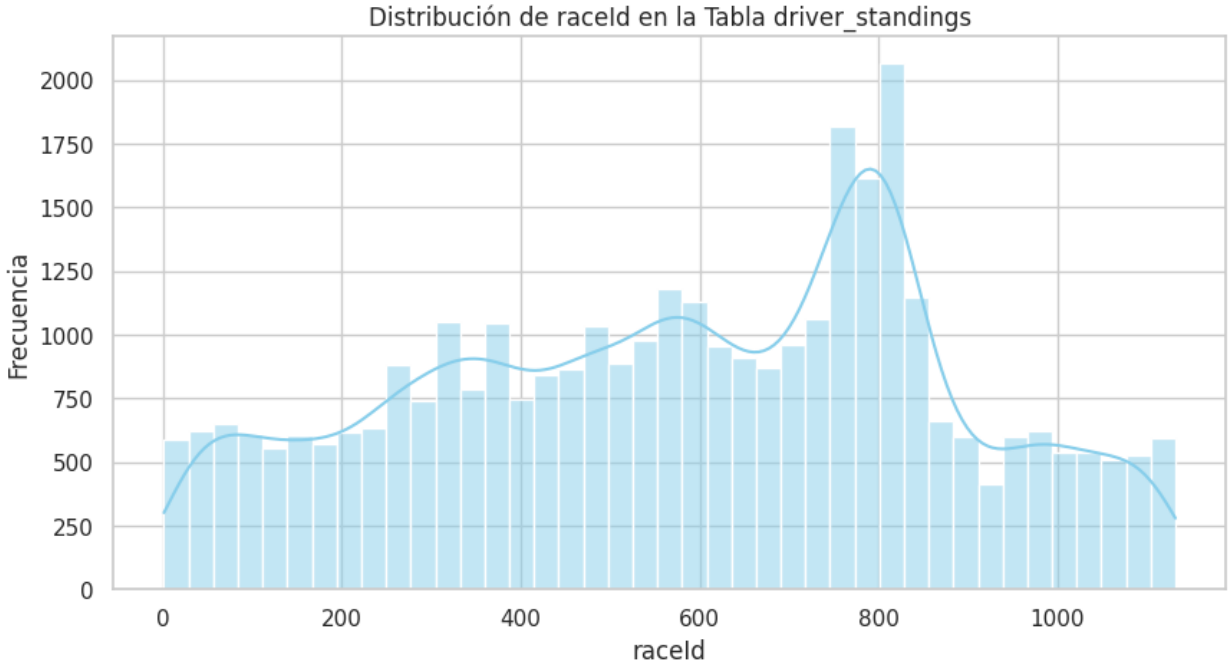

Distribución de positionOrder en la Tabla results
Frecuencia
0
200
400
600
800
1000
0
5
10
15
20
25
30
35
40
positionOrder

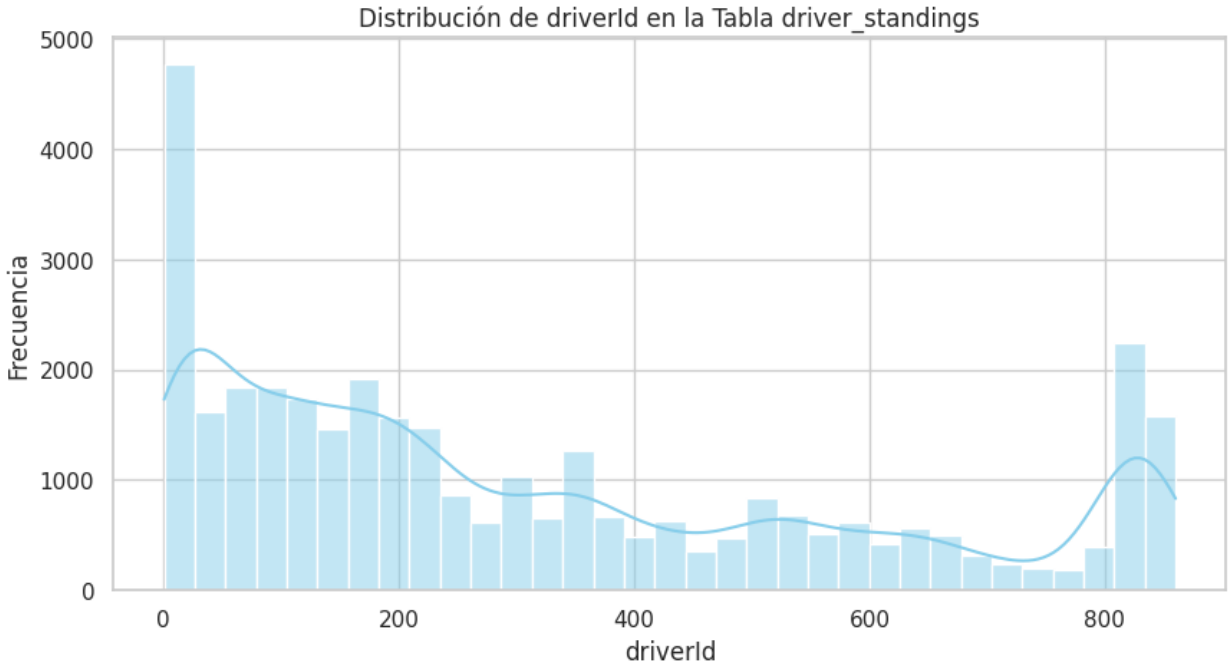

Distribución de points en la Tabla results
Frecuencia
0
2500
5000
7500
10000
12500
15000
17500
0
10
20
30
40
50
points

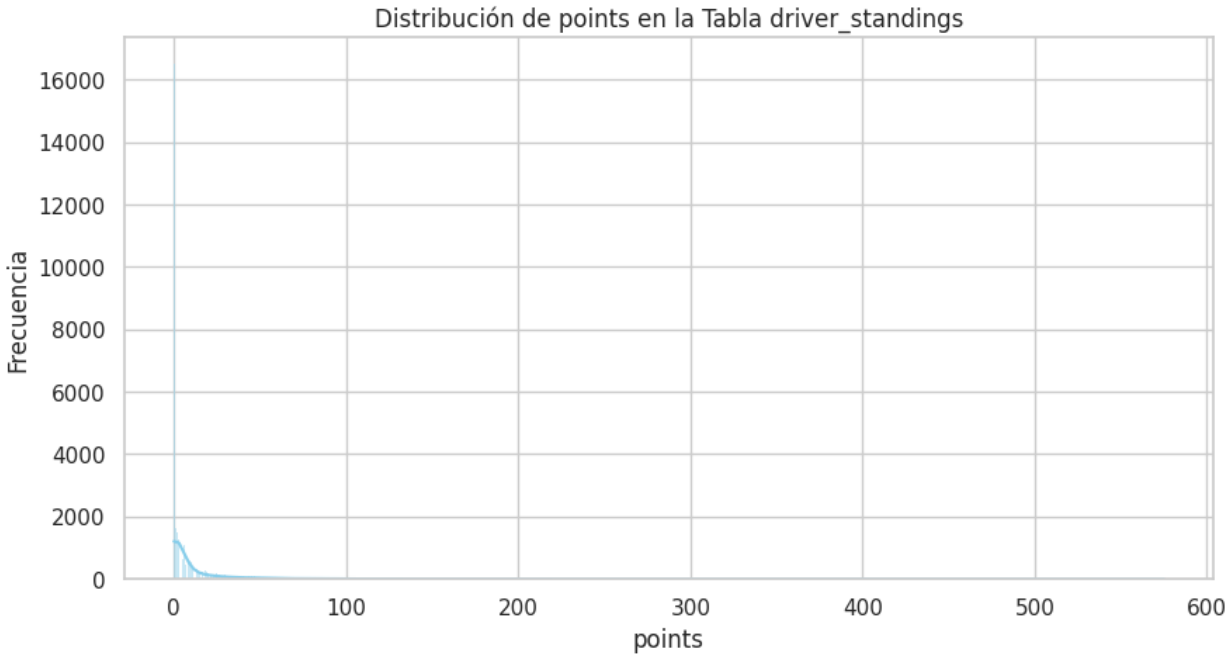

Distribución de laps en la Tabla results
Frecuencia
laps
0
500
1000
1500
2000
2500
3000
0
25
50
75
100
125
150
175
200

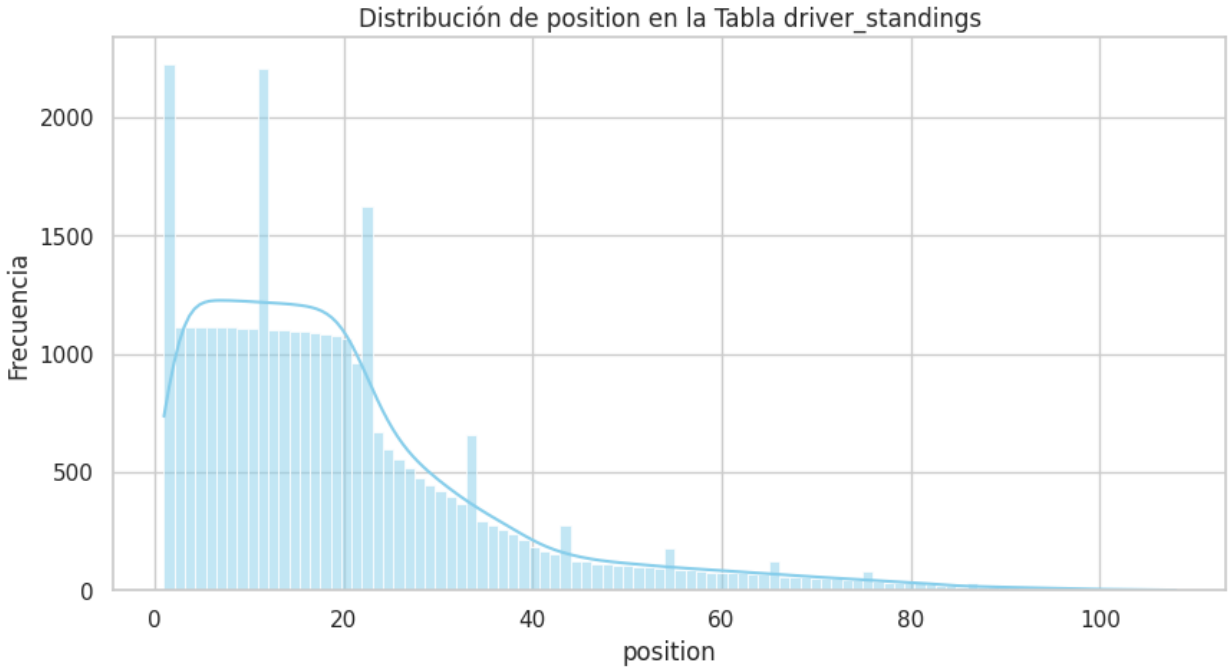

Distribución de statusId en la Tabla results
Frecuencia
statusId
0
1000
2000
3000
4000
5000
6000
7000
0
20
40
60
80
100
120
140

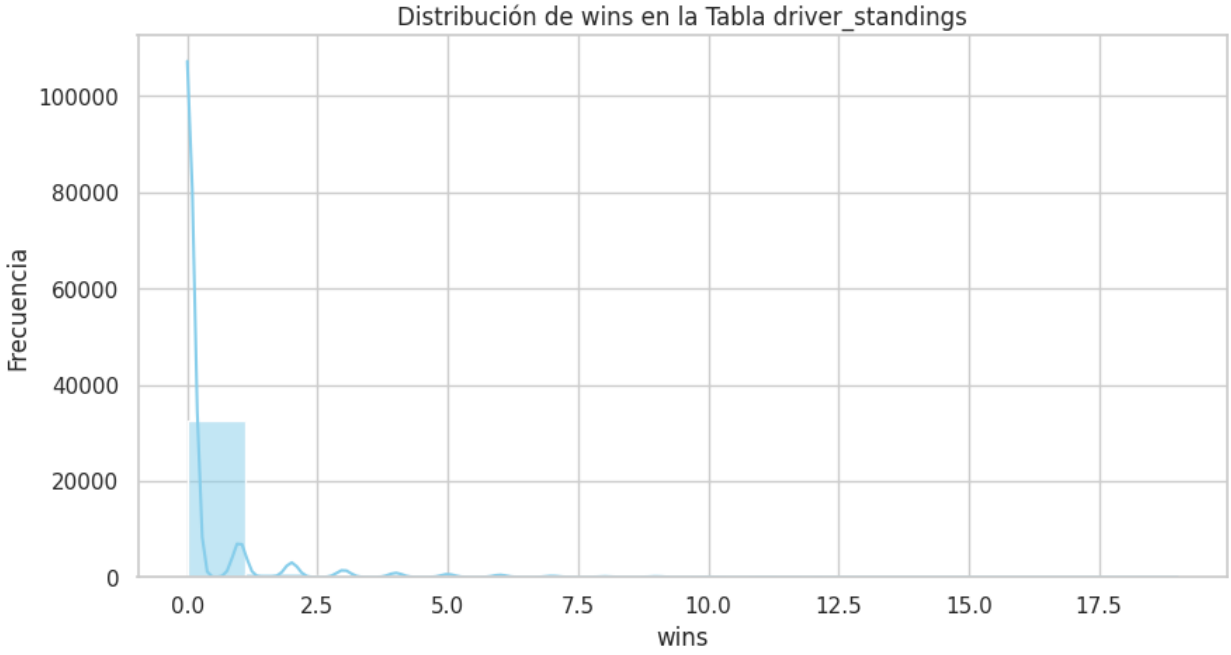

Distribución de driverStandingsId en la Tabla driver_standings
Frecuencia
0
500
1000
1500
2000
0
10000
20000
30000
40000
50000
60000
70000
driverStandingsId

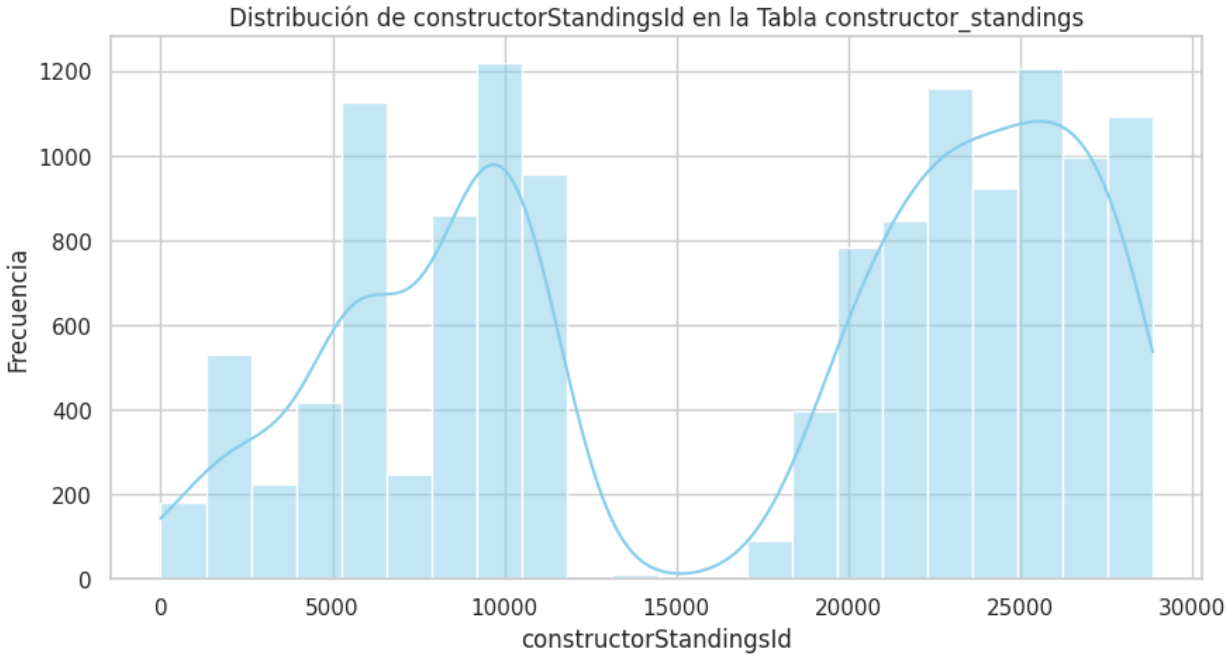

Distribución de raceId en la Tabla driver_standings
Frecuencia
0
250
500
750
1000
1250
1500
1750
2000
0
200
400
600
800
1000
raceId

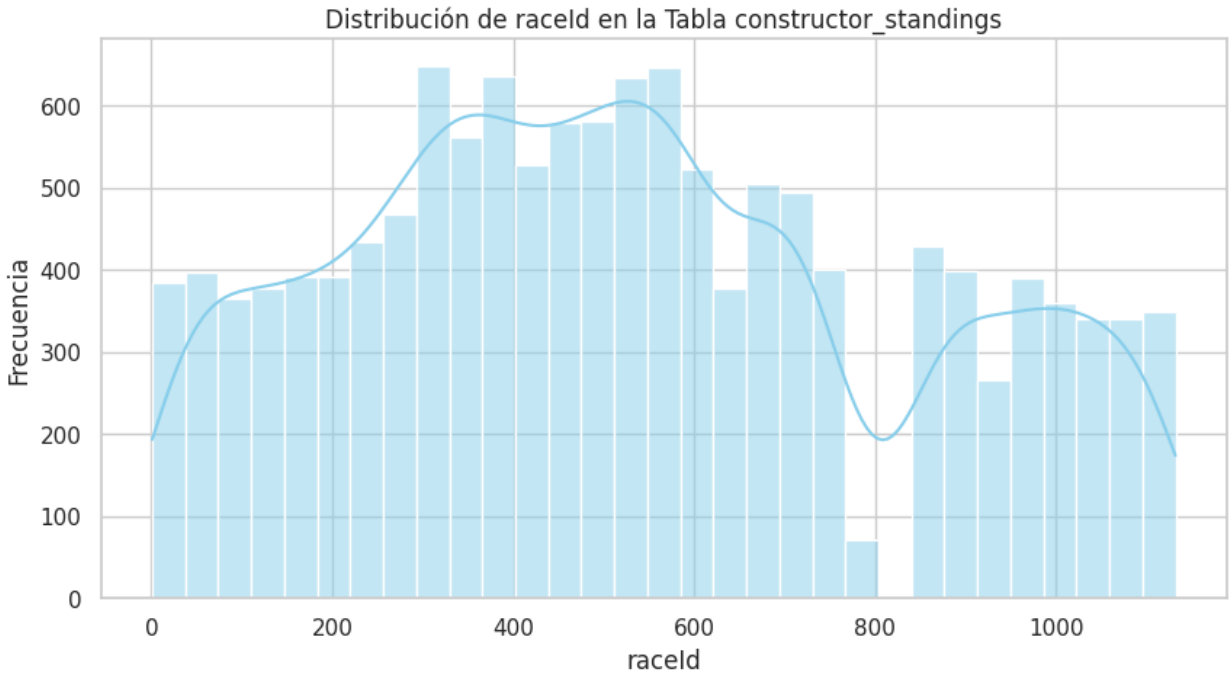

Distribución de driverId en la Tabla driver_standings
Frecuencia
driverId
0
1000
2000
3000
4000
5000
0
200
400
600
800

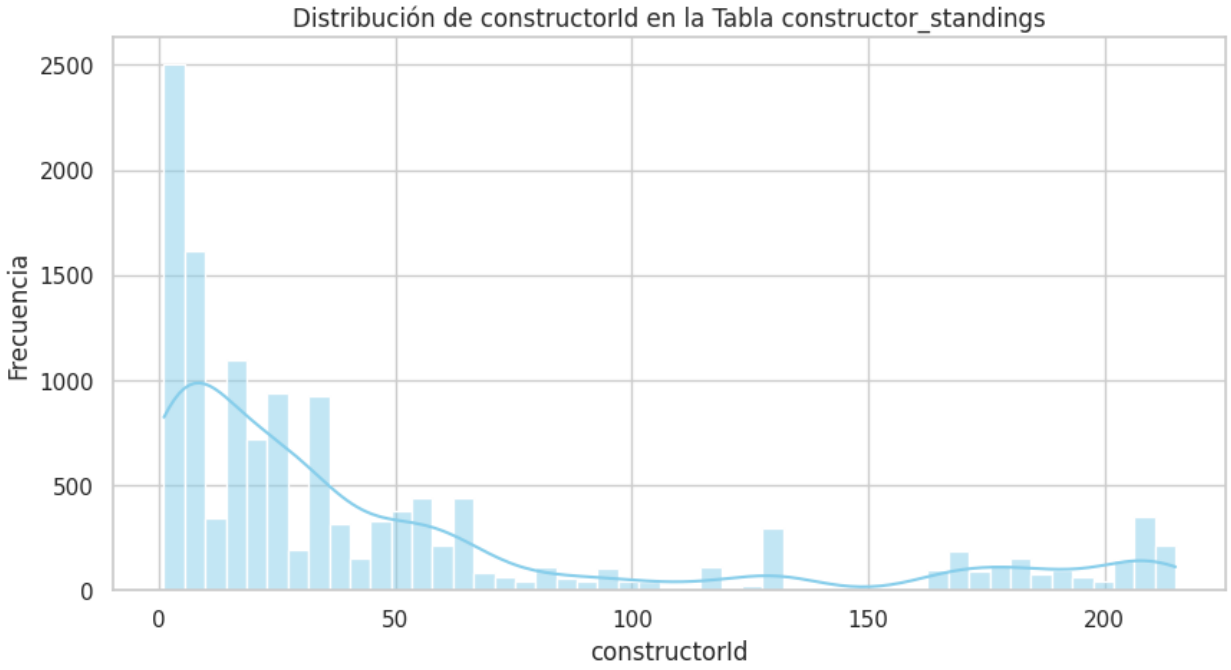

Distribución de points en la Tabla driver_standings
Frecuencia
points
0
2000
4000
6000
8000
10000
12000
14000
16000
0
100
200
300
400
500
600

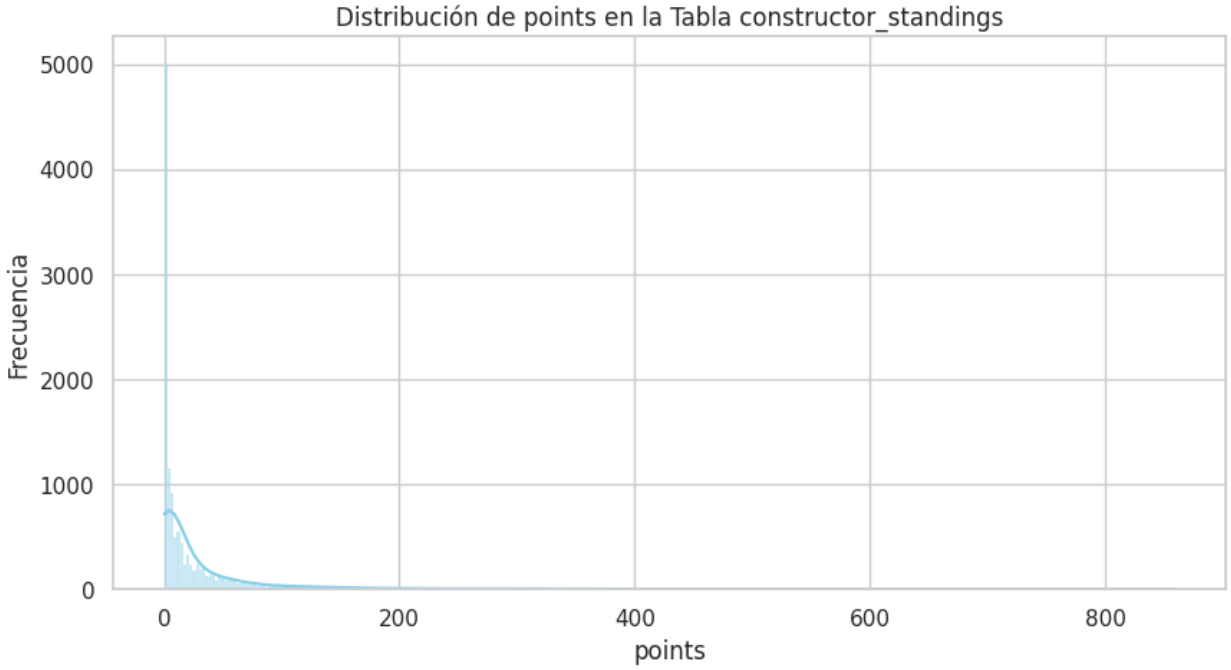

Distribución de position en la Tabla driver_standings
Frecuencia
2000
1500
1000
500
0
0
20
40
60
80
100
position

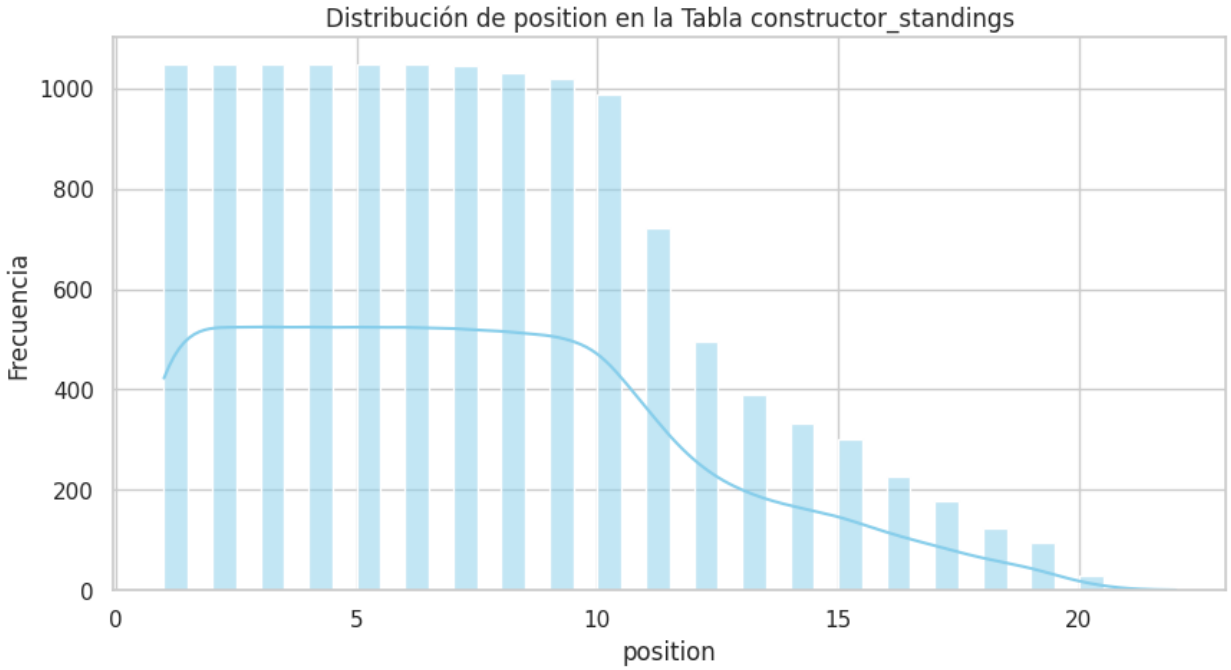

Distribución de wins en la Tabla driver_standings
Frecuencia
100000
80000
60000
40000
20000
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
wins

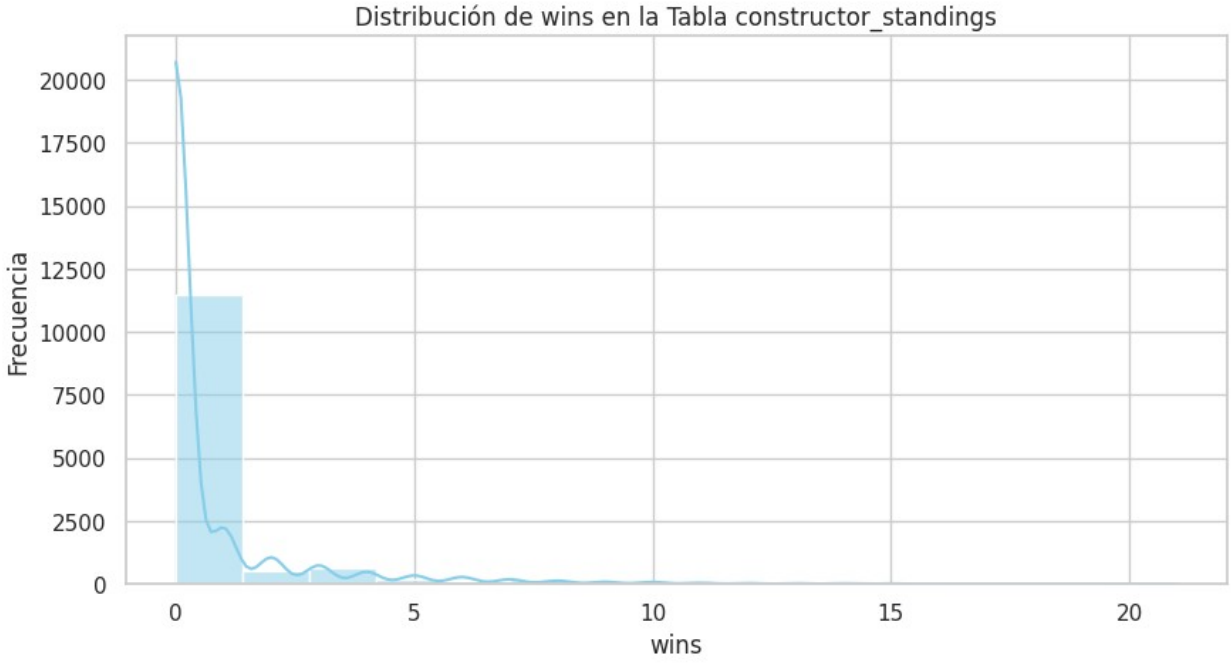

Distribución de constructorStandingsId en la Tabla constructor_standings
Frecuencia
0
200
400
600
800
1000
1200
0
5000
10000
15000
20000
25000
30000
constructorStandingsId


Distribución de raceId en la Tabla constructor_standings
Frecuencia
0
100
200
300
400
500
600
0
200
400
600
800
1000
raceId


Distribución de constructorId en la Tabla constructor_standings
Frecuencia
2500
2000
1500
1000
500
0
0
50
100
150
200
constructorId

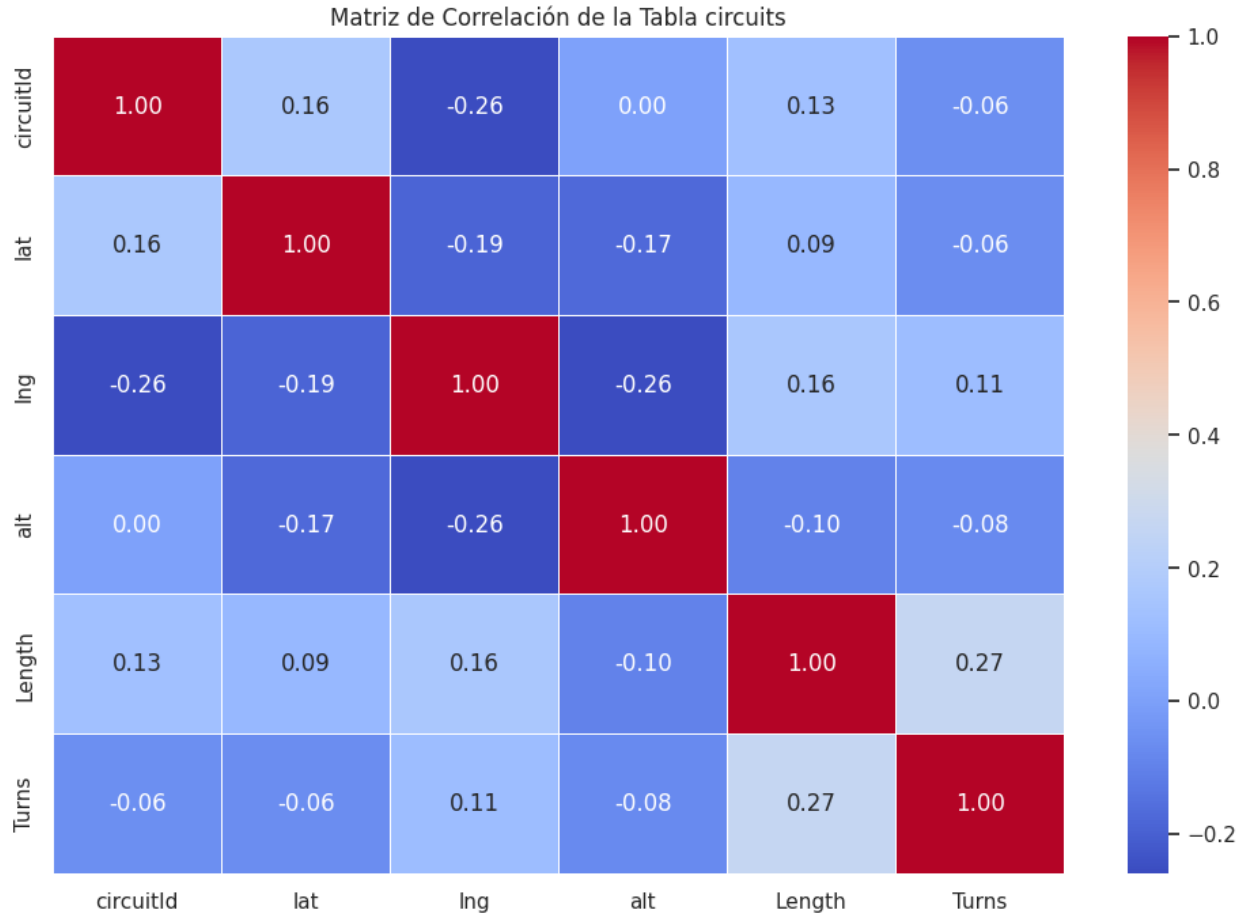

Distribución de points en la Tabla constructor_standings
Frecuencia
5000
4000
3000
2000
1000
0
0
200
400
600
800
points

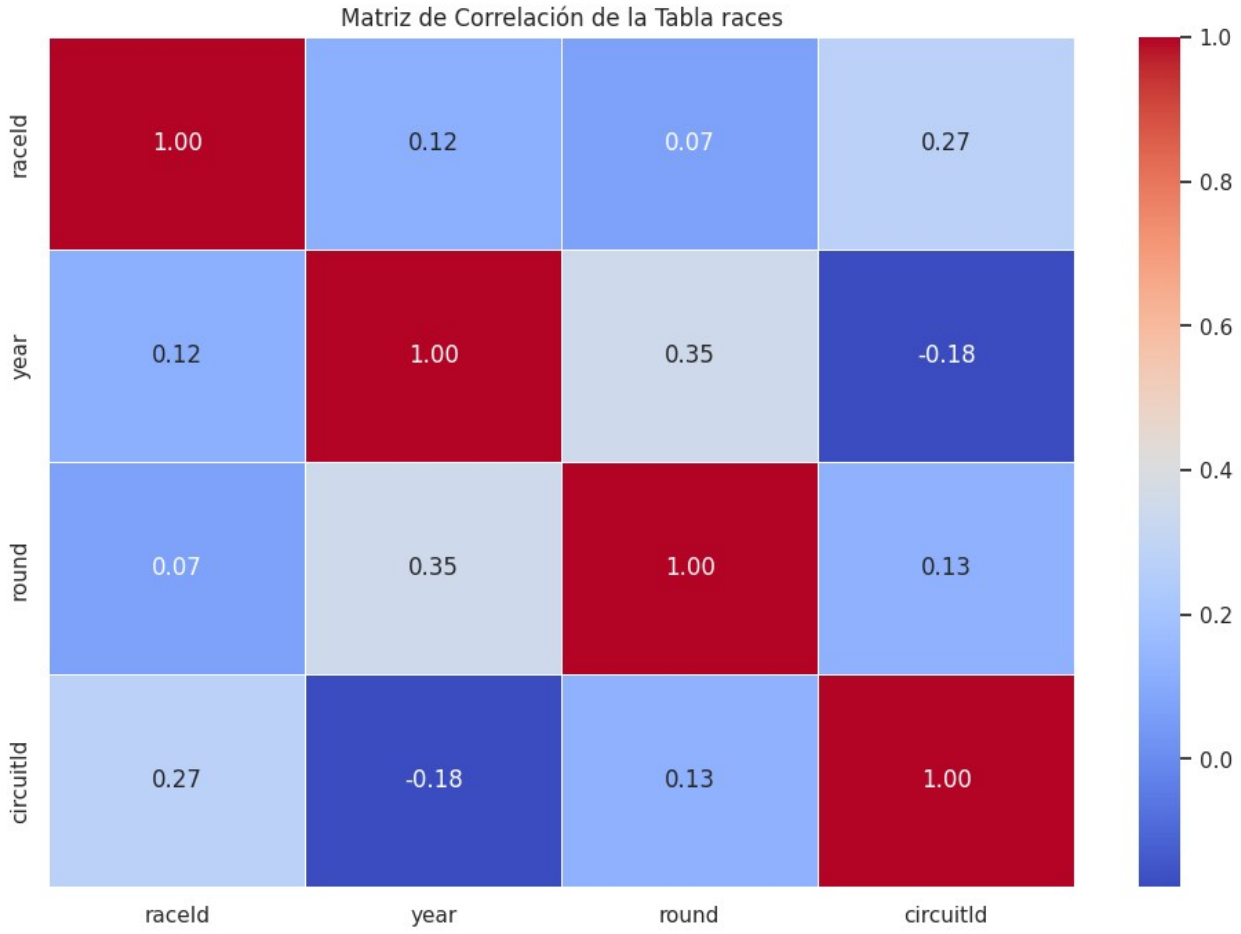


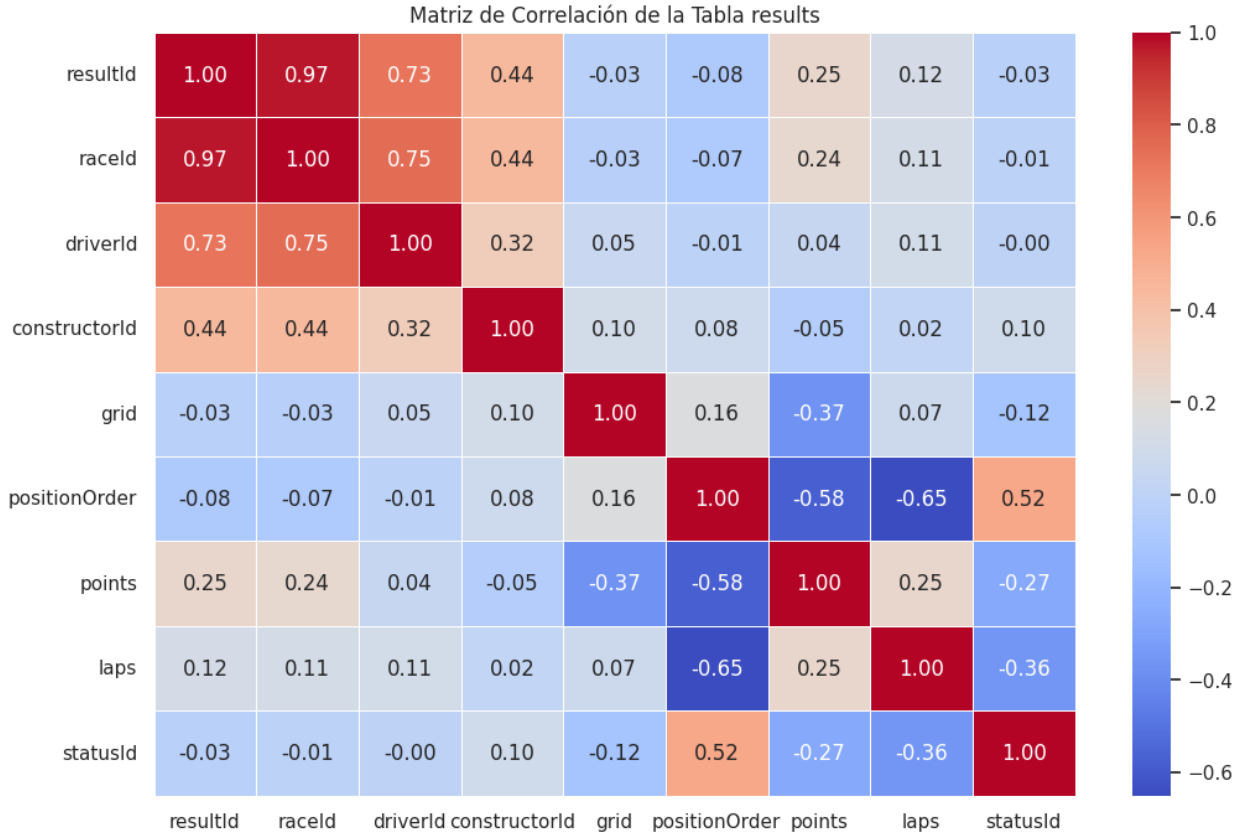


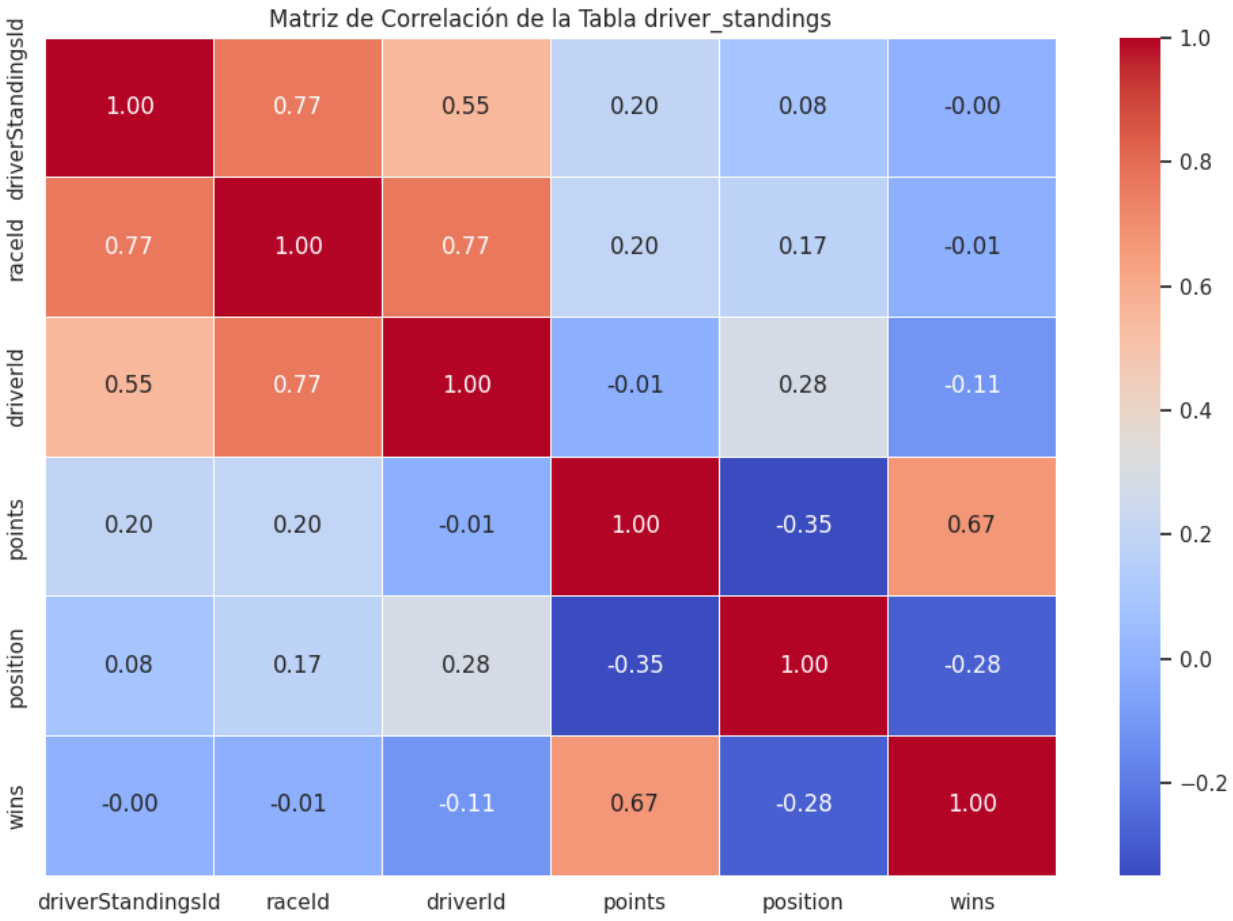

```
# Función para mostrar la matriz de correlación y graficarla
def plot_correlation_matrix(df, name):
    numeric_columns = df.select_dtypes(include=['float64', 'int64'])
    if not numeric_columns.empty:
        plt.figure(figsize=(12, 8))
        correlation = numeric_columns.corr()
        sns.heatmap(correlation, annot=True, cmap='coolwarm',
fmt=".2f", linewidths=0.5)
```

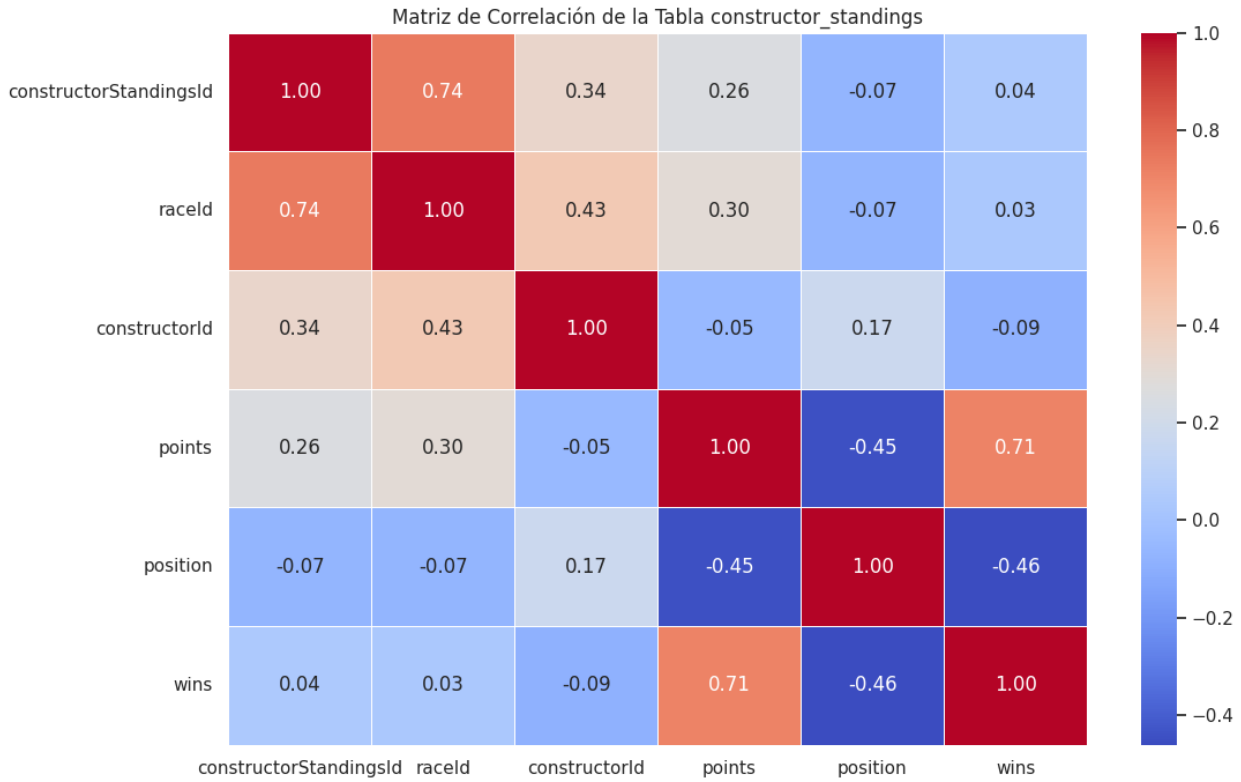
```
        plt.title(f'Matriz de Correlación de la Tabla {name}')
        plt.show()
    else:
        print(f"No hay columnas numéricas para mostrar correlaciones
en la tabla {name}")

# Graficar matrices de correlación para tablas importantes
plot_correlation_matrix(circuits, "circuits")
plot_correlation_matrix(races, "races")
plot_correlation_matrix(results, "results")
plot_correlation_matrix(driver_standings, "driver_standings")
plot_correlation_matrix(constructor_standings,
"constructor_standings")
```




Matriz de Correlación de la Tabla races
raceId
year
round
circuitId
1.00
0.12
0.07
0.27
0.12
1.00
0.35
-0.18
0.07
0.35
1.00
0.13
0.27
-0.18
0.13
1.00
raceId
year
round
circuitId
1.0
0.8
0.6
0.4
0.2
0.0


Matriz de Correlación de la Tabla results
resultId 1.00 0.97 0.73 0.44 -0.03 -0.08 0.25 0.12 -0.03
raceId 0.97 1.00 0.75 0.44 -0.03 -0.07 0.24 0.11 -0.01
driverId 0.73 0.75 1.00 0.32 0.05 -0.01 0.04 0.11 -0.00
constructorId 0.44 0.44 0.32 1.00 0.10 0.08 -0.05 0.02 0.10
grid -0.03 -0.03 0.05 0.10 1.00 0.16 -0.37 0.07 -0.12
positionOrder -0.08 -0.07 -0.01 0.08 0.16 1.00 -0.58 -0.65 0.52
points 0.25 0.24 0.04 -0.05 -0.37 -0.58 1.00 0.25 -0.27
laps 0.12 0.11 0.11 0.02 0.07 -0.65 0.25 1.00 -0.36
statusId -0.03 -0.01 -0.00 0.10 -0.12 0.52 -0.27 -0.36 1.00
resultId raceId driverId constructorId grid positionOrder points laps statusId
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


Matriz de Correlación de la Tabla driver_standings
driverStandingsId
raceId
driverId
points
position
wins
1.00 0.77 0.55 0.20 0.08 -0.00
0.77 1.00 0.77 0.20 0.17 -0.01
0.55 0.77 1.00 -0.01 0.28 -0.11
0.20 0.20 -0.01 1.00 -0.35 0.67
0.08 0.17 0.28 -0.35 1.00 -0.28
-0.00 -0.01 -0.11 0.67 -0.28 1.00
driverStandingsId
raceId
driverId
points
position
wins
1.0
0.8
0.6
0.4
0.2
0.0
−0.2



#PCA Y CLUSTERING

¿Cuáles son los circuitos más parecidos?

Este análisis se realiza desde el punto de vista de los mecánicos, ya que para ellos es importante conocer las características de los circuitos a la hora de preparar los coches. Reduciendo la dimensionalidad y haciendo clusters podemos observar cuales son los circuitos que más se parecen para qeu si en alguna carrera el coche ha hecho un gran desempeño, puedan replicarlo para los circuitos similares.

```
import pandas as pd
import requests
from bs4 import BeautifulSoup

# Crear función para obtener los datos de cada página de Wikipedia del
circuito
def extract_circuit_data(url):
    # Realizamos una solicitud GET para obtener el contenido de la
página
    response = requests.get(url)

    if response.status_code != 200:
        return None, None  # Si no se obtiene la página correctamente,
retornar valores nulos

    # Parsear el HTML usando BeautifulSoup
```

```
    soup = BeautifulSoup(response.text, 'html.parser')

    # Variables para almacenar los datos
    length = None
    turns = None

    # Buscar los infobox de cada circuito que suelen tener la
información en tablas
    infobox = soup.find('table', {'class': 'infobox'})

    if infobox:
        # Buscar la fila con la información sobre la longitud
        length_row = infobox.find('th', string="Length")
        if length_row:
            length = length_row.find_next('td').text.strip().split()
[0]  # Solo tomar los kilómetros

        # Buscar la fila con la información sobre las vueltas
        turns_row = infobox.find('th', string="Turns")
        if turns_row:
            turns = turns_row.find_next('td').text.strip()

    return length, turns

# Suponemos que tienes un dataframe 'circuits' con la columna 'url'

# Aplicar la función a cada URL de la tabla 'circuits' y extraer los
datos
circuits['Length'], circuits['Turns'] =
zip(*circuits['url'].apply(extract_circuit_data))

# Eliminar la columna 'Race lap record' si ya no es necesaria
circuits = circuits.drop(columns=['Race lap record'], errors='ignore')

# Mostrar el resultado para revisar los datos
print(circuits.head())
```

```
   circuitId   circuitRef                            name
location  \
0          1  albert_park  Albert Park Grand Prix Circuit
Melbourne
1          2       sepang    Sepang International Circuit  Kuala
Lumpur
2          3      bahrain   Bahrain International Circuit
Sakhir
3          4    catalunya  Circuit de Barcelona-Catalunya
Montmeló
4          5     istanbul                  Istanbul Park
Istanbul
```

```
     country       lat        lng  alt  \
0  Australia -37.84970  144.96800   10
1   Malaysia   2.76083  101.73800   18
2    Bahrain  26.03250   50.51060    7
3      Spain  41.57000    2.26111  109
4     Turkey  40.95170   29.40500  130

                                                 url Length Turns
0  http://en.wikipedia.org/wiki/Melbourne_Grand_P...  5.278    14
1  http://en.wikipedia.org/wiki/Sepang_Internatio...  5.543    15
2  http://en.wikipedia.org/wiki/Bahrain_Internati...  5.412    15
3  http://en.wikipedia.org/wiki/Circuit_de_Barcel...  4.657    14
4         http://en.wikipedia.org/wiki/Istanbul_Park  5.338    14
```

```python
import pandas as pd

# Configurar pandas para mostrar todas las filas (sin truncar la tabla)
pd.set_option('display.max_rows', None)  # Sin límite de filas
pd.set_option('display.max_columns', None)  # Mostrar todas las columnas
pd.set_option('display.width', None)  # Ajustar el ancho automáticamente
pd.set_option('display.max_colwidth', None)  # Mostrar todo el contenido de las celdas

# Mostrar toda la tabla 'circuits'
print(circuits)
```

```
    circuitId      circuitRef                                  name  \
location
0           1     albert_park         Albert Park Grand Prix Circuit
Melbourne
1           2           sepang           Sepang International Circuit
Kuala Lumpur
2           3         bahrain          Bahrain International Circuit
Sakhir
3           4       catalunya         Circuit de Barcelona-Catalunya
Montmeló
4           5        istanbul                         Istanbul Park
Istanbul
5           6          monaco                      Circuit de Monaco
Monte-Carlo
6           7      villeneuve              Circuit Gilles Villeneuve
Montreal
7           8     magny_cours          Circuit de Nevers Magny-Cours
Magny Cours
8           9     silverstone                    Silverstone Circuit
Silverstone
9          10  hockenheimring                         Hockenheimring
```

```
Hockenheim
10          11      hungaroring                              Hungaroring
Budapest
11          12         valencia                 Valencia Street Circuit
Valencia
12          13              spa             Circuit de Spa-Francorchamps
Spa
13          14            monza             Autodromo Nazionale di Monza
Monza
14          15       marina_bay                Marina Bay Street Circuit
Marina Bay
15          16             fuji                            Fuji Speedway
Oyama
16          17         shanghai          Shanghai International Circuit
Shanghai
17          18       interlagos              Autódromo José Carlos Pace
São Paulo
18          19     indianapolis             Indianapolis Motor Speedway
Indianapolis
19          20      nurburgring                             Nürburgring
Nürburg
20          21            imola           Autodromo Enzo e Dino Ferrari
Imola
21          22           suzuka                          Suzuka Circuit
Suzuka
22          80            vegas          Las Vegas Strip Street Circuit
Las Vegas
23          24       yas_marina                      Yas Marina Circuit
Abu Dhabi
24          25           galvez           Autódromo Juan y Oscar Gálvez
Buenos Aires
25          26            jerez                       Circuito de Jerez
Jerez de la Frontera
26          27          estoril                   Autódromo do Estoril
Estoril
27          28          okayama           Okayama International Circuit
Okayama
28          29         adelaide                 Adelaide Street Circuit
Adelaide
29          30          kyalami                                 Kyalami
Midrand
30          31        donington                          Donington Park
Castle Donington
31          32        rodriguez             Autódromo Hermanos Rodríguez
Mexico City
32          33          phoenix                   Phoenix street circuit
Phoenix
33          34           ricard                     Circuit Paul Ricard
Le Castellet
```

```
34          35          yeongam            Korean International Circuit
Yeongam County
35          36      jacarepagua  Autódromo Internacional Nelson Piquet
Rio de Janeiro
36          37          detroit                  Detroit Street Circuit
Detroit
37          38     brands_hatch                             Brands Hatch
Kent
38          39        zandvoort                 Circuit Park Zandvoort
Zandvoort
39          40           zolder                                 Zolder
Heusden-Zolder
40          41            dijon                           Dijon-Prenois
Dijon
41          42           dallas                                Fair Park
Dallas
42          43       long_beach                               Long Beach
California
43          44        las_vegas               Las Vegas Street Circuit
Nevada
44          45           jarama                                 Jarama
Madrid
45          46     watkins_glen                             Watkins Glen
New York State
46          47       anderstorp                   Scandinavian Raceway
Anderstorp
47          48          mosport          Mosport International Raceway
Ontario
48          49         montjuic                               Montjuïc
Barcelona
49          50         nivelles                       Nivelles-Baulers
Brussels
50          51          charade                        Charade Circuit
Clermont-Ferrand
51          52        tremblant                 Circuit Mont-Tremblant
Quebec
52          53          essarts                      Rouen-Les-Essarts
Rouen
53          54           lemans                                 Le Mans
Le Mans
54          55            reims                             Reims-Gueux
Reims
55          56           george                  Prince George Circuit
Eastern Cape Province
56          57          zeltweg                                Zeltweg
Styria
57          58          aintree                                Aintree
Liverpool
58          59         boavista                   Circuito da Boavista
```

```
Oporto
59         60       riverside        Riverside International Raceway
California
60         61            avus                                    AVUS
Berlin
61         62        monsanto                  Monsanto Park Circuit
Lisbon
62         63         sebring          Sebring International Raceway
Florida
63         64        ain-diab                             Ain Diab
Casablanca
64         65         pescara                       Pescara Circuit
Pescara
65         66      bremgarten                     Circuit Bremgarten
Bern
66         67       pedralbes                   Circuit de Pedralbes
Barcelona
67         68           buddh            Buddh International Circuit
Uttar Pradesh
68         69        americas                Circuit of the Americas
Austin
69         70   red_bull_ring                          Red Bull Ring
Spielberg
70         71           sochi                        Sochi Autodrom
Sochi
71         73            baku                      Baku City Circuit
Baku
72         75        portimao     Autódromo Internacional do Algarve
Portimão
73         76         mugello   Autodromo Internazionale del Mugello
Mugello
74         77          jeddah                Jeddah Corniche Circuit
Jeddah
75         78          losail           Losail International Circuit
Al Daayen
76         79           miami          Miami International Autodrome
Miami

          country       lat         lng   alt  \
0       Australia -37.84970  144.968000    10
1        Malaysia   2.76083  101.738000    18
2         Bahrain  26.03250   50.510600     7
3           Spain  41.57000    2.261110   109
4          Turkey  40.95170   29.405000   130
5          Monaco  43.73470    7.420560     7
6          Canada  45.50000  -73.522800    13
7          France  46.86420    3.163610   228
8              UK  52.07860   -1.016940   153
9         Germany  49.32780    8.565830   103
```

10	Hungary	47.57890	19.248600	264
11	Spain	39.45890	-0.331667	4
12	Belgium	50.43720	5.971390	401
13	Italy	45.61560	9.281110	162
14	Singapore	1.29140	103.864000	18
15	Japan	35.37170	138.927000	583
16	China	31.33890	121.220000	5
17	Brazil	-23.70360	-46.699700	785
18	USA	39.79500	-86.234700	223
19	Germany	50.33560	6.947500	578
20	Italy	44.34390	11.716700	37
21	Japan	34.84310	136.541000	45
22	United States	36.11470	-115.173000	642
23	UAE	24.46720	54.603100	3
24	Argentina	-34.69430	-58.459300	8
25	Spain	36.70830	-6.034170	37
26	Portugal	38.75060	-9.394170	130
27	Japan	34.91500	134.221000	266
28	Australia	-34.92720	138.617000	58
29	South Africa	-25.98940	28.076700	1460
30	UK	52.83060	-1.375280	88
31	Mexico	19.40420	-99.090700	2227
32	USA	33.44790	-112.075000	345
33	France	43.25060	5.791670	432
34	Korea	34.73330	126.417000	0
35	Brazil	-22.97560	-43.395000	1126
36	USA	42.32980	-83.040100	177
37	UK	51.35690	0.263056	145
38	Netherlands	52.38880	4.540920	6
39	Belgium	50.98940	5.256940	36
40	France	47.36250	4.899130	484
41	USA	32.77740	-96.758700	139
42	USA	33.76510	-118.189000	12
43	USA	36.11620	-115.174000	639
44	Spain	40.61710	-3.585580	609
45	USA	42.33690	-76.927200	485
46	Sweden	57.26530	13.604200	153
47	Canada	44.04810	-78.675600	332
48	Spain	41.36640	2.151670	79
49	Belgium	50.62110	4.326940	139
50	France	45.74720	3.038890	790
51	Canada	46.18770	-74.609900	214
52	France	49.33060	1.004580	81
53	France	47.95000	0.224231	67
54	France	49.25420	3.930830	88
55	South Africa	-33.04860	27.873600	15
56	Austria	47.20390	14.747800	676
57	UK	53.47690	-2.940560	20
58	Portugal	41.17050	-8.673250	28

```
59            USA  33.93700 -117.273000   470
60        Germany  52.48060   13.251400    53
61       Portugal  38.71970   -9.203060   158
62            USA  27.45470  -81.348300    18
63        Morocco  33.57860   -7.687500    19
64          Italy  42.47500   14.150800   129
65    Switzerland  46.95890    7.401940   551
66          Spain  41.39030    2.116670    85
67          India  28.34870   77.533100   194
68            USA  30.13280  -97.641100   161
69        Austria  47.21970   14.764700   678
70         Russia  43.40570   39.957800     2
71     Azerbaijan  40.37250   49.853300    -7
72       Portugal  37.22700   -8.626700   108
73          Italy  43.99750   11.371900   255
74   Saudi Arabia  21.63190   39.104400    15
75          Qatar  25.49000   51.454200    12
76            USA  25.95810  -80.238900     0


url       Length  Turns
0
http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit        5.278
14
1
http://en.wikipedia.org/wiki/Sepang_International_Circuit        5.543
15
2
http://en.wikipedia.org/wiki/Bahrain_International_Circuit
5.412     15
3               http://en.wikipedia.org/wiki/Circuit_de_Barcelona-
Catalunya        4.657     14
4
http://en.wikipedia.org/wiki/Istanbul_Park        5.338     14
5
http://en.wikipedia.org/wiki/Circuit_de_Monaco        3.337     19
6
http://en.wikipedia.org/wiki/Circuit_Gilles_Villeneuve        4.361
14
7               http://en.wikipedia.org/wiki/Circuit_de_Nevers_Magny-
Cours        4.411     17
8
http://en.wikipedia.org/wiki/Silverstone_Circuit     5.891[1]     18
9
http://en.wikipedia.org/wiki/Hockenheimring        4.574     17
10
http://en.wikipedia.org/wiki/Hungaroring        4.381     14
11
http://en.wikipedia.org/wiki/Valencia_Street_Circuit        5.419
```

```
25
12                http://en.wikipedia.org/wiki/Circuit_de_Spa-
Francorchamps        7.004     19
13
http://en.wikipedia.org/wiki/Autodromo_Nazionale_Monza  5.793[3][4]
11
14
http://en.wikipedia.org/wiki/Marina_Bay_Street_Circuit        4.940
19
15
http://en.wikipedia.org/wiki/Fuji_Speedway        4.563     16
16
http://en.wikipedia.org/wiki/Shanghai_International_Circuit
5.451     16
17        http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Jos
%C3%A9_Carlos_Pace        4.309     15
18
http://en.wikipedia.org/wiki/Indianapolis_Motor_Speedway        2.500
4
19                              http://en.wikipedia.org/wiki/N
%C3%BCrburgring       5.1480     15
20
http://en.wikipedia.org/wiki/Autodromo_Enzo_e_Dino_Ferrari
4.909     19
21
http://en.wikipedia.org/wiki/Suzuka_Circuit        5.807     18
22
https://en.wikipedia.org/wiki/Las_Vegas_Grand_Prix#Circuit
None   None
23
http://en.wikipedia.org/wiki/Yas_Marina_Circuit        5.281     16
24    http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Oscar_Alfredo_G
%C3%A1lvez        4.259     19
25
http://en.wikipedia.org/wiki/Circuito_Permanente_de_Jerez        4.428
15
26                    http://en.wikipedia.org/wiki/Aut
%C3%B3dromo_do_Estoril        4.182     14
27
http://en.wikipedia.org/wiki/TI_Circuit        3.703     13
28
http://en.wikipedia.org/wiki/Adelaide_Street_Circuit        3.219
14
29
http://en.wikipedia.org/wiki/Kyalami        4.529     16
30
http://en.wikipedia.org/wiki/Donington_Park        4.020     12
31      http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Hermanos_Rodr
%C3%ADguez     4.304[1]  17[1]
```

```
32
http://en.wikipedia.org/wiki/Phoenix_street_circuit        2.312
15
33
http://en.wikipedia.org/wiki/Paul_Ricard_Circuit        5.842     15
34
http://en.wikipedia.org/wiki/Korean_International_Circuit        5.615
18
35  http://en.wikipedia.org/wiki/Aut
%C3%B3dromo_Internacional_Nelson_Piquet        3.336      7
36
http://en.wikipedia.org/wiki/Detroit_street_circuit        1.645
10
37
http://en.wikipedia.org/wiki/Brands_Hatch        3.916      9
38
http://en.wikipedia.org/wiki/Circuit_Zandvoort        4.259     14
39
http://en.wikipedia.org/wiki/Zolder        4.010     10
40                              http://en.wikipedia.org/wiki/Dijon-
Prenois        3.801     12
41
http://en.wikipedia.org/wiki/Fair_Park         None   None
42
http://en.wikipedia.org/wiki/Long_Beach,_California         None
None
43
http://en.wikipedia.org/wiki/Las_Vegas_Street_Circuit        3.853
17
44
http://en.wikipedia.org/wiki/Circuito_Permanente_Del_Jarama
3.850     14
45
http://en.wikipedia.org/wiki/Watkins_Glen_International        3.450
11
46
http://en.wikipedia.org/wiki/Scandinavian_Raceway        4.025      8

47
http://en.wikipedia.org/wiki/Mosport        3.957     10
48                        http://en.wikipedia.org/wiki/Montju
%C3%AFc_circuit        3.791     12
49                             http://en.wikipedia.org/wiki/Nivelles-
Baulers        3.724      7
50
http://en.wikipedia.org/wiki/Charade_Circuit        3.975     18
51                       http://en.wikipedia.org/wiki/Circuit_Mont-
Tremblant        2.621     17
52                           http://en.wikipedia.org/wiki/Rouen-Les-
```

```
Essarts        5.543     13
53
http://en.wikipedia.org/wiki/Circuit_de_la_Sarthe#Bugatti_Circuit
13.626     38
54                                  http://en.wikipedia.org/wiki/Reims-
Gueux        8.302      7
55
http://en.wikipedia.org/wiki/Prince_George_Circuit        3.920      9

56
http://en.wikipedia.org/wiki/Zeltweg_Airfield         None   None
57
http://en.wikipedia.org/wiki/Aintree_Motor_Racing_Circuit        4.828
8
58
http://en.wikipedia.org/wiki/Circuito_da_Boavista        4.800     23

59
http://en.wikipedia.org/wiki/Riverside_International_Raceway
3.300      9
60
http://en.wikipedia.org/wiki/AVUS        2.660      6
61
http://en.wikipedia.org/wiki/Monsanto_Park_Circuit        5.440   None

62
http://en.wikipedia.org/wiki/Sebring_Raceway        3.741     17
63                             http://en.wikipedia.org/wiki/Ain-
Diab_Circuit        7.603   None
64
http://en.wikipedia.org/wiki/Pescara_Circuit       25.801   None
65
http://en.wikipedia.org/wiki/Circuit_Bremgarten        7.280     13
66
http://en.wikipedia.org/wiki/Pedralbes_Circuit        6.333      6
67
http://en.wikipedia.org/wiki/Buddh_International_Circuit        5.125
16
68
http://en.wikipedia.org/wiki/Circuit_of_the_Americas        3.426
20
69
http://en.wikipedia.org/wiki/Red_Bull_Ring        4.318  10[2]
70
http://en.wikipedia.org/wiki/Sochi_Autodrom        2.313     11
71
http://en.wikipedia.org/wiki/Baku_City_Circuit        6.003     20
72
http://en.wikipedia.org/wiki/Algarve_International_Circuit
4.653     15
```

```
73
http://en.wikipedia.org/wiki/Mugello_Circuit        5.245     15
74
http://en.wikipedia.org/wiki/Jeddah_Street_Circuit        6.174     27

75
http://en.wikipedia.org/wiki/Losail_International_Circuit        5.419
16
76
http://en.wikipedia.org/wiki/Miami_International_Autodrome
3.363     19
```

```python
import pandas as pd

# Limpiar la columna Length para quitar los corchetes y el contenido
dentro
circuits['Length'] = circuits['Length'].str.replace(r'\[.*\]', '',
regex=True)

# Limpiar la columna Turns para quitar los corchetes y el contenido
dentro
circuits['Turns'] = circuits['Turns'].str.replace(r'\[.*\]', '',
regex=True)

# Imputar los valores faltantes en 'Length' con la media de la columna
circuits['Length'] = circuits['Length'].apply(pd.to_numeric,
errors='coerce')  # Convertir a numérico (NaN si no se puede)
mean_length = circuits['Length'].mean()  # Calcular la media de
'Length'
circuits['Length'] = circuits['Length'].fillna(mean_length)  #
Rellenar los valores NaN con la media

# Imputar los valores faltantes en 'Turns' con la moda de la columna
circuits['Turns'] = circuits['Turns'].apply(pd.to_numeric,
errors='coerce')  # Convertir a numérico (NaN si no se puede)
mode_turns = circuits['Turns'].mode()[0]  # Calcular la moda de
'Turns'
circuits['Turns'] = circuits['Turns'].fillna(mode_turns)  # Rellenar
los valores NaN con la moda

# Mostrar el DataFrame actualizado
print(circuits)
```

```
    circuitId      circuitRef                                  name
location  \
0           1     albert_park         Albert Park Grand Prix Circuit
Melbourne
1           2          sepang           Sepang International Circuit
Kuala Lumpur
2           3         bahrain          Bahrain International Circuit
```

```
Sakhir
3           4       catalunya         Circuit de Barcelona-Catalunya
Montmeló
4           5        istanbul                        Istanbul Park
Istanbul
5           6          monaco                    Circuit de Monaco
Monte-Carlo
6           7      villeneuve              Circuit Gilles Villeneuve
Montreal
7           8     magny_cours          Circuit de Nevers Magny-Cours
Magny Cours
8           9     silverstone                  Silverstone Circuit
Silverstone
9          10  hockenheimring                       Hockenheimring
Hockenheim
10         11     hungaroring                          Hungaroring
Budapest
11         12        valencia               Valencia Street Circuit
Valencia
12         13             spa           Circuit de Spa-Francorchamps
Spa
13         14           monza           Autodromo Nazionale di Monza
Monza
14         15      marina_bay              Marina Bay Street Circuit
Marina Bay
15         16            fuji                        Fuji Speedway
Oyama
16         17        shanghai         Shanghai International Circuit
Shanghai
17         18      interlagos             Autódromo José Carlos Pace
São Paulo
18         19    indianapolis            Indianapolis Motor Speedway
Indianapolis
19         20     nurburgring                        Nürburgring
Nürburg
20         21           imola          Autodromo Enzo e Dino Ferrari
Imola
21         22          suzuka                      Suzuka Circuit
Suzuka
22         80           vegas         Las Vegas Strip Street Circuit
Las Vegas
23         24      yas_marina                  Yas Marina Circuit
Abu Dhabi
24         25          galvez          Autódromo Juan y Oscar Gálvez
Buenos Aires
25         26           jerez                   Circuito de Jerez
Jerez de la Frontera
26         27         estoril                Autódromo do Estoril
Estoril
```

```
27          28          okayama           Okayama International Circuit
Okayama
28          29         adelaide                 Adelaide Street Circuit
Adelaide
29          30          kyalami                                  Kyalami
Midrand
30          31        donington                        Donington Park
Castle Donington
31          32        rodriguez           Autódromo Hermanos Rodríguez
Mexico City
32          33          phoenix                  Phoenix street circuit
Phoenix
33          34           ricard                     Circuit Paul Ricard
Le Castellet
34          35          yeongam           Korean International Circuit
Yeongam County
35          36      jacarepagua  Autódromo Internacional Nelson Piquet
Rio de Janeiro
36          37          detroit                  Detroit Street Circuit
Detroit
37          38     brands_hatch                            Brands Hatch
Kent
38          39        zandvoort                Circuit Park Zandvoort
Zandvoort
39          40           zolder                                 Zolder
Heusden-Zolder
40          41            dijon                           Dijon-Prenois
Dijon
41          42           dallas                              Fair Park
Dallas
42          43       long_beach                             Long Beach
California
43          44        las_vegas              Las Vegas Street Circuit
Nevada
44          45           jarama                                 Jarama
Madrid
45          46     watkins_glen                           Watkins Glen
New York State
46          47       anderstorp                  Scandinavian Raceway
Anderstorp
47          48          mosport          Mosport International Raceway
Ontario
48          49         montjuic                               Montjuïc
Barcelona
49          50         nivelles                      Nivelles-Baulers
Brussels
50          51          charade                       Charade Circuit
Clermont-Ferrand
51          52        tremblant                Circuit Mont-Tremblant
```

```
Quebec
52          53          essarts                        Rouen-Les-Essarts
Rouen
53          54           lemans                                 Le Mans
Le Mans
54          55            reims                             Reims-Gueux
Reims
55          56           george                  Prince George Circuit
Eastern Cape Province
56          57          zeltweg                                Zeltweg
Styria
57          58          aintree                                Aintree
Liverpool
58          59         boavista                   Circuito da Boavista
Oporto
59          60        riverside        Riverside International Raceway
California
60          61            avus                                   AVUS
Berlin
61          62         monsanto                  Monsanto Park Circuit
Lisbon
62          63          sebring          Sebring International Raceway
Florida
63          64         ain-diab                             Ain Diab
Casablanca
64          65          pescara                       Pescara Circuit
Pescara
65          66       bremgarten                    Circuit Bremgarten
Bern
66          67        pedralbes                   Circuit de Pedralbes
Barcelona
67          68            buddh             Buddh International Circuit
Uttar Pradesh
68          69         americas                Circuit of the Americas
Austin
69          70   red_bull_ring                          Red Bull Ring
Spielberg
70          71            sochi                        Sochi Autodrom
Sochi
71          73             baku                       Baku City Circuit
Baku
72          75         portimao     Autódromo Internacional do Algarve
Portimão
73          76          mugello   Autodromo Internazionale del Mugello
Mugello
74          77           jeddah                Jeddah Corniche Circuit
Jeddah
75          78           losail           Losail International Circuit
Al Daayen
```

```
76         79           miami          Miami International Autodrome
Miami

          country       lat         lng   alt  \
0       Australia -37.84970  144.968000    10
1        Malaysia   2.76083  101.738000    18
2         Bahrain  26.03250   50.510600     7
3           Spain  41.57000    2.261110   109
4          Turkey  40.95170   29.405000   130
5          Monaco  43.73470    7.420560     7
6          Canada  45.50000  -73.522800    13
7          France  46.86420    3.163610   228
8              UK  52.07860   -1.016940   153
9         Germany  49.32780    8.565830   103
10        Hungary  47.57890   19.248600   264
11          Spain  39.45890   -0.331667     4
12        Belgium  50.43720    5.971390   401
13          Italy  45.61560    9.281110   162
14      Singapore   1.29140  103.864000    18
15          Japan  35.37170  138.927000   583
16          China  31.33890  121.220000     5
17         Brazil -23.70360  -46.699700   785
18            USA  39.79500  -86.234700   223
19        Germany  50.33560    6.947500   578
20          Italy  44.34390   11.716700    37
21          Japan  34.84310  136.541000    45
22  United States  36.11470 -115.173000   642
23            UAE  24.46720   54.603100     3
24      Argentina -34.69430  -58.459300     8
25          Spain  36.70830   -6.034170    37
26       Portugal  38.75060   -9.394170   130
27          Japan  34.91500  134.221000   266
28      Australia -34.92720  138.617000    58
29   South Africa -25.98940   28.076700  1460
30             UK  52.83060   -1.375280    88
31         Mexico  19.40420  -99.090700  2227
32            USA  33.44790 -112.075000   345
33         France  43.25060    5.791670   432
34          Korea  34.73330  126.417000     0
35         Brazil -22.97560  -43.395000  1126
36            USA  42.32980  -83.040100   177
37             UK  51.35690    0.263056   145
38    Netherlands  52.38880    4.540920     6
39        Belgium  50.98940    5.256940    36
40         France  47.36250    4.899130   484
41            USA  32.77740  -96.758700   139
42            USA  33.76510 -118.189000    12
43            USA  36.11620 -115.174000   639
44          Spain  40.61710   -3.585580   609
```

```
45            USA  42.33690  -76.927200   485
46         Sweden  57.26530   13.604200   153
47         Canada  44.04810  -78.675600   332
48          Spain  41.36640    2.151670    79
49        Belgium  50.62110    4.326940   139
50         France  45.74720    3.038890   790
51         Canada  46.18770  -74.609900   214
52         France  49.33060    1.004580    81
53         France  47.95000    0.224231    67
54         France  49.25420    3.930830    88
55   South Africa -33.04860   27.873600    15
56         Austria  47.20390   14.747800   676
57              UK  53.47690   -2.940560    20
58       Portugal  41.17050   -8.673250    28
59            USA  33.93700 -117.273000   470
60        Germany  52.48060   13.251400    53
61       Portugal  38.71970   -9.203060   158
62            USA  27.45470  -81.348300    18
63         Morocco  33.57860   -7.687500    19
64           Italy  42.47500   14.150800   129
65     Switzerland  46.95890    7.401940   551
66           Spain  41.39030    2.116670    85
67           India  28.34870   77.533100   194
68             USA  30.13280  -97.641100   161
69         Austria  47.21970   14.764700   678
70          Russia  43.40570   39.957800     2
71      Azerbaijan  40.37250   49.853300    -7
72        Portugal  37.22700   -8.626700   108
73           Italy  43.99750   11.371900   255
74   Saudi Arabia  21.63190   39.104400    15
75           Qatar  25.49000   51.454200    12
76             USA  25.95810  -80.238900     0


url     Length  Turns
0
http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit   5.278000
14.0
1
http://en.wikipedia.org/wiki/Sepang_International_Circuit   5.543000
15.0
2
http://en.wikipedia.org/wiki/Bahrain_International_Circuit   5.412000
15.0
3               http://en.wikipedia.org/wiki/Circuit_de_Barcelona-
Catalunya   4.657000   14.0
4
http://en.wikipedia.org/wiki/Istanbul_Park   5.338000   14.0
5
```

```
http://en.wikipedia.org/wiki/Circuit_de_Monaco   3.337000   19.0
6
http://en.wikipedia.org/wiki/Circuit_Gilles_Villeneuve   4.361000
14.0
7                http://en.wikipedia.org/wiki/Circuit_de_Nevers_Magny-
Cours   4.411000   17.0
8
http://en.wikipedia.org/wiki/Silverstone_Circuit   5.891000   18.0
9
http://en.wikipedia.org/wiki/Hockenheimring   4.574000   17.0
10
http://en.wikipedia.org/wiki/Hungaroring   4.381000   14.0
11
http://en.wikipedia.org/wiki/Valencia_Street_Circuit   5.419000   25.0

12                http://en.wikipedia.org/wiki/Circuit_de_Spa-
Francorchamps   7.004000   19.0
13
http://en.wikipedia.org/wiki/Autodromo_Nazionale_Monza   5.793000
11.0
14
http://en.wikipedia.org/wiki/Marina_Bay_Street_Circuit   4.940000
19.0
15
http://en.wikipedia.org/wiki/Fuji_Speedway   4.563000   16.0
16
http://en.wikipedia.org/wiki/Shanghai_International_Circuit   5.451000
16.0
17        http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Jos
%C3%A9_Carlos_Pace   4.309000   15.0
18
http://en.wikipedia.org/wiki/Indianapolis_Motor_Speedway   2.500000
4.0
19                              http://en.wikipedia.org/wiki/N
%C3%BCrburgring   5.148000   15.0
20
http://en.wikipedia.org/wiki/Autodromo_Enzo_e_Dino_Ferrari   4.909000
19.0
21
http://en.wikipedia.org/wiki/Suzuka_Circuit   5.807000   18.0
22
https://en.wikipedia.org/wiki/Las_Vegas_Grand_Prix#Circuit   4.988342
14.0
23
http://en.wikipedia.org/wiki/Yas_Marina_Circuit   5.281000   16.0
24    http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Oscar_Alfredo_G
%C3%A1lvez   4.259000   19.0
25
http://en.wikipedia.org/wiki/Circuito_Permanente_de_Jerez   4.428000
```

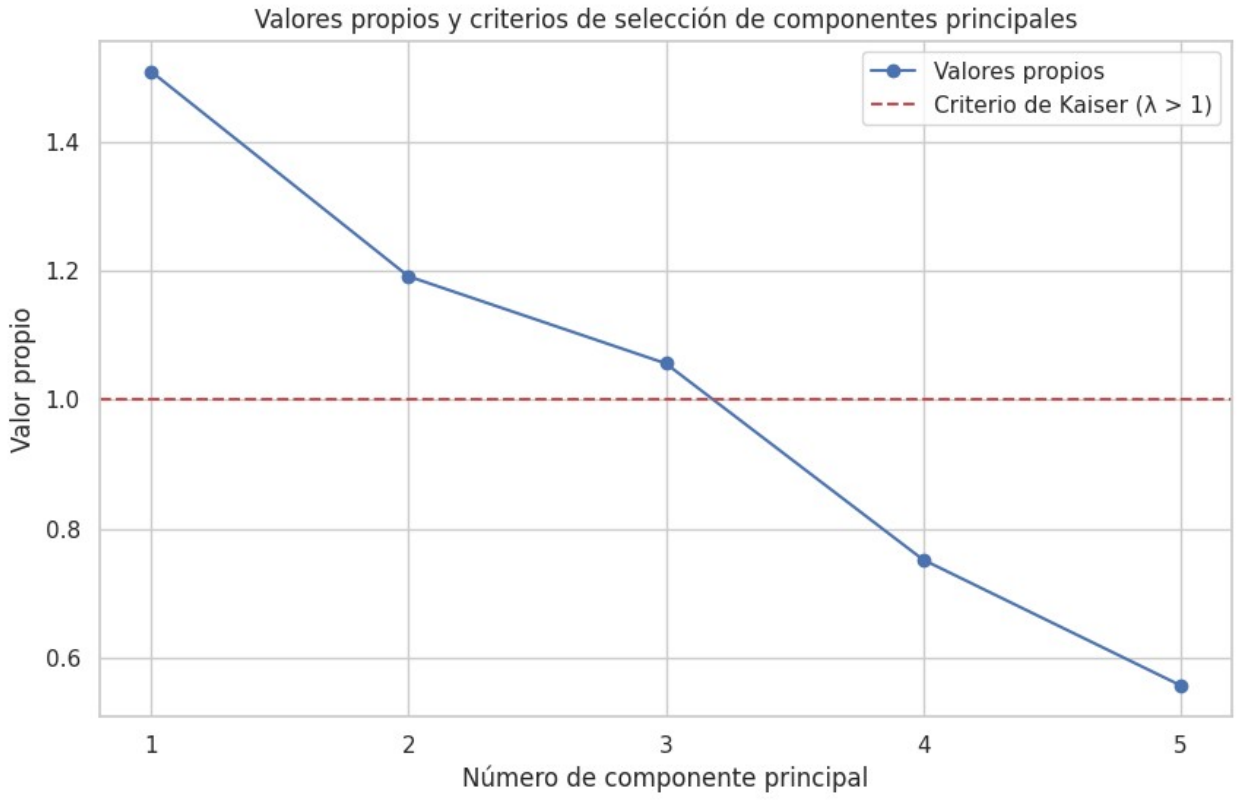
```
15.0
26                    http://en.wikipedia.org/wiki/Aut
%C3%B3dromo_do_Estoril   4.182000   14.0
27
http://en.wikipedia.org/wiki/TI_Circuit   3.703000   13.0
28
http://en.wikipedia.org/wiki/Adelaide_Street_Circuit   3.219000   14.0

29
http://en.wikipedia.org/wiki/Kyalami   4.529000   16.0
30
http://en.wikipedia.org/wiki/Donington_Park   4.020000   12.0
31      http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Hermanos_Rodr
%C3%ADguez   4.304000   17.0
32
http://en.wikipedia.org/wiki/Phoenix_street_circuit   2.312000   15.0

33
http://en.wikipedia.org/wiki/Paul_Ricard_Circuit   5.842000   15.0
34
http://en.wikipedia.org/wiki/Korean_International_Circuit   5.615000
18.0
35  http://en.wikipedia.org/wiki/Aut
%C3%B3dromo_Internacional_Nelson_Piquet   3.336000    7.0
36
http://en.wikipedia.org/wiki/Detroit_street_circuit   1.645000   10.0

37
http://en.wikipedia.org/wiki/Brands_Hatch   3.916000    9.0
38
http://en.wikipedia.org/wiki/Circuit_Zandvoort   4.259000   14.0
39
http://en.wikipedia.org/wiki/Zolder   4.010000   10.0
40                                  http://en.wikipedia.org/wiki/Dijon-
Prenois   3.801000   12.0
41
http://en.wikipedia.org/wiki/Fair_Park   4.988342   14.0
42
http://en.wikipedia.org/wiki/Long_Beach,_California   4.988342   14.0

43
http://en.wikipedia.org/wiki/Las_Vegas_Street_Circuit   3.853000
17.0
44
http://en.wikipedia.org/wiki/Circuito_Permanente_Del_Jarama   3.850000
14.0
45
http://en.wikipedia.org/wiki/Watkins_Glen_International   3.450000
11.0
46
```

```
http://en.wikipedia.org/wiki/Scandinavian_Raceway   4.025000    8.0
47
http://en.wikipedia.org/wiki/Mosport   3.957000   10.0
48                        http://en.wikipedia.org/wiki/Montju
%C3%AFc_circuit   3.791000   12.0
49                             http://en.wikipedia.org/wiki/Nivelles-
Baulers   3.724000    7.0
50
http://en.wikipedia.org/wiki/Charade_Circuit   3.975000   18.0
51                         http://en.wikipedia.org/wiki/Circuit_Mont-
Tremblant   2.621000   17.0
52                              http://en.wikipedia.org/wiki/Rouen-Les-
Essarts   5.543000   13.0
53
http://en.wikipedia.org/wiki/Circuit_de_la_Sarthe#Bugatti_Circuit
13.626000   38.0
54                                      http://en.wikipedia.org/wiki/Reims-
Gueux   8.302000    7.0
55
http://en.wikipedia.org/wiki/Prince_George_Circuit   3.920000    9.0
56
http://en.wikipedia.org/wiki/Zeltweg_Airfield   4.988342   14.0
57
http://en.wikipedia.org/wiki/Aintree_Motor_Racing_Circuit   4.828000
8.0
58
http://en.wikipedia.org/wiki/Circuito_da_Boavista   4.800000   23.0
59
http://en.wikipedia.org/wiki/Riverside_International_Raceway
3.300000    9.0
60
http://en.wikipedia.org/wiki/AVUS   2.660000    6.0
61
http://en.wikipedia.org/wiki/Monsanto_Park_Circuit   5.440000   14.0
62
http://en.wikipedia.org/wiki/Sebring_Raceway   3.741000   17.0
63                               http://en.wikipedia.org/wiki/Ain-
Diab_Circuit   7.603000   14.0
64
http://en.wikipedia.org/wiki/Pescara_Circuit  25.801000   14.0
65
http://en.wikipedia.org/wiki/Circuit_Bremgarten   7.280000   13.0
66
http://en.wikipedia.org/wiki/Pedralbes_Circuit   6.333000    6.0
67
http://en.wikipedia.org/wiki/Buddh_International_Circuit   5.125000
16.0
68
http://en.wikipedia.org/wiki/Circuit_of_the_Americas   3.426000   20.0
```

```
69
http://en.wikipedia.org/wiki/Red_Bull_Ring   4.318000   10.0
70
http://en.wikipedia.org/wiki/Sochi_Autodrom   2.313000   11.0
71
http://en.wikipedia.org/wiki/Baku_City_Circuit   6.003000   20.0
72
http://en.wikipedia.org/wiki/Algarve_International_Circuit   4.653000
15.0
73
http://en.wikipedia.org/wiki/Mugello_Circuit   5.245000   15.0
74
http://en.wikipedia.org/wiki/Jeddah_Street_Circuit   6.174000   27.0
75
http://en.wikipedia.org/wiki/Losail_International_Circuit   5.419000
16.0
76
http://en.wikipedia.org/wiki/Miami_International_Autodrome   3.363000
19.0
```

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Seleccionar las columnas relevantes para el PCA
features = ['lat', 'lng', 'alt', 'Length', 'Turns']  # Asegúrate de
que estas columnas estén en el DataFrame
X = circuits[features]

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Crear el modelo PCA y ajustarlo a los datos
pca = PCA(n_components=len(features))  # Número máximo de componentes
igual al número de características
pca.fit(X_scaled)

# Valores propios (explained variance)
explained_variance = pca.explained_variance_
explained_variance_ratio = pca.explained_variance_ratio_

# Gráfico de valores propios
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance) + 1), explained_variance,
marker='o', label='Valores propios')

# Líneas horizontales para los criterios de Kaiser y Jolliffe
```

```
plt.axhline(y=1, color='r', linestyle='--', label='Criterio de Kaiser
(λ > 1)')

# Etiquetas y leyenda
plt.title('Valores propios y criterios de selección de componentes
principales')
plt.xlabel('Número de componente principal')
plt.ylabel('Valor propio')
plt.xticks(range(1, len(explained_variance) + 1))
plt.legend()
plt.grid(True)

# Mostrar el gráfico
plt.show()

# Determinar el número de componentes según el criterio de Kaiser
kaiser_components = sum(explained_variance > 1)
print(f"Componentes seleccionados por Kaiser: {kaiser_components}")
```




```
Componentes seleccionados por Kaiser: 3
```

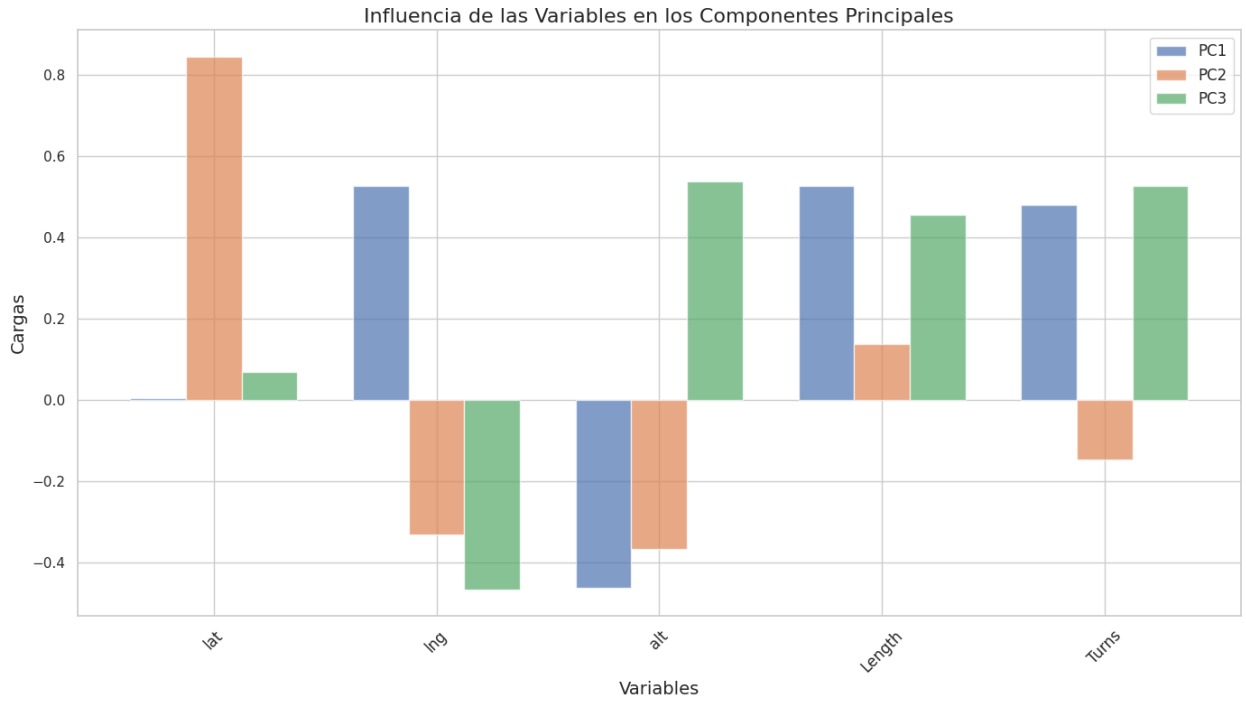
```
from sklearn.decomposition import PCA
import numpy as np
import pandas as pd
```

```python
# Crear PCA considerando el número máximo de componentes (igual al número de características)
pca_full = PCA(n_components=len(features))  # `features` es la lista de las columnas seleccionadas
pca_full.fit(X_scaled)

# Obtener la varianza explicada por cada componente
explained_variance_ratio = pca_full.explained_variance_ratio_

# Calcular la varianza explicada acumulada
cumulative_variance = np.cumsum(explained_variance_ratio)

# Crear un DataFrame para mostrar los resultados
variance_df = pd.DataFrame({
    'Componente': [f'PC{i+1}' for i in range(len(features))],
    'Varianza Explicada (%)': explained_variance_ratio * 100,
    'Varianza Acumulada (%)': cumulative_variance * 100
})

# Mostrar el DataFrame con formato redondeado
pd.options.display.float_format = '{:.2f}'.format  # Formato para mostrar los porcentajes con dos decimales
print(variance_df)
```

	Componente	Varianza Explicada (%)	Varianza Acumulada (%)
0	PC1	29.78	29.78
1	PC2	23.51	53.29
2	PC3	20.85	74.14
3	PC4	14.85	88.99
4	PC5	11.01	100.00

```python
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np

# Selección de las columnas relevantes para el PCA
features = ['lat', 'lng', 'alt', 'Length', 'Turns']
X = circuits[features]

# Normalización de los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Crear el modelo PCA y ajustarlo a los datos
n_components = 3  # Seleccionamos 3 componentes principales
pca = PCA(n_components=n_components)
pca.fit(X_scaled)
```

```
# Extraer las cargas de los componentes principales
loadings = pca.components_

# Crear un DataFrame para las cargas
loadings_df = pd.DataFrame(loadings.T, columns=[f'PC{i+1}' for i in
range(n_components)], index=features)

# Graficar las cargas de las variables para los 3 componentes
principales
plt.figure(figsize=(14, 8))

# Bar plot para las cargas de los tres primeros componentes
principales
bar_width = 0.25
index = np.arange(len(features))

# Graficar cada componente en el mismo gráfico
for i in range(n_components):
    plt.bar(index + i * bar_width, loadings_df[f'PC{i+1}'], bar_width,
label=f'PC{i+1}', alpha=0.7)

# Configurar etiquetas y leyenda
plt.xlabel('Variables', fontsize=14)
plt.ylabel('Cargas', fontsize=14)
plt.title('Influencia de las Variables en los Componentes
Principales', fontsize=16)
plt.xticks(index + bar_width, loadings_df.index, rotation=45,
fontsize=12)
plt.legend(fontsize=12)
plt.tight_layout()
plt.show()

# Expresión en función de las variables de cada componente principal
for i in range(n_components):
    print(f"\nExpresión del Componente Principal {i+1}:")
    expression = " + ".join([f"{round(loadings[i, j], 2)} *
{features[j]}" for j in range(len(features))])
    print(expression)
```

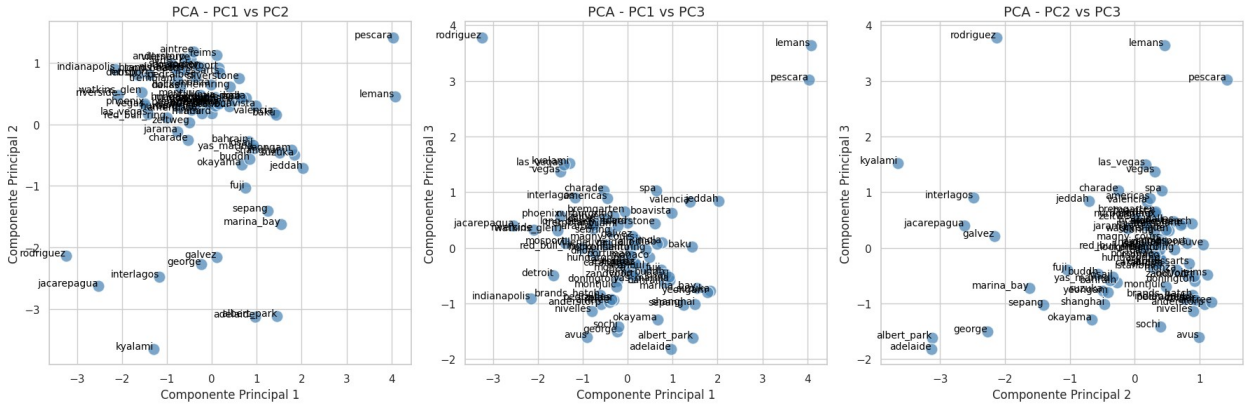



```
Expresión del Componente Principal 1:
0.0 * lat + 0.53 * lng + -0.46 * alt + 0.53 * Length + 0.48 * Turns

Expresión del Componente Principal 2:
0.85 * lat + -0.33 * lng + -0.37 * alt + 0.14 * Length + -0.15 * Turns

Expresión del Componente Principal 3:
0.07 * lat + -0.47 * lng + 0.54 * alt + 0.46 * Length + 0.53 * Turns
```

```python
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Configuración de estilo general
sns.set_theme(style="whitegrid")

# Asumimos que la tabla 'circuits' ya está cargada y que 'circuitRef'
es la columna con los nombres
features = ['lat', 'lng', 'alt', 'Length', 'Turns']
X = circuits[features]

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Crear el modelo PCA y ajustarlo a los datos
```

```
pca = PCA(n_components=3)  # Seleccionamos 3 componentes para
visualización
X_pca = pca.fit_transform(X_scaled)

# Extraer los nombres de los circuitos
circuit_names = circuits['circuitRef']

# Crear un DataFrame con los componentes principales y los nombres de
los circuitos
pca_df = pd.DataFrame(X_pca, columns=['PC1', 'PC2', 'PC3'])
pca_df['name'] = circuit_names

# Tamaño del texto para etiquetas
label_fontsize = 10

# Crear un gráfico de dispersión para cada combinación de componentes
principales en la misma fila
fig, axs = plt.subplots(1, 3, figsize=(18, 6), sharey=False)

# Configuración común para los gráficos
point_color = "steelblue"  # Color uniforme para todos los puntos

# PC1 vs PC2
axs[0].scatter(pca_df['PC1'], pca_df['PC2'], color=point_color,
alpha=0.7, edgecolor='w', s=150)
for i, row in pca_df.iterrows():
    axs[0].text(row['PC1'], row['PC2'], row['name'],
fontsize=label_fontsize, ha='right', color='black')
axs[0].set_xlabel('Componente Principal 1', fontsize=12)
axs[0].set_ylabel('Componente Principal 2', fontsize=12)
axs[0].set_title('PCA - PC1 vs PC2', fontsize=14)
axs[0].grid(True)

# PC1 vs PC3
axs[1].scatter(pca_df['PC1'], pca_df['PC3'], color=point_color,
alpha=0.7, edgecolor='w', s=150)
for i, row in pca_df.iterrows():
    axs[1].text(row['PC1'], row['PC3'], row['name'],
fontsize=label_fontsize, ha='right', color='black')
axs[1].set_xlabel('Componente Principal 1', fontsize=12)
axs[1].set_ylabel('Componente Principal 3', fontsize=12)
axs[1].set_title('PCA - PC1 vs PC3', fontsize=14)
axs[1].grid(True)

# PC2 vs PC3
axs[2].scatter(pca_df['PC2'], pca_df['PC3'], color=point_color,
alpha=0.7, edgecolor='w', s=150)
for i, row in pca_df.iterrows():
    axs[2].text(row['PC2'], row['PC3'], row['name'],
fontsize=label_fontsize, ha='right', color='black')
```

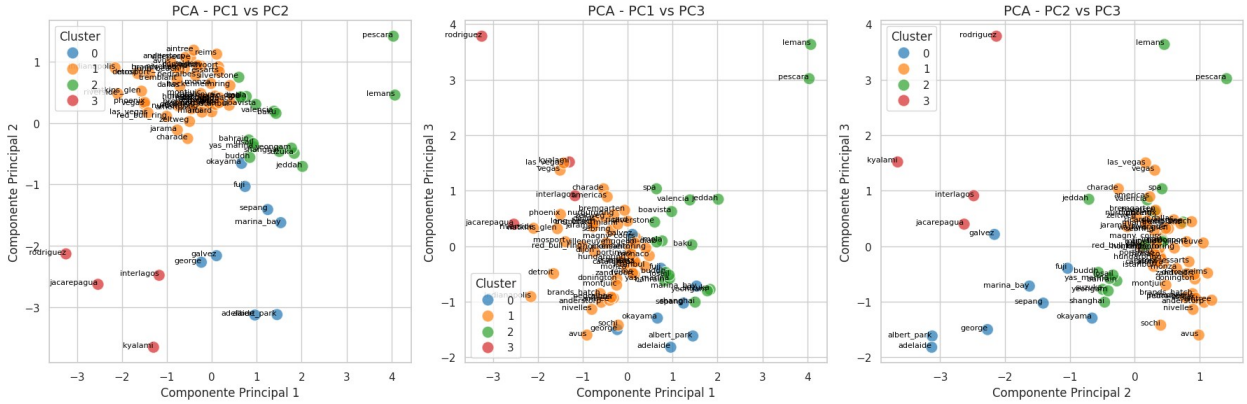
```
axs[2].set_xlabel('Componente Principal 2', fontsize=12)
axs[2].set_ylabel('Componente Principal 3', fontsize=12)
axs[2].set_title('PCA - PC2 vs PC3', fontsize=14)
axs[2].grid(True)

# Ajustar el diseño para que los gráficos se alineen correctamente
plt.tight_layout()
plt.show()
```




```
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Configuración de estilo general
sns.set_theme(style="whitegrid")

# Asumimos que la tabla 'circuits' ya está cargada y que 'circuitRef'
es la columna con los nombres
features = ['lat', 'lng', 'alt', 'Length', 'Turns']
X = circuits[features]

# Normalizar los datos
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Crear el modelo PCA y ajustarlo a los datos
pca = PCA(n_components=3)  # Seleccionamos 3 componentes para
visualización
X_pca = pca.fit_transform(X_scaled)

# Extraer los nombres de los circuitos
circuit_names = circuits['circuitRef']

# Crear un DataFrame con los componentes principales y los nombres de
```

```
los circuitos
pca_df = pd.DataFrame(X_pca, columns=['PC1', 'PC2', 'PC3'])
pca_df['name'] = circuit_names

# Aplicar K-Means para clustering
n_clusters = 4  # Número de clústeres deseado
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
pca_df['Cluster'] = kmeans.fit_predict(X_pca)

# Paleta de colores para los clústeres
palette = sns.color_palette('tab10', n_clusters)

# Tamaño del texto para etiquetas
label_fontsize = 8

# Crear un gráfico de dispersión para cada combinación de componentes
principales
fig, axs = plt.subplots(1, 3, figsize=(18, 6), sharey=False)

# Configuración común para los gráficos
scatter_kwargs = dict(alpha=0.7, edgecolor='w', s=150)

# PC1 vs PC2
sns.scatterplot(data=pca_df, x='PC1', y='PC2', hue='Cluster',
palette=palette, ax=axs[0], **scatter_kwargs)
for i, row in pca_df.iterrows():
    axs[0].text(row['PC1'], row['PC2'], row['name'],
fontsize=label_fontsize, ha='right', color='black')
axs[0].set_xlabel('Componente Principal 1', fontsize=12)
axs[0].set_ylabel('Componente Principal 2', fontsize=12)
axs[0].set_title('PCA - PC1 vs PC2', fontsize=14)

# PC1 vs PC3
sns.scatterplot(data=pca_df, x='PC1', y='PC3', hue='Cluster',
palette=palette, ax=axs[1], **scatter_kwargs)
for i, row in pca_df.iterrows():
    axs[1].text(row['PC1'], row['PC3'], row['name'],
fontsize=label_fontsize, ha='right', color='black')
axs[1].set_xlabel('Componente Principal 1', fontsize=12)
axs[1].set_ylabel('Componente Principal 3', fontsize=12)
axs[1].set_title('PCA - PC1 vs PC3', fontsize=14)

# PC2 vs PC3
sns.scatterplot(data=pca_df, x='PC2', y='PC3', hue='Cluster',
palette=palette, ax=axs[2], **scatter_kwargs)
for i, row in pca_df.iterrows():
    axs[2].text(row['PC2'], row['PC3'], row['name'],
fontsize=label_fontsize, ha='right', color='black')
axs[2].set_xlabel('Componente Principal 2', fontsize=12)
axs[2].set_ylabel('Componente Principal 3', fontsize=12)
```

```
axs[2].set_title('PCA - PC2 vs PC3', fontsize=14)

# Ajustar el diseño para que los gráficos se alineen correctamente
plt.tight_layout()
plt.show()
```



#PERCEPTRON MULTICAPA

¿Quedará nuestro corredor entre las primeras tres posiciones a partir de un entrenamiento en el circuito?

Este análisis esta hecho desde el punto de vista del equipo en general, se aplican múltiples variables al análisis para ver si el corredor quedará en podio o no. A raíz de lo que nos otorgue el perceptron, podremos ver si vamos bien encaminados o no, necesitando así más entrenamiento por parte del conductor o tal vez una mejora en el vehículo.

```
def preprocess_and_export():
    # Crear experiencia del piloto en años
    drivers['experience_years'] = 2023 -
pd.to_datetime(drivers['dob']).dt.year

    # Unir las tablas relevantes
    merged = results.merge(races, on='raceId') \
                    .merge(circuits, on='circuitId') \
                    .merge(drivers, on='driverId') \
                    .merge(driver_standings, on=['raceId',
'driverId']) \
                    .merge(constructor_standings, on=['raceId',
'constructorId'])

    # Crear etiquetas
    merged['is_podium'] = merged['positionOrder'].apply(lambda x: 1 if
x <= 3 else 0)
    merged['is_finished'] = merged['statusId'].apply(lambda x: 1 if
status.loc[status['statusId'] == x, 'status'].iloc[0] == 'Finished'
else 0)
```

```
    # Exportar la tabla a un archivo CSV
    merged.to_csv("merged_table.csv", index=False)

    print("La tabla fusionada ha sido exportada a
'merged_table.csv'.")
    return merged

# Llamar a la función
merged_table = preprocess_and_export()
```

```
La tabla fusionada ha sido exportada a 'merged_table.csv'.
```

```
import pandas as pd

# Configuración para mostrar más columnas y filas
pd.set_option('display.max_columns', None)  # Mostrar todas las
columnas
pd.set_option('display.max_rows', 100)      # Mostrar hasta 100 filas

# Visualizar una parte de la tabla en consola
print(merged_table)
```

```
       resultId  raceId  driverId  constructorId number_x  grid
position_x positionText_x  \
0             1      18         1              1       22     1
1              1
1             2      18         2              2        3     5
2              2
2             3      18         3              3        7     7
3              3
3             4      18         4              4        5    11
4              4
4             5      18         5              1       23     3
5              5
...         ...     ...       ...            ...      ...   ...
...            ...
24373     26520    1132       839            214       31    18
16             16
24374     26521    1132       815              9       11     0
17             17
24375     26522    1132       855             15       24    14
18             18
24376     26523    1132       847            131       63     1
NaN              R
24377     26524    1132       842            214       10    19
NaN              W

       positionOrder  points_x  laps       time_x milliseconds
fastestLap rank fastestLapTime  \
0                  1      10.0    58  1:34:50.616      5690616
```

```
39    2       1:27.452
1                2       8.0    58       +5.478      5696094
41    3       1:27.739
2                3       6.0    58       +8.163      5698779
41    5       1:28.090
3                4       5.0    58      +17.181      5707797
58    7       1:28.603
4                5       4.0    58      +18.014      5708630
43    1       1:27.418
...              ...      ...   ...          ...          ...
...  ...           ...
24373            16       0.0    50          NaN          NaN
46   16       1:30.875
24374            17       0.0    50          NaN          NaN
50    6       1:29.707
24375            18       0.0    50          NaN          NaN
43   17       1:31.014
24376            19       0.0    33          NaN          NaN
3    19      1:31.298
24377            20       0.0     0          NaN          NaN
NaN    0           NaN

      fastestLapSpeed  statusId  year  round  circuitId
name_x        date  \
0              218.300         1  2008      1          1  Australian
Grand Prix  2008-03-16
1              217.586         1  2008      1          1  Australian
Grand Prix  2008-03-16
2              216.719         1  2008      1          1  Australian
Grand Prix  2008-03-16
3              215.464         1  2008      1          1  Australian
Grand Prix  2008-03-16
4              218.385         1  2008      1          1  Australian
Grand Prix  2008-03-16
...                ...       ...   ...    ...        ...
...          ...
24373          233.371        12  2024     12          9     British
Grand Prix  2024-07-07
24374          236.409        12  2024     12          9     British
Grand Prix  2024-07-07
24375          233.014        12  2024     12          9     British
Grand Prix  2024-07-07
24376          232.289        34  2024     12          9     British
Grand Prix  2024-07-07
24377              NaN         6  2024     12          9     British
Grand Prix  2024-07-07

         time_y
url_x    fp1_date  fp1_time  \
```

```
0      04:30:00
http://en.wikipedia.org/wiki/2008_Australian_Grand_Prix         NaN
NaN
1      04:30:00
http://en.wikipedia.org/wiki/2008_Australian_Grand_Prix         NaN
NaN
2      04:30:00
http://en.wikipedia.org/wiki/2008_Australian_Grand_Prix         NaN
NaN
3      04:30:00
http://en.wikipedia.org/wiki/2008_Australian_Grand_Prix         NaN
NaN
4      04:30:00
http://en.wikipedia.org/wiki/2008_Australian_Grand_Prix         NaN
NaN
...         ...                                                    .
..         ...       ...
24373  14:00:00
https://en.wikipedia.org/wiki/2024_British_Grand_Prix  2024-07-05
11:30:00
24374  14:00:00
https://en.wikipedia.org/wiki/2024_British_Grand_Prix  2024-07-05
11:30:00
24375  14:00:00
https://en.wikipedia.org/wiki/2024_British_Grand_Prix  2024-07-05
11:30:00
24376  14:00:00
https://en.wikipedia.org/wiki/2024_British_Grand_Prix  2024-07-05
11:30:00
24377  14:00:00
https://en.wikipedia.org/wiki/2024_British_Grand_Prix  2024-07-05
11:30:00

         fp2_date  fp2_time    fp3_date  fp3_time  quali_date
quali_time sprint_date sprint_time  \
0             NaN       NaN         NaN       NaN         NaN
NaN         NaN         NaN
1             NaN       NaN         NaN       NaN         NaN
NaN         NaN         NaN
2             NaN       NaN         NaN       NaN         NaN
NaN         NaN         NaN
3             NaN       NaN         NaN       NaN         NaN
NaN         NaN         NaN
4             NaN       NaN         NaN       NaN         NaN
NaN         NaN         NaN
...           ...       ...         ...       ...         ...        .
..         ...         ...
24373  2024-07-05  15:00:00  2024-07-06  10:30:00  2024-07-06
14:00:00         NaN         NaN
```

```
24374  2024-07-05  15:00:00  2024-07-06  10:30:00  2024-07-06
14:00:00         NaN         NaN
24375  2024-07-05  15:00:00  2024-07-06  10:30:00  2024-07-06
14:00:00         NaN         NaN
24376  2024-07-05  15:00:00  2024-07-06  10:30:00  2024-07-06
14:00:00         NaN         NaN
24377  2024-07-05  15:00:00  2024-07-06  10:30:00  2024-07-06
14:00:00         NaN         NaN

        circuitRef                          name_y     location
country      lat        lng  \
0      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia -37.8497  144.96800
1      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia -37.8497  144.96800
2      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia -37.8497  144.96800
3      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia -37.8497  144.96800
4      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia -37.8497  144.96800
...            ...                                ...          ...
...      ...        ...
24373  silverstone             Silverstone Circuit  Silverstone
UK  52.0786   -1.01694
24374  silverstone             Silverstone Circuit  Silverstone
UK  52.0786   -1.01694
24375  silverstone             Silverstone Circuit  Silverstone
UK  52.0786   -1.01694
24376  silverstone             Silverstone Circuit  Silverstone
UK  52.0786   -1.01694
24377  silverstone             Silverstone Circuit  Silverstone
UK  52.0786   -1.01694

       alt                                                    url_y
Surface  Length  Turns  \
0       10  http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit
Asphalt   5.278   14.0
1       10  http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit
Asphalt   5.278   14.0
2       10  http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit
Asphalt   5.278   14.0
3       10  http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit
Asphalt   5.278   14.0
4       10  http://en.wikipedia.org/wiki/Melbourne_Grand_Prix_Circuit
Asphalt   5.278   14.0
...    ...                                                     ...
...     ...    ...
24373  153           http://en.wikipedia.org/wiki/Silverstone_Circuit
```

```
Asphalt   5.891   18.0
24374  153          http://en.wikipedia.org/wiki/Silverstone_Circuit
Asphalt   5.891   18.0
24375  153          http://en.wikipedia.org/wiki/Silverstone_Circuit
Asphalt   5.891   18.0
24376  153          http://en.wikipedia.org/wiki/Silverstone_Circuit
Asphalt   5.891   18.0
24377  153          http://en.wikipedia.org/wiki/Silverstone_Circuit
Asphalt   5.891   18.0

       Surface_encoded   driverRef number_y code  forename     surname
dob nationality  \
0                    0    hamilton       44  HAM     Lewis    Hamilton
1985-01-07     British
1                    0    heidfeld      NaN  HEI      Nick    Heidfeld
1977-05-10      German
2                    0     rosberg        6  ROS      Nico     Rosberg
1985-06-27      German
3                    0      alonso       14  ALO  Fernando      Alonso
1981-07-29     Spanish
4                    0  kovalainen      NaN  KOV    Heikki  Kovalainen
1981-10-19     Finnish
...                ...         ...      ...  ...       ...         ...
...         ...
24373                0        ocon       31  OCO   Esteban        Ocon
1996-09-17      French
24374                0       perez       11  PER    Sergio       Pérez
1990-01-26     Mexican
24375                0        zhou       24  ZHO    Guanyu        Zhou
1999-05-30     Chinese
24376                0     russell       63  RUS    George     Russell
1998-02-15     British
24377                0       gasly       10  GAS    Pierre       Gasly
1996-02-07      French

                                                                    url
experience_years  \
0                     http://en.wikipedia.org/wiki/Lewis_Hamilton
38
1                      http://en.wikipedia.org/wiki/Nick_Heidfeld
46
2                       http://en.wikipedia.org/wiki/Nico_Rosberg
38
3                     http://en.wikipedia.org/wiki/Fernando_Alonso
42
4                   http://en.wikipedia.org/wiki/Heikki_Kovalainen
42
...                                                           ...
...
```

```
24373                    http://en.wikipedia.org/wiki/Esteban_Ocon
27
24374                http://en.wikipedia.org/wiki/Sergio_P%C3%A9rez
33
24375                     http://en.wikipedia.org/wiki/Zhou_Guanyu
24
24376  http://en.wikipedia.org/wiki/George_Russell_(racing_driver)
25
24377                    http://en.wikipedia.org/wiki/Pierre_Gasly
27

       driverStandingsId  points_y  position_y positionText_y  wins_x
constructorStandingsId  \
0                      1      10.0           1              1       1
1
1                      2       8.0           2              2       0
2
2                      3       6.0           3              3       0
3
3                      4       5.0           4              4       0
4
4                      5       4.0           5              5       0
1
...                  ...       ...         ...            ...     ...
...
24373              72867       3.0          18             18       0
28852
24374              72852     118.0           6              6       0
28843
24375              72861       0.0          19             19       0
28848
24376              72855     111.0           7              7       1
28845
24377              72868       6.0          15             15       0
28852

       points  position positionText  wins_y  is_podium  is_finished
0        14.0         1            1       1          1            1
1         8.0         3            3       0          1            1
2         9.0         2            2       0          1            1
3         5.0         4            4       0          0            1
4        14.0         1            1       1          0            1
...       ...       ...          ...     ...        ...          ...
24373     9.0         8            8       0          0            0
24374   373.0         1            1       7          0            0
24375     0.0        10           10       0          0            0
24376   221.0         4            4       2          0            0
24377     9.0         8            8       0          0            0

[24378 rows x 68 columns]
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report

# Preprocesamiento de datos
def preprocess_data():
    # Crear experiencia del piloto en años
    drivers['experience_years'] = 2023 -
pd.to_datetime(drivers['dob']).dt.year - 18

    # Unir las tablas relevantes
    merged = results.merge(races, on='raceId') \
                    .merge(circuits, on='circuitId') \
                    .merge(drivers, on='driverId') \
                    .merge(driver_standings, on=['raceId',
'driverId']) \
                    .merge(constructor_standings, on=['raceId',
'constructorId']) \
                    .merge(constructors[['constructorId', 'name']],
on='constructorId', how='left')  # Unir la tabla constructors

    # Renombrar la columna 'name' a 'constructor_name'
    merged['constructor_name'] = merged['name']
    merged = merged.drop('name', axis=1)  # Eliminar la columna 'name'
original

    # Reemplazar '/N' y '\\N' por 0 en todas las columnas del
DataFrame
    merged.replace({'/N': 0, '\\N': 0}, inplace=True)

    # Asegurarse de que las columnas milliseconds y fastestlapspeed
estén en formato numérico, reemplazando valores no válidos por 0
    merged['milliseconds'] = pd.to_numeric(merged['milliseconds'],
errors='coerce').fillna(0)
    merged['fastestLapSpeed'] =
pd.to_numeric(merged['fastestLapSpeed'], errors='coerce').fillna(0)

    # Crear etiqueta: si el piloto termina en podio (positionOrder <=
3)
    merged['is_podium'] = merged['positionOrder'].apply(lambda x: 1 if
x <= 3 else 0)

    # Seleccionar características relevantes para la predicción
    X = merged[['grid','nationality', 'experience_years',
'circuitRef', 'constructor_name', 'milliseconds', 'fastestLapSpeed']]
# Añadido 'fastestlapspeed'
    y_podium = merged['is_podium']
```

```
    # Codificar datos categóricos (nationality, circuitRef,
constructor_name)
    encoder = OneHotEncoder()
    encoded = encoder.fit_transform(X[['nationality', 'circuitRef',
'constructor_name']]).toarray()
    X = X.drop(['nationality', 'circuitRef', 'constructor_name'],
axis=1)
    X = np.hstack([X.values, encoded])

    # Escalar características numéricas
    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    return X, y_podium

# Preparar los datos
X, y_podium = preprocess_data()

# Dividir en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y_podium,
test_size=0.2, random_state=42)

# Crear y entrenar el modelo para predecir el podio
podium_model = MLPClassifier(hidden_layer_sizes=(64, 32),
activation='relu', max_iter=500, random_state=42)
podium_model.fit(X_train, y_train)

# Evaluar el modelo
y_pred = podium_model.predict(X_test)

# Mostrar rendimiento del modelo
print("Rendimiento del modelo (¿Terminará en el podio?):")
print(classification_report(y_test, y_pred))
```

```
Rendimiento del modelo (¿Terminará en el podio?):
              precision    recall  f1-score   support

           0       0.95      0.95      0.95      4286
           1       0.64      0.62      0.63       590

    accuracy                           0.91      4876
   macro avg       0.80      0.79      0.79      4876
weighted avg       0.91      0.91      0.91      4876
```

```
def preprocess_data():
    # Crear experiencia del piloto en años
    drivers['experience_years'] = 2023 -
pd.to_datetime(drivers['dob']).dt.year - 18
```

```
    # Unir las tablas relevantes
    merged = results.merge(races, on='raceId') \
                    .merge(circuits, on='circuitId') \
                    .merge(drivers, on='driverId') \
                    .merge(driver_standings, on=['raceId',
'driverId']) \
                    .merge(constructor_standings, on=['raceId',
'constructorId']) \
                    .merge(constructors[['constructorId', 'name']],
on='constructorId', how='left')  # Unir la tabla constructors

    # Renombrar la columna 'name' a 'constructor_name'
    merged['constructor_name'] = merged['name']
    merged = merged.drop('name', axis=1)  # Eliminar la columna 'name'
original

    # Reemplazar '/N' y '\\N' por 0 en todas las columnas del
DataFrame
    merged.replace({'/N': 0, '\\N': 0}, inplace=True)

    # Asegurarse de que las columnas milliseconds y fastestlapspeed
estén en formato numérico, reemplazando valores no válidos por 0
    merged['milliseconds'] = pd.to_numeric(merged['milliseconds'],
errors='coerce').fillna(0)
    merged['fastestLapSpeed'] =
pd.to_numeric(merged['fastestLapSpeed'], errors='coerce').fillna(0)

    # Crear etiqueta: si el piloto termina en podio (positionOrder <=
3)
    merged['is_podium'] = merged['positionOrder'].apply(lambda x: 1 if
x <= 3 else 0)

    # Seleccionar características relevantes para la predicción
    X = merged[['grid','nationality', 'experience_years',
'circuitRef', 'constructor_name', 'milliseconds', 'fastestLapSpeed']]
# Añadido 'fastestlapspeed'
    y_podium = merged['is_podium']

    # Codificar datos categóricos (nationality, circuitRef,
constructor_name)
    encoder = OneHotEncoder()
    encoded = encoder.fit_transform(X[['nationality', 'circuitRef',
'constructor_name']]).toarray()
    X = X.drop(['nationality', 'circuitRef', 'constructor_name'],
axis=1)
    X = np.hstack([X.values, encoded])

    # Escalar características numéricas
    scaler = StandardScaler()
    X = scaler.fit_transform(X)
```

```
    return X, y_podium, encoder, scaler

# Preparar los datos
X, y_podium, encoder, scaler = preprocess_data()

# Crear un individuo de ejemplo
individual = {
    'grid': 3,
    'nationality': 'British',
    'experience_years': 21,
    'circuitRef': 'albert_park',
    'constructor_name': 'McLaren',
    'milliseconds': 5690616,
    'fastestLapSpeed': 218.300
}

# Convertir a DataFrame para consistencia con los datos originales
individual_df = pd.DataFrame([individual])

# Preprocesar el individuo
encoded_individual = encoder.transform(individual_df[['nationality',
'circuitRef', 'constructor_name']]).toarray()
individual_df = individual_df.drop(['nationality', 'circuitRef',
'constructor_name'], axis=1)
individual_array = np.hstack([individual_df.values,
encoded_individual])
individual_scaled = scaler.transform(individual_array)

# Hacer la predicción
prediction = podium_model.predict(individual_scaled)

# Mostrar el resultado
print(f"Predicción: {'Podio' if prediction[0] == 1 else 'No podio'}")
```

```
Predicción: Podio
```

```
# Crear un individuo de ejemplo
individual = {
    'grid': 1,
    'nationality': 'Finnish',
    'experience_years': 25,
    'circuitRef': 'albert_park',
    'constructor_name': 'Toro Rosso',
    'milliseconds': 5708630,
    'fastestLapSpeed': 218.385
}

# Convertir a DataFrame para consistencia con los datos originales
individual_df = pd.DataFrame([individual])
```

```
# Preprocesar el individuo
encoded_individual = encoder.transform(individual_df[['nationality',
'circuitRef', 'constructor_name']]).toarray()
individual_df = individual_df.drop(['nationality', 'circuitRef',
'constructor_name'], axis=1)
individual_array = np.hstack([individual_df.values,
encoded_individual])
individual_scaled = scaler.transform(individual_array)

# Hacer la predicción
prediction = podium_model.predict(individual_scaled)

# Mostrar el resultado
print(f"Predicción: {'Podio' if prediction[0] == 1 else 'No podio'}")
```

```
Predicción: No podio
```

ANEXO 2

Carga de archivos

```
# Subir el archivo ZIP
from google.colab import files
import pandas as pd
import zipfile
import os

# Subir el archivo ZIP con los datasets
uploaded = files.upload()

# Descomprimir el archivo ZIP
zip_filename = next(iter(uploaded))  # Obtener el nombre del archivo
subido
with zipfile.ZipFile(zip_filename, 'r') as zip_ref:
    zip_ref.extractall('./datasets')  # Extraer los archivos a la
carpeta './datasets'

# Definir la ruta donde se descomprimieron los archivos
path = './datasets/'

# Cargar los 14 datasets en DataFrames
circuits = pd.read_csv(os.path.join(path, 'circuits.csv'))
constructor_results = pd.read_csv(os.path.join(path,
'constructor_results.csv'))
constructor_standings = pd.read_csv(os.path.join(path,
'constructor_standings.csv'))
constructors = pd.read_csv(os.path.join(path, 'constructors.csv'))
driver_standings = pd.read_csv(os.path.join(path,
'driver_standings.csv'))
drivers = pd.read_csv(os.path.join(path, 'drivers.csv'))
lap_times = pd.read_csv(os.path.join(path, 'lap_times.csv'))
pit_stops = pd.read_csv(os.path.join(path, 'pit_stops.csv'))
qualifying = pd.read_csv(os.path.join(path, 'qualifying.csv'))
races = pd.read_csv(os.path.join(path, 'races.csv'))
results = pd.read_csv(os.path.join(path, 'results.csv'))
seasons = pd.read_csv(os.path.join(path, 'seasons.csv'))
sprint_results = pd.read_csv(os.path.join(path, 'sprint_results.csv'))
status = pd.read_csv(os.path.join(path, 'status.csv'))
```

```
<IPython.core.display.HTML object>
```

```
Saving archive.zip to archive (2).zip
```

Preparación árboles

```
import pandas as pd
import numpy as np
```

```
# Supongamos que ya tienes los datasets cargados como DataFrames
# pit_stops = pd.read_csv('path_to_pit_stops.csv')
# results = pd.read_csv('path_to_results.csv')
# qualifying = pd.read_csv('path_to_qualifying.csv')
# constructors_results =
pd.read_csv('path_to_constructors_results.csv')

# Filtrar las paradas que tengan menos de 60,000 milisegundos
pit_stops = pit_stops[pit_stops['milliseconds'] <= 60000]

# Seleccionar columnas de interés de pit_stops
pit_stops = pit_stops[['raceId', 'driverId', 'stop',
'milliseconds','lap']]

# Seleccionar columnas de interés de results
results = results[['raceId', 'driverId', 'grid',
'positionOrder','constructorId']]

# Calcular el número de paradas por piloto en cada carrera
pit_stops['num_stops'] = pit_stops.groupby(['raceId', 'driverId'])
['stop'].transform('count')

# Realizar la combinación mediante un left join con results
pit_stops_with_results = pd.merge(
    pit_stops,
    results,
    on=['raceId', 'driverId'],  # Claves para hacer la unión
    how='left'                  # Tipo de unión
)

# 1. Crear las categorías simplificadas basadas en la posición final
def categorize_position(position):
    if position in [1, 2, 3]:
        return '1_Podio'
    elif 4 <= position <= 10:
        return '2_Top 10 sin podio'
    else:
        return '3_Sin puntos'


# Aplicar la función de categorización
pit_stops_with_results['position_category'] =
pit_stops_with_results['positionOrder'].apply(categorize_position)

# Eliminar la columna 'stop'
pit_stops_with_results = pit_stops_with_results.drop(columns=['stop'])

# Calcular el percentil de cada parada dentro de su carrera
pit_stops_with_results['Percentil_parada'] =
pit_stops_with_results.groupby('raceId')['milliseconds'] \
```

```
    .transform(lambda x: x.rank(pct=True) - 1/x.count())

# Redondear los percentiles a dos decimales
pit_stops_with_results['Percentil_parada'] =
pit_stops_with_results['Percentil_parada'].round(2)

# Filtrar la tabla constructor_results para quedarnos solo con las
columnas necesarias
constructor_results_filtered = constructor_results[['raceId',
'constructorId', 'points']]

# Realizar el merge entre pit_stops_with_results y
constructor_results_filtered
pit_stops_with_results = pit_stops_with_results.merge(
    constructor_results_filtered,
    on=['raceId', 'constructorId'],
    how='left'
)

#Renombrar la columna de puntos para mayor claridad
pit_stops_with_results =
pit_stops_with_results.rename(columns={'points':
'points_constructor'})

# Reordenar las columnas del DataFrame
desired_columns = [
    'position_category',
    'positionOrder',
    'raceId',
    'constructorId',
    'driverId',
    'num_stops',
    'milliseconds',
    'Percentil_parada',
    'grid',
    'points_constructor'
]

# Aplicar el nuevo orden de las columnas
pit_stops_with_results = pit_stops_with_results[desired_columns]

pit_stops_with_results
```


```
{"summary":"{\n  \"name\": \"pit_stops_with_results\",\n  \"rows\":
10490,\n  \"fields\": [\n    {\n      \"column\":
\"position_category\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 3,\n        \"samples\":
[\n          \"3_Sin puntos\",\n          \"2_Top 10 sin podio\",\n
\"1_Podio\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"positionOrder\",\n      \"properties\": {\n        \"dtype\":
```

```
\"number\",\n        \"std\": 5,\n        \"min\": 1,\n
\"max\": 24,\n        \"num_unique_values\": 24,\n        \"samples\":
[\n          14,\n          13,\n          11\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"raceId\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 88,\n
\"min\": 841,\n        \"max\": 1132,\n        \"num_unique_values\":
273,\n        \"samples\": [\n          871,\n          969,\n
928\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"constructorId\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 85,\n        \"min\": 1,\n
\"max\": 215,\n        \"num_unique_values\": 23,\n
\"samples\": [\n          209,\n          15,\n          5\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"driverId\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 386,\n        \"min\": 1,\n
\"max\": 860,\n        \"num_unique_values\": 74,\n
\"samples\": [\n          13,\n          851,\n          5\
n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"num_stops\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 1,\n        \"min\": 1,\n
\"max\": 6,\n        \"num_unique_values\": 6,\n        \"samples\":
[\n          3,\n          2,\n          5\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"milliseconds\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
4600,\n        \"min\": 12897,\n        \"max\": 59555,\n
\"num_unique_values\": 6935,\n        \"samples\": [\n
27466,\n          25420,\n          23816\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Percentil_parada\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.28862890682762443,\n        \"min\": 0.0,\n        \"max\": 0.99,\n
\"num_unique_values\": 100,\n        \"samples\": [\n          0.45,\n
0.9,\n          0.34\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"grid\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 6,\n        \"min\": 0,\n        \"max\": 24,\n
\"num_unique_values\": 25,\n        \"samples\": [\n          22,\n
20,\n          12\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"points_constructor\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 12.802798352197454,\n        \"min\":
0.0,\n        \"max\": 66.0,\n        \"num_unique_values\": 53,\n
\"samples\": [\n          8.0,\n          29.0,\n          57.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
```

```
}\n    }\n  ]\
n}","type":"dataframe","variable_name":"pit_stops_with_results"}
```

RANDOM FOREST PRESENTADO EN LA MEMORIA

En la siguiente celda se muestra el Random Forest implementado para contestar a la pregunta: ¿Es fundamental tener rápidas paradas en boxes durante la carrera?

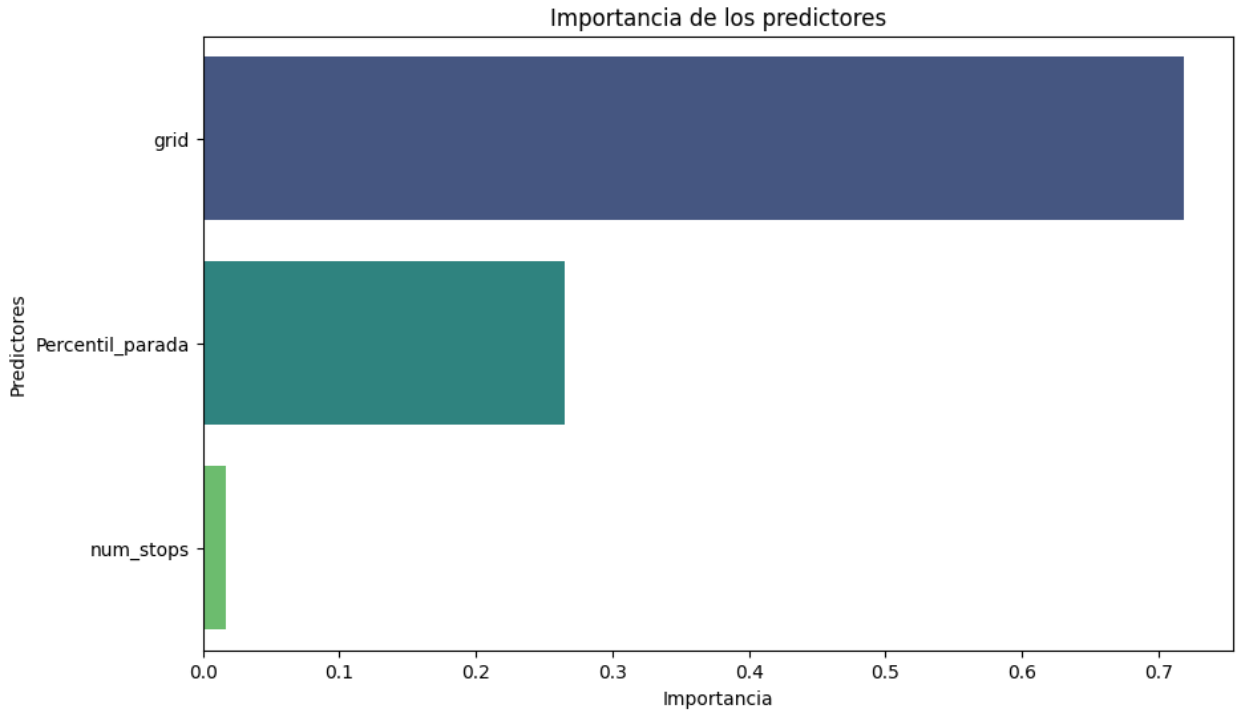
Se utiliza como base lo presentado en árboles de decisión anteriores, además se añade la técnica SMOTE, para hacer sobremuestreo en las clases minoritarias y finalmente se extraen resultados: matriz de confusión, accuracy, gráfico de importancia y gráfico de una de las 500 simulaciones de árboles

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier  # Usar
RandomForest
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns  # Para visualización de la importancia de las
variables
from imblearn.over_sampling import SMOTE  # Para balanceo de clases

# Dataset y preprocesamiento
df_tree = pit_stops_with_results[['position_category', 'num_stops',
'Percentil_parada', 'grid']]

# Mostrar la distribución de clases original
print("Distribución de la variable objetivo antes del balanceo:")
print(df_tree['position_category'].value_counts())

# Convertir la variable objetivo 'position_category' a valores
numéricos
label_encoder = LabelEncoder()
df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])

# Seleccionar las variables predictoras y la variable objetivo
X = df_tree[['num_stops', 'Percentil_parada', 'grid']]
y = df_tree['position_category_encoded']

# Dividir el dataset en conjunto de entrenamiento (train) y prueba
(test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# **Balancear el conjunto de entrenamiento con SMOTE**
smote = SMOTE(random_state=42)
```

```
X_train_balanced, y_train_balanced = smote.fit_resample(X_train,
y_train)

# Mostrar la distribución de clases después del balanceo
print("\nDistribución de la variable objetivo después del balanceo
(sobremuestreo con SMOTE):")
print(pd.Series(y_train_balanced).value_counts())

# Crear y entrenar el modelo de Random Forest
model = RandomForestClassifier(
    random_state=42,
    n_estimators=500,            # Usar 500 árboles para mayor
estabilidad
    max_depth=3,                # Mayor profundidad
    class_weight='balanced',     # Ajustar automáticamente los pesos
de las clases
    n_jobs=-1                    # Usar todos los núcleos para
entrenamiento más rápido
)
model.fit(X_train_balanced, y_train_balanced)

# Realizar predicciones
y_pred = model.predict(X_test)

# Evaluar el modelo
print("\nClassification Report:")
print(classification_report(y_test, y_pred,
target_names=label_encoder.classes_))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Visualizar la importancia de las variables predictoras
importances = model.feature_importances_
features = X.columns

# Crear un DataFrame para visualizar la importancia de los predictores
importance_df = pd.DataFrame({'Predictor': features, 'Importance':
importances})

# Ordenar de mayor a menor importancia
importance_df = importance_df.sort_values(by='Importance',
ascending=False)

# Graficar la importancia de los predictores
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Predictor', data=importance_df,
palette='viridis')
plt.title('Importancia de los predictores')
plt.xlabel('Importancia')
```

```
plt.ylabel('Predictores')
plt.show()

# (Opcional) Visualizar uno de los árboles individuales dentro del
Random Forest
from sklearn.tree import plot_tree

# Extraer uno de los árboles individuales del Random Forest
tree = model.estimators_[0]  # Tomar el primer árbol de los 500
plt.figure(figsize=(15, 8))
plot_tree(tree,
          feature_names=X.columns,
          class_names=label_encoder.classes_,
          filled=True,
          rounded=True,
          fontsize=10)
plt.title('Un árbol individual dentro del Random Forest')
plt.show()

Distribución de la variable objetivo antes del balanceo:
position_category
3_Sin puntos          5308
2_Top 10 sin podio    3644
1_Podio               1538
Name: count, dtype: int64

Distribución de la variable objetivo después del balanceo
(sobremuestreo con SMOTE):
position_category_encoded
1    4257
0    4257
2    4257
Name: count, dtype: int64

<ipython-input-6-fa732e4a7dde>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])


Classification Report:
                    precision    recall  f1-score   support

           1_Podio       0.51      0.81      0.63       307
2_Top 10 sin podio       0.54      0.46      0.50       740
      3_Sin puntos       0.76      0.71      0.74      1051
```

```
          accuracy                           0.64      2098
         macro avg       0.60      0.66      0.62      2098
      weighted avg       0.65      0.64      0.64      2098


Confusion Matrix:
[[250  43  14]
 [179 338 223]
 [ 60 241 750]]

<ipython-input-6-fa732e4a7dde>:68: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.barplot(x='Importance', y='Predictor', data=importance_df,
palette='viridis')
```



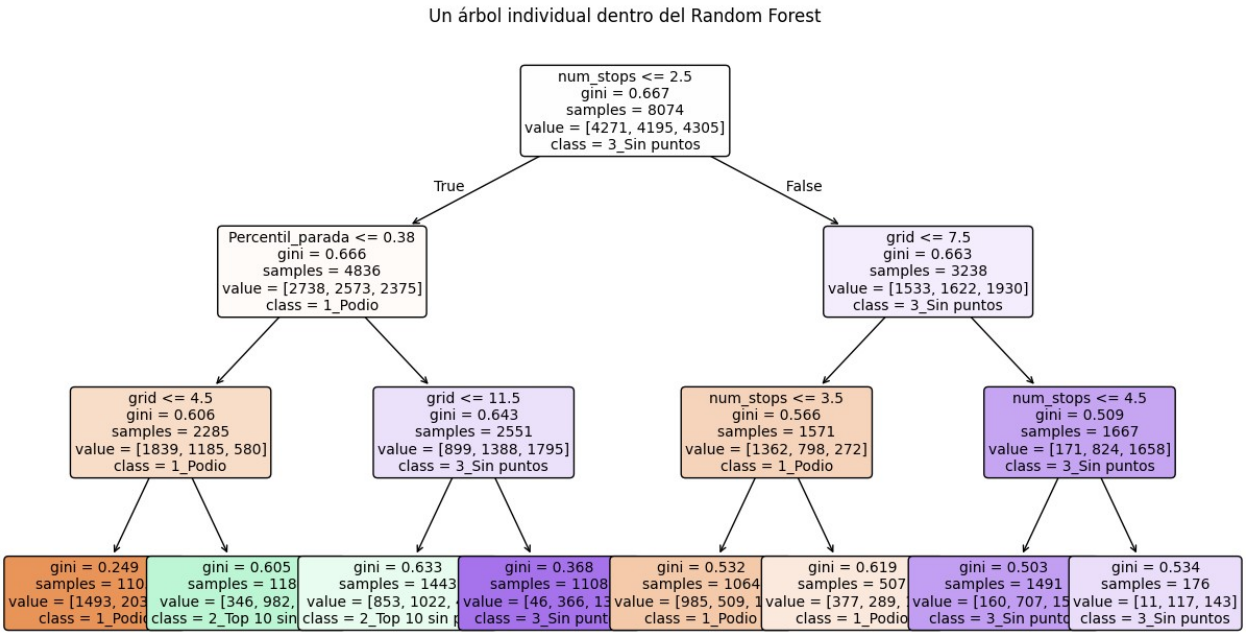
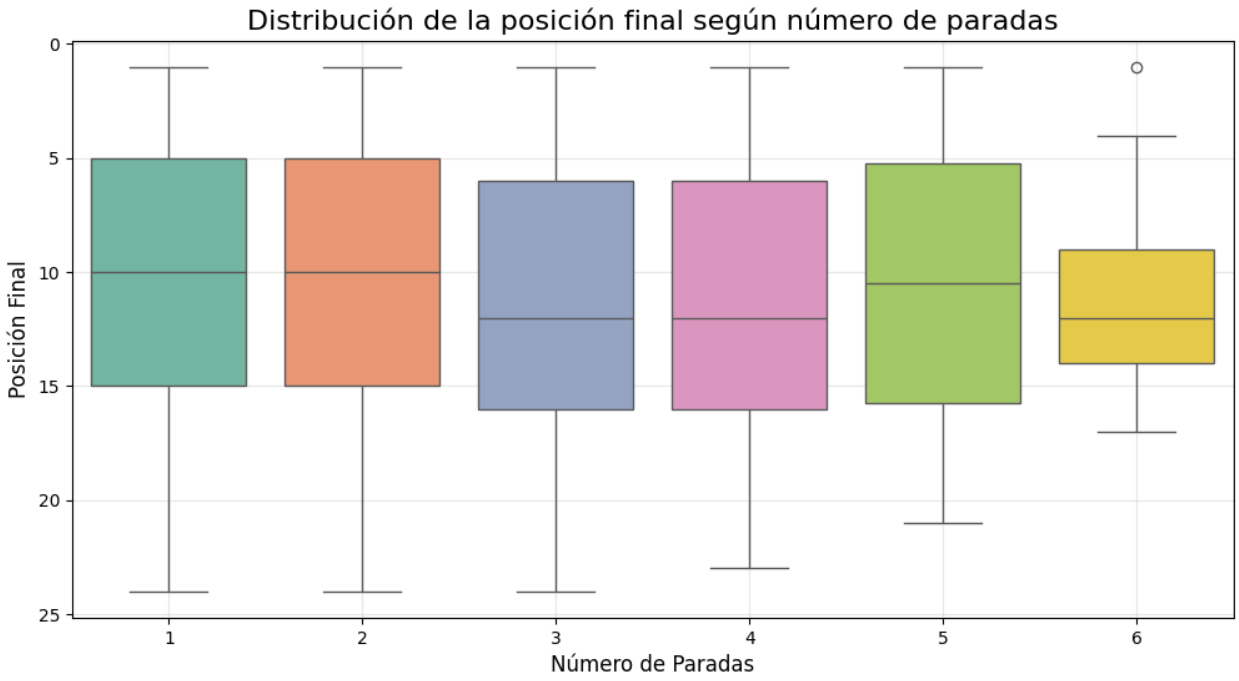


Gráfico boxplots número de paradas vs posición final

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Filtrar el dataset para incluir solo una fila por piloto y carrera
# Usamos drop_duplicates para mantener solo una fila por combinación
de raceId y driverId
unique_pit_stops_with_results =
pit_stops_with_results.drop_duplicates(subset=['raceId', 'driverId'])

# Diagrama de caja: distribución de la posición final según el número
de paradas
plt.figure(figsize=(12, 6))
sns.boxplot(
    data=unique_pit_stops_with_results,
    x='num_stops',
    y='positionOrder',
    palette='Set2'
)
plt.title('Distribución de la posición final según número de paradas',
fontsize=16)
plt.xlabel('Número de Paradas', fontsize=12)
plt.ylabel('Posición Final', fontsize=12)
plt.gca().invert_yaxis()  # Invertir el eje Y para reflejar que la
posición 1 es la mejor
plt.grid(alpha=0.3)
plt.show()
```

```
<ipython-input-7-8844a0833ead>:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(
```



Árboles adicionales

```
# 7. Seleccionar las columnas de interés para el árbol de
clasificación
df_tree =
pit_stops_with_results[['position_category','num_stops','Percentil_par
ada','grid','points_constructor']]
#df_tree =
pit_stops_with_results[['position_category','num_stops','Percentil_par
ada']]

df_tree
```


```
{"summary":"{\n  \"name\": \"df_tree\",\n  \"rows\": 10490,\n
\"fields\": [\n    {\n      \"column\": \"position_category\",\n
\"properties\": {\n        \"dtype\": \"category\",\n
\"num_unique_values\": 3,\n        \"samples\": [\n          \"3_Sin
puntos\",\n          \"2_Top 10 sin podio\",\n          \"1_Podio\"\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"num_stops\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
```

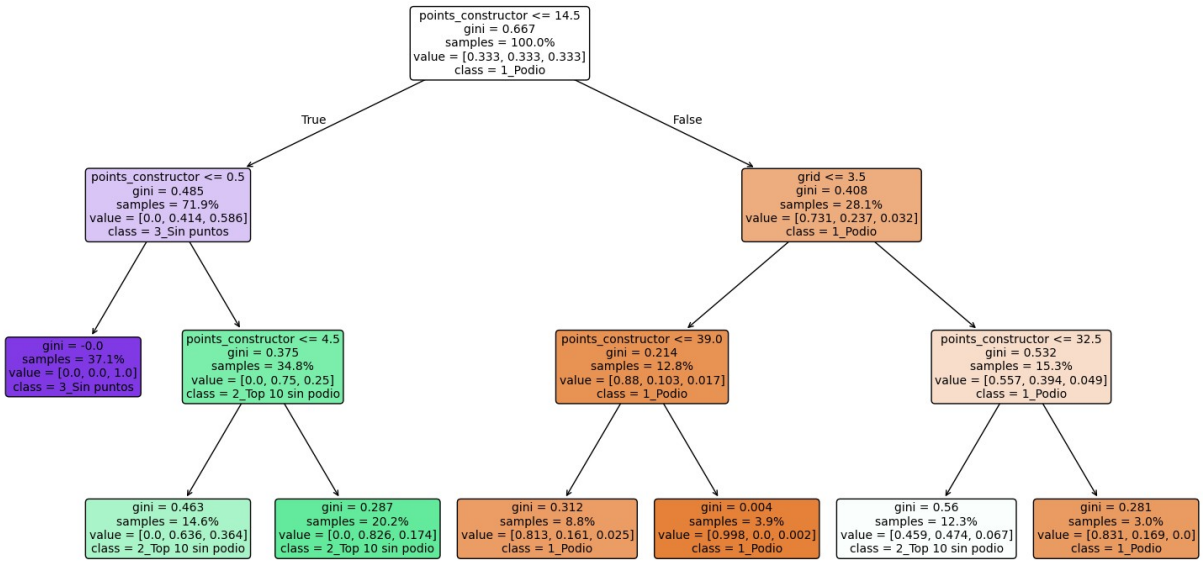
```
1,\n        \"min\": 1,\n        \"max\": 6,\n
\"num_unique_values\": 6,\n        \"samples\": [\n          3,\n
2,\n          5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Percentil_parada\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.28862890682762443,\n        \"min\":
0.0,\n        \"max\": 0.99,\n        \"num_unique_values\": 100,\n
\"samples\": [\n          0.45,\n          0.9,\n          0.34\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"grid\",\n      \"properties\":
{\n        \"dtype\": \"number\",\n        \"std\": 6,\n
\"min\": 0,\n        \"max\": 24,\n        \"num_unique_values\": 25,\
n        \"samples\": [\n          22,\n          20,\n          12\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"points_constructor\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
12.802798352197454,\n        \"min\": 0.0,\n        \"max\": 66.0,\n
\"num_unique_values\": 53,\n        \"samples\": [\n          8.0,\n
29.0,\n          57.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_tree"}
```


MODELO ÁRBOL DF_TREE CON 4 PREDICTORES

```
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, plot_tree
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder

df_tree =
pit_stops_with_results[['position_category','num_stops','Percentil_par
ada','grid','points_constructor']]

# 1. Preprocesamiento
# Convertir la variable objetivo 'position_category' a valores
numéricos
label_encoder = LabelEncoder()
df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])

# Seleccionar las variables predictoras y la variable objetivo
X = df_tree[['num_stops', 'Percentil_parada', 'grid',
'points_constructor']]
y = df_tree['position_category_encoded']

# 2. Dividir el dataset en conjunto de entrenamiento (train) y prueba
```

```
(test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 3. Crear y entrenar el modelo
model = DecisionTreeClassifier(
    random_state=42,
    max_depth=3,
    class_weight='balanced'  # Ajustar automáticamente los pesos según
la distribución de las clases
)
model.fit(X_train, y_train)

# 4. Realizar predicciones
y_pred = model.predict(X_test)

# 5. Evaluar el modelo
print("Classification Report:")
print(classification_report(y_test, y_pred,
target_names=label_encoder.classes_))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Graficar el árbol
plt.figure(figsize=(15, 8))  # Ajustar el tamaño de la figura
plot_tree(model,
          feature_names=X.columns,
          class_names=label_encoder.classes_,
          filled=True,
          rounded=True,
          fontsize=10,
          proportion=True)
plt.tight_layout()
plt.show()


<ipython-input-10-0e5982d2721b>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])

Classification Report:
                   precision    recall  f1-score   support
```

```
           1_Podio       0.75      0.79      0.77       307
2_Top 10 sin podio       0.65      0.90      0.75       740
      3_Sin puntos       1.00      0.71      0.83      1051

          accuracy                           0.79      2098
         macro avg       0.80      0.80      0.78      2098
      weighted avg       0.84      0.79      0.79      2098

Confusion Matrix:
[[241  66   0]
 [ 72 668   0]
 [  8 301 742]]
```



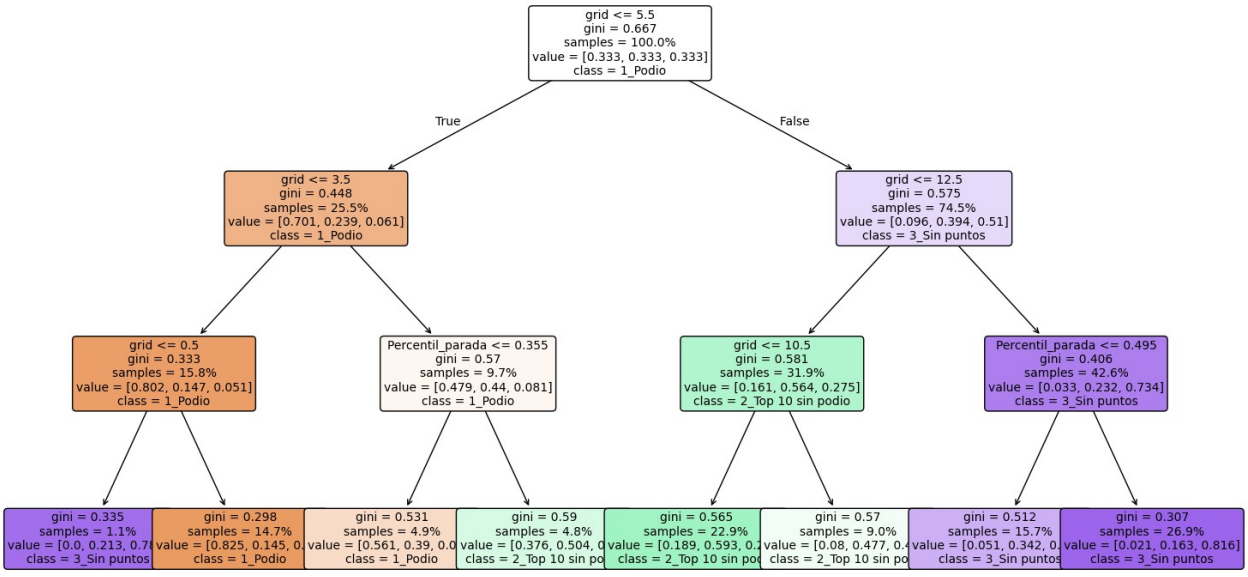
MODELO DF_TREE CON 3 PREDICTORES

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

# Dataset y preprocesamiento
df_tree = pit_stops_with_results[['position_category', 'num_stops',
'Percentil_parada','grid']]

# Convertir la variable objetivo 'position_category' a valores
numéricos
```

```
label_encoder = LabelEncoder()
df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])

# Seleccionar las variables predictoras y la variable objetivo
X = df_tree[['num_stops', 'Percentil_parada','grid']]
y = df_tree['position_category_encoded']

# Dividir el dataset en conjunto de entrenamiento (train) y prueba
(test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Crear y entrenar el modelo, ajustando los pesos para manejar
desbalanceo
model = DecisionTreeClassifier(
    random_state=42,
    max_depth=3,
    class_weight='balanced'  # Ajustar automáticamente los pesos según
la distribución de las clases
)
model.fit(X_train, y_train)

# Realizar predicciones
y_pred = model.predict(X_test)

# Evaluar el modelo
print("Classification Report:")
print(classification_report(y_test, y_pred,
target_names=label_encoder.classes_))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Graficar el árbol
plt.figure(figsize=(15, 8))  # Ajustar el tamaño de la figura
plot_tree(model,
          feature_names=X.columns,
          class_names=label_encoder.classes_,
          filled=True,
          rounded=True,
          fontsize=10,
          proportion=True)
plt.tight_layout()
plt.show()
```

```
<ipython-input-11-58a52a1b0040>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])

Classification Report:
                    precision    recall  f1-score   support

           1_Podio       0.56      0.75      0.64       307
2_Top 10 sin podio       0.53      0.56      0.55       740
      3_Sin puntos       0.79      0.69      0.74      1051

          accuracy                           0.65      2098
         macro avg       0.63      0.67      0.64      2098
      weighted avg       0.67      0.65      0.66      2098

Confusion Matrix:
[[231  64  12]
 [152 413 175]
 [ 32 295 724]]
```



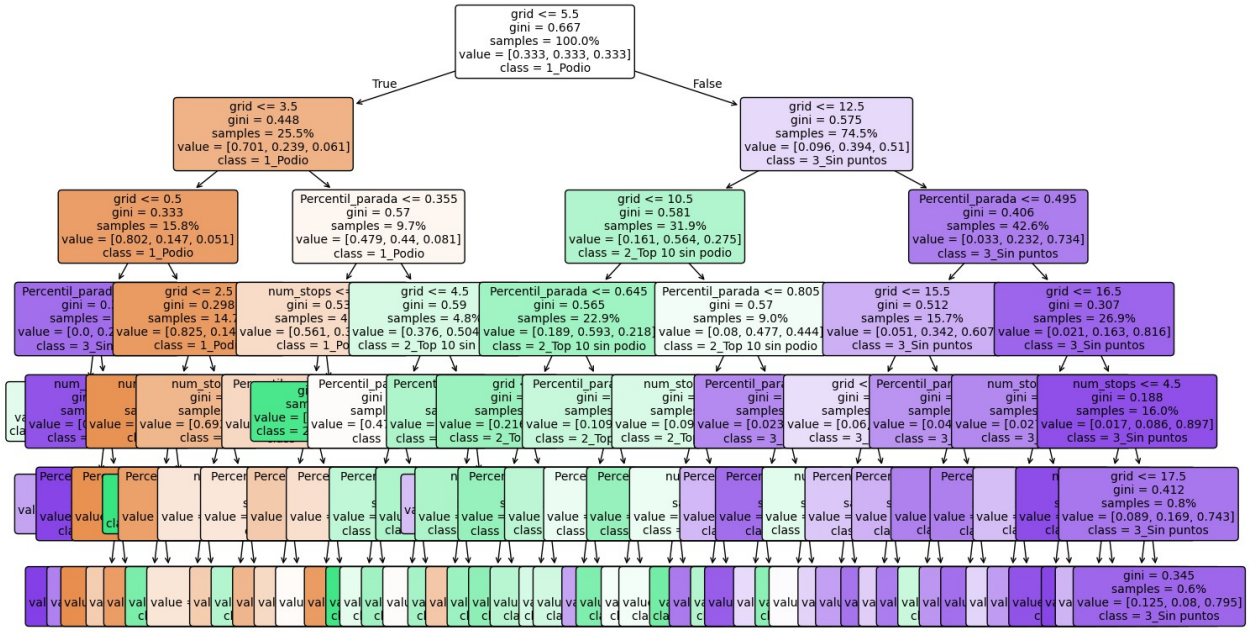

```python
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
```

```
# Dataset y preprocesamiento
df_tree = pit_stops_with_results[['position_category', 'num_stops',
'Percentil_parada','grid']]

# Convertir la variable objetivo 'position_category' a valores
numéricos
label_encoder = LabelEncoder()
df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])

# Seleccionar las variables predictoras y la variable objetivo
X = df_tree[['num_stops', 'Percentil_parada','grid']]
y = df_tree['position_category_encoded']

# Dividir el dataset en conjunto de entrenamiento (train) y prueba
(test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Configurar el rango de hiperparámetros para GridSearchCV
param_grid = {
    'max_depth': [4, 5, 6, None],  # Profundidad máxima del árbol
    'min_samples_split': [2, 5, 10, 20],  # Mínimo número de muestras
para dividir un nodo
    'min_samples_leaf': [1, 2, 5, 10],  # Mínimo número de muestras en
una hoja
    'class_weight': ['balanced'],  # Pesos para manejar desbalanceo
    'criterion': ['gini', 'entropy']  # Función de división
}

# Crear el modelo base
base_model = DecisionTreeClassifier(random_state=42)

# Realizar la búsqueda de los mejores hiperparámetros con GridSearchCV
grid_search = GridSearchCV(
    estimator=base_model,
    param_grid=param_grid,
    scoring='accuracy',  # Maximizar el accuracy
    cv=5,  # Validación cruzada con 5 particiones
    verbose=1,  # Mostrar progreso
    n_jobs=-1  # Usar todos los núcleos disponibles
)

# Ajustar el modelo
grid_search.fit(X_train, y_train)

# Imprimir los mejores hiperparámetros encontrados
print("Mejores hiperparámetros:", grid_search.best_params_)
```

```python
# Evaluar el mejor modelo en el conjunto de prueba
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test)

print("\nClassification Report:")
print(classification_report(y_test, y_pred,
target_names=label_encoder.classes_))

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Graficar el árbol optimizado
plt.figure(figsize=(15, 8))
plot_tree(best_model,
          feature_names=X.columns,
          class_names=label_encoder.classes_,
          filled=True,
          rounded=True,
          fontsize=10,
          proportion=True)
plt.tight_layout()
plt.show()
```

```
Fitting 5 folds for each of 128 candidates, totalling 640 fits

<ipython-input-12-38eea1c3aedb>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df_tree['position_category_encoded'] =
label_encoder.fit_transform(df_tree['position_category'])

Mejores hiperparámetros: {'class_weight': 'balanced', 'criterion':
'gini', 'max_depth': 6, 'min_samples_leaf': 5, 'min_samples_split':
20}

Classification Report:
                    precision    recall  f1-score   support

           1_Podio       0.56      0.78      0.66       307
2_Top 10 sin podio       0.56      0.56      0.56       740
      3_Sin puntos       0.80      0.71      0.75      1051

          accuracy                           0.67      2098
         macro avg       0.64      0.68      0.66      2098
      weighted avg       0.68      0.67      0.67      2098
```

```
Confusion Matrix:
[[240  54  13]
 [148 415 177]
 [ 37 269 745]]
```



Red Neuronal probada pero no mostrada en la memoria

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# Preprocesamiento
# Convertir las variables categóricas en numéricas (One-Hot Encoding
para 'Categoria_parada')
X = pit_stops_with_results[['num_stops', 'milliseconds',
'Percentil_parada', 'grid']]

# Escalar las variables numéricas
scaler = StandardScaler()
X[['num_stops', 'milliseconds', 'Percentil_parada', 'grid']] =
scaler.fit_transform(X[['num_stops', 'milliseconds',
'Percentil_parada', 'grid']])

# Variable objetivo (PositionOrder)
y = pit_stops_with_results['positionOrder']
```

```python
# Dividir el dataset en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Definir el modelo de la red neuronal
model = Sequential()

# Capa de entrada: se asume que X tiene 4 columnas después del One-Hot
Encoding
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))  #
Primera capa oculta
model.add(Dropout(0.2))  # Regularización con Dropout para evitar
sobreajuste
model.add(Dense(32, activation='relu'))  # Segunda capa oculta
model.add(Dense(1, activation='linear'))  # Capa de salida con
activación lineal

# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')

# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_test, y_test))

# Evaluación
y_pred = model.predict(X_test)

# Métricas
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

```
Epoch 1/100

<ipython-input-13-7870194cbf95>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  X[['num_stops', 'milliseconds', 'Percentil_parada', 'grid']] =
scaler.fit_transform(X[['num_stops', 'milliseconds',
'Percentil_parada', 'grid']])
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py
:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to
a layer. When using Sequential models, prefer using an `Input(shape)`
object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 86.1889 - val_loss:
21.1345
Epoch 2/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 22.2654 - val_loss:
19.3463
Epoch 3/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 19.5571 - val_loss:
18.6369
Epoch 4/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 19.8562 - val_loss:
18.1062
Epoch 5/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 19.5163 - val_loss:
17.9964
Epoch 6/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 19.2739 - val_loss:
18.0163
Epoch 7/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.1433 - val_loss:
17.9679
Epoch 8/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.0760 - val_loss:
17.9514
Epoch 9/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2413 - val_loss:
17.9442
Epoch 10/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.1049 - val_loss:
17.7757
Epoch 11/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.2868 - val_loss:
17.8103
Epoch 12/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7770 - val_loss:
17.9818
Epoch 13/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.9738 - val_loss:
17.7804
Epoch 14/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.9211 - val_loss:
17.7521
Epoch 15/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5543 - val_loss:
17.7629
Epoch 16/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 19.3792 - val_loss:
17.7974
Epoch 17/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.5354 - val_loss:
17.7092
```

```
Epoch 18/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 19.2841 - val_loss:
17.7557
Epoch 19/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.3418 - val_loss:
17.7112
Epoch 20/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.1124 - val_loss:
17.7711
Epoch 21/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.6041 - val_loss:
17.6704
Epoch 22/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.9987 - val_loss:
17.6826
Epoch 23/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.7375 - val_loss:
17.8774
Epoch 24/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.3192 - val_loss:
17.8371
Epoch 25/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.5694 - val_loss:
17.6913
Epoch 26/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4487 - val_loss:
17.6115
Epoch 27/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1423 - val_loss:
17.6851
Epoch 28/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5534 - val_loss:
17.6040
Epoch 29/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 19.0393 - val_loss:
17.7970
Epoch 30/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 19.0026 - val_loss:
17.6844
Epoch 31/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 17.8797 - val_loss:
17.6285
Epoch 32/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.3881 - val_loss:
17.7889
Epoch 33/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0207 - val_loss:
17.6762
Epoch 34/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4613 - val_loss:
```

```
17.6441
Epoch 35/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7007 - val_loss:
17.7928
Epoch 36/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8294 - val_loss:
17.6037
Epoch 37/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3264 - val_loss:
17.7028
Epoch 38/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8675 - val_loss:
17.7662
Epoch 39/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.6407 - val_loss:
17.5863
Epoch 40/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1590 - val_loss:
17.6411
Epoch 41/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8947 - val_loss:
17.6376
Epoch 42/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7584 - val_loss:
17.5916
Epoch 43/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 18.6919 - val_loss:
17.6487
Epoch 44/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.3733 - val_loss:
17.6910
Epoch 45/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.9102 - val_loss:
17.5644
Epoch 46/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 17.7550 - val_loss:
17.9685
Epoch 47/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9206 - val_loss:
18.0751
Epoch 48/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.9097 - val_loss:
17.6082
Epoch 49/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5443 - val_loss:
17.6832
Epoch 50/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3045 - val_loss:
17.5730
Epoch 51/100
```

```
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4915 - val_loss:
17.5852
Epoch 52/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3183 - val_loss:
17.6921
Epoch 53/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7219 - val_loss:
17.6766
Epoch 54/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0494 - val_loss:
17.7230
Epoch 55/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1926 - val_loss:
17.6611
Epoch 56/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 17.9952 - val_loss:
17.6065
Epoch 57/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.2062 - val_loss:
17.7306
Epoch 58/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.6649 - val_loss:
17.5578
Epoch 59/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.2066 - val_loss:
17.7000
Epoch 60/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.8529 - val_loss:
17.5872
Epoch 61/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8957 - val_loss:
17.6262
Epoch 62/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0397 - val_loss:
17.6128
Epoch 63/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.9321 - val_loss:
17.6705
Epoch 64/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.2324 - val_loss:
17.6254
Epoch 65/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3009 - val_loss:
17.6267
Epoch 66/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.2360 - val_loss:
17.7558
Epoch 67/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2596 - val_loss:
17.7537
```

```
Epoch 68/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7021 - val_loss:
17.6955
Epoch 69/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1232 - val_loss:
17.5660
Epoch 70/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8957 - val_loss:
17.6238
Epoch 71/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.5068 - val_loss:
17.5833
Epoch 72/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.0282 - val_loss:
17.6531
Epoch 73/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.2559 - val_loss:
17.6375
Epoch 74/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.5468 - val_loss:
17.6238
Epoch 75/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3437 - val_loss:
17.5279
Epoch 76/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.6587 - val_loss:
17.6096
Epoch 77/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1560 - val_loss:
17.5640
Epoch 78/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7085 - val_loss:
17.6542
Epoch 79/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0923 - val_loss:
17.6462
Epoch 80/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1814 - val_loss:
17.5450
Epoch 81/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4427 - val_loss:
17.5935
Epoch 82/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4242 - val_loss:
17.6015
Epoch 83/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.9526 - val_loss:
17.5730
Epoch 84/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7228 - val_loss:
```

```
17.6762
Epoch 85/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1727 - val_loss:
17.5125
Epoch 86/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 17.9001 - val_loss:
17.6998
Epoch 87/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 17.9037 - val_loss:
17.5502
Epoch 88/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.4769 - val_loss:
17.9346
Epoch 89/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1797 - val_loss:
17.5449
Epoch 90/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9342 - val_loss:
17.5494
Epoch 91/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8889 - val_loss:
17.6099
Epoch 92/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.8702 - val_loss:
17.5255
Epoch 93/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8915 - val_loss:
17.6145
Epoch 94/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5693 - val_loss:
17.5042
Epoch 95/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.6008 - val_loss:
17.8587
Epoch 96/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8303 - val_loss:
17.5564
Epoch 97/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4247 - val_loss:
17.6127
Epoch 98/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2529 - val_loss:
17.9432
Epoch 99/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.3656 - val_loss:
17.5691
Epoch 100/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.0135 - val_loss:
17.8200
```

```
66/66 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
Mean Squared Error: 17.820046457517773
```

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd

# ============================
# 🔹 Preprocesamiento
# ============================

# Convertir las variables categóricas en numéricas (One-Hot Encoding para 'Categoria_parada')
X = pit_stops_with_results[['num_stops', 'milliseconds', 'Percentil_parada', 'grid']]

# Escalar las variables numéricas
scaler = StandardScaler()
X[['num_stops', 'milliseconds', 'Percentil_parada', 'grid']] = scaler.fit_transform(X[['num_stops', 'milliseconds', 'Percentil_parada', 'grid']])

# Variable objetivo (PositionOrder)
y = pit_stops_with_results['positionOrder']

# Dividir el dataset en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# ============================
# 🔹 Definición de la red neuronal
# ============================

# Definir el modelo de la red neuronal
model = Sequential()

# Capa de entrada: se asume que X tiene 4 columnas después del One-Hot Encoding
model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))  # Primera capa oculta
model.add(Dropout(0.2))  # Regularización con Dropout para evitar sobreajuste
model.add(Dense(32, activation='relu'))  # Segunda capa oculta
model.add(Dense(1, activation='linear'))  # Capa de salida con activación lineal
```

```
# Compilar el modelo
model.compile(optimizer='adam', loss='mean_squared_error')

# ============================
# □ Entrenamiento del modelo
# ============================

# Entrenar el modelo
history = model.fit(X_train, y_train, epochs=100, batch_size=32,
validation_data=(X_test, y_test))

# ============================
# □ Evaluación y predicción
# ============================

# Predecir los valores de Y
y_pred = model.predict(X_test)

# ============================
# □ Crear la tabla resultante
# ============================

# Restablecer el escalado de X_test para mostrar los valores
originales (opcional)
X_test_original = pd.DataFrame(scaler.inverse_transform(X_test),
columns=['num_stops', 'milliseconds', 'Percentil_parada', 'grid'])

# Convertir predicciones a formato de una sola columna
y_pred = y_pred.flatten()

# Crear un DataFrame con las columnas de entrada (X), la predicción de
Y y el valor real de Y
results_df = X_test_original.copy()
results_df['Prediccion_Y'] = y_pred  # Predicción de la red
results_df['Valor_Real_Y'] = y_test.reset_index(drop=True)  # Valor
real de Y

# ============================
# □ Evaluar la métrica MSE
# ============================

# Calcular el error cuadrático medio
mse = mean_squared_error(y_test, y_pred)
print(f'□ Mean Squared Error (MSE): {mse}')

# ============================
# □ Mostrar la tabla completa
# ============================

print(f'□ Tabla resultante con {len(results_df)} filas:')
```

```python
print(results_df.head(10))  # Mostrar las primeras 10 filas

# Opcional: Guardar la tabla en un archivo CSV
results_df.to_csv('resultados_red_neuronal.csv', index=False)
print('✅ La tabla se ha guardado como "resultados_red_neuronal.csv"')
```

```
Epoch 1/100

<ipython-input-14-65bc6001301b>:18: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  X[['num_stops', 'milliseconds', 'Percentil_parada', 'grid']] =
scaler.fit_transform(X[['num_stops', 'milliseconds',
'Percentil_parada', 'grid']])
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py
:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to
a layer. When using Sequential models, prefer using an `Input(shape)`
object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

263/263 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - loss: 75.4172 - val_loss:
21.3721
Epoch 2/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 22.3599 - val_loss:
19.1854
Epoch 3/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 20.8970 - val_loss:
18.6172
Epoch 4/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.7381 - val_loss:
18.3069
Epoch 5/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.8484 - val_loss:
18.3774
Epoch 6/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.3805 - val_loss:
18.0635
Epoch 7/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.8104 - val_loss:
18.0837
Epoch 8/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.9570 - val_loss:
18.1353
Epoch 9/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7173 - val_loss:
```

```
18.0023
Epoch 10/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7595 - val_loss:
17.9159
Epoch 11/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7078 - val_loss:
18.0692
Epoch 12/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 19.4409 - val_loss:
17.9506
Epoch 13/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.3725 - val_loss:
18.0128
Epoch 14/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.5249 - val_loss:
17.9227
Epoch 15/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.5351 - val_loss:
18.0733
Epoch 16/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.4949 - val_loss:
17.8216
Epoch 17/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5791 - val_loss:
17.8374
Epoch 18/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.3010 - val_loss:
17.9845
Epoch 19/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2261 - val_loss:
17.8140
Epoch 20/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.8658 - val_loss:
17.7887
Epoch 21/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.2776 - val_loss:
17.7991
Epoch 22/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9273 - val_loss:
17.7361
Epoch 23/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.3420 - val_loss:
17.7829
Epoch 24/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1001 - val_loss:
17.7756
Epoch 25/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.2998 - val_loss:
18.0395
```

```
Epoch 26/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 18.0703 - val_loss:
17.7596
Epoch 27/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.3897 - val_loss:
17.7626
Epoch 28/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.0702 - val_loss:
17.7893
Epoch 29/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 17.8789 - val_loss:
17.7365
Epoch 30/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4433 - val_loss:
17.9972
Epoch 31/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4149 - val_loss:
18.3274
Epoch 32/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1726 - val_loss:
17.7374
Epoch 33/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8285 - val_loss:
17.7000
Epoch 34/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.7401 - val_loss:
17.7636
Epoch 35/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5035 - val_loss:
17.7211
Epoch 36/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5264 - val_loss:
17.7332
Epoch 37/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7687 - val_loss:
17.7199
Epoch 38/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.6324 - val_loss:
17.8715
Epoch 39/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3571 - val_loss:
18.1737
Epoch 40/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.5665 - val_loss:
17.6827
Epoch 41/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.7884 - val_loss:
17.6965
Epoch 42/100
```

```
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.2001 - val_loss:
17.7530
Epoch 43/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.3470 - val_loss:
17.6530
Epoch 44/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.2998 - val_loss:
17.7493
Epoch 45/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0000 - val_loss:
17.7475
Epoch 46/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.7035 - val_loss:
17.6290
Epoch 47/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.0135 - val_loss:
17.9443
Epoch 48/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1050 - val_loss:
17.6398
Epoch 49/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 19.2189 - val_loss:
17.7222
Epoch 50/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2362 - val_loss:
17.6167
Epoch 51/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9844 - val_loss:
17.8300
Epoch 52/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.2170 - val_loss:
17.6855
Epoch 53/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.2096 - val_loss:
17.6567
Epoch 54/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - loss: 17.8652 - val_loss:
17.8560
Epoch 55/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 17.3200 - val_loss:
18.0303
Epoch 56/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.2270 - val_loss:
17.7600
Epoch 57/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.1410 - val_loss:
17.6698
Epoch 58/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1825 - val_loss:
```

```
17.7555
Epoch 59/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1886 - val_loss:
17.6056
Epoch 60/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0802 - val_loss:
17.6143
Epoch 61/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3391 - val_loss:
17.7371
Epoch 62/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2559 - val_loss:
17.9122
Epoch 63/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4011 - val_loss:
17.7717
Epoch 64/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.5370 - val_loss:
17.6072
Epoch 65/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2538 - val_loss:
17.6214
Epoch 66/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7180 - val_loss:
17.6521
Epoch 67/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0329 - val_loss:
17.6247
Epoch 68/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3496 - val_loss:
17.7282
Epoch 69/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.1269 - val_loss:
17.8279
Epoch 70/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.2983 - val_loss:
17.7128
Epoch 71/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.0428 - val_loss:
17.5841
Epoch 72/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 17.8352 - val_loss:
17.7465
Epoch 73/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.4631 - val_loss:
17.6221
Epoch 74/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8239 - val_loss:
17.6714
```

```
Epoch 75/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9872 - val_loss:
17.6453
Epoch 76/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7642 - val_loss:
17.6291
Epoch 77/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.6668 - val_loss:
17.7685
Epoch 78/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8507 - val_loss:
17.6967
Epoch 79/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0001 - val_loss:
17.6613
Epoch 80/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7095 - val_loss:
17.7183
Epoch 81/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7536 - val_loss:
17.6306
Epoch 82/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.3288 - val_loss:
17.8892
Epoch 83/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2291 - val_loss:
17.7832
Epoch 84/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - loss: 18.0617 - val_loss:
17.6563
Epoch 85/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.2796 - val_loss:
17.6280
Epoch 86/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 18.1778 - val_loss:
17.6678
Epoch 87/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9633 - val_loss:
17.6671
Epoch 88/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9472 - val_loss:
17.6823
Epoch 89/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0761 - val_loss:
17.7336
Epoch 90/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7908 - val_loss:
17.7806
Epoch 91/100
```

```
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.9137 - val_loss:
17.5842
Epoch 92/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7299 - val_loss:
17.6808
Epoch 93/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.2851 - val_loss:
17.8145
Epoch 94/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.0962 - val_loss:
17.7959
Epoch 95/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.1809 - val_loss:
17.7222
Epoch 96/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.7110 - val_loss:
17.6826
Epoch 97/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 18.0384 - val_loss:
17.7155
Epoch 98/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8574 - val_loss:
17.6418
Epoch 99/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - loss: 17.8382 - val_loss:
17.6137
Epoch 100/100
263/263 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 18.3796 - val_loss:
17.6385
66/66 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
 Mean Squared Error (MSE): 17.638461712743545
 Tabla resultante con 2098 filas:
   num_stops  milliseconds  Percentil_parada  grid  Prediccion_Y
Valor_Real_Y
0        3.0       30112.0              0.33  12.0     11.949498
17
1        3.0       21644.0              0.13   2.0      3.526083
3
2        1.0       25126.0              0.95   9.0     11.177624
7
3        1.0       24583.0              0.61  15.0     13.219305
15
4        2.0       22455.0              0.90   8.0     10.590532
4
5        1.0       23100.0              0.60  19.0     15.396774
13
6        3.0       29306.0              0.44   8.0      9.781833
17
7        1.0       38277.0              0.76   6.0      9.295921
```

```
3
8        3.0       22062.0              0.14   3.0      4.219584
17
9        2.0       23696.0              0.38   1.0      3.894502
1
 La tabla se ha guardado como "resultados_red_neuronal.csv"
```

Análisis laps

Este ha sido un análisis adicional que se ha realizado sobre los tiempos de vuelta pero que no ha sido incluido en la memoria final

```
import pandas as pd
import numpy as np

# Calcular el número de paradas por piloto en cada carrera
pit_stops['num_stops'] = pit_stops.groupby(['raceId', 'driverId'])
['stop'].transform('count')



# Seleccionar columnas de interés de lap_times
#Explorar para filtrar y quitar vueltas atipicamente lentas!!!!
lap_times = lap_times[['raceId', 'driverId', 'lap',
'position','milliseconds']]

# Seleccionar columnas de interés de pit_stops
# Filtrar las paradas que tengan menos de 60,000 milisegundos
pit_stops = pit_stops[pit_stops['milliseconds'] <= 60000]
col_num_stops = pit_stops[['raceId', 'driverId','num_stops']]
pit_stops = pit_stops[['raceId', 'driverId', 'stop',
'milliseconds','lap']]

# Seleccionar columnas de interés de results
results = results[['raceId', 'driverId', 'grid',
'positionOrder','constructorId']]

#Seleccionar columnas de interés de races
races = races[['raceId','year','circuitId','name']]

#Seleccionar columnas de interés de circuits
circuits = circuits[['circuitId','location']]

#Seleccionar columnas de interés de constructor_results
constructor_results =
constructor_results[['raceId','constructorId','points']]


# Realizar la combinación mediante un left join con results
lap_times_w_pit_stops = pd.merge(
```

```
    lap_times,
    pit_stops,
    on=['raceId', 'driverId','lap'],  # Claves para hacer la unión
    how='left'                  # Tipo de unión
)

# Realizar la combinación mediante un left join con results
lap_times_w_pit_stops = pd.merge(
    lap_times_w_pit_stops,
    results,
    on=['raceId', 'driverId',],  # Claves para hacer la unión
    how='left'                  # Tipo de unión
)

# Eliminar duplicados basados en las columnas 'raceId' y 'driverId'
col_num_stops = col_num_stops.drop_duplicates(subset=['raceId',
'driverId'])

# Realizar la combinación mediante un left join con results
lap_times_w_pit_stops = pd.merge(
    lap_times_w_pit_stops,
    col_num_stops,
    on=['raceId', 'driverId',],  # Claves para hacer la unión
    how='left'                  # Tipo de unión
)

# Realizar la combinación mediante un left join con results
lap_times_w_pit_stops = pd.merge(
    lap_times_w_pit_stops,
    races,
    on=['raceId'],  # Claves para hacer la unión
    how='left'                  # Tipo de unión
)

# Realizar la combinación mediante un left join con results
lap_times_w_pit_stops = pd.merge(
    lap_times_w_pit_stops,
    circuits,
    on=['circuitId'],  # Claves para hacer la unión
    how='left'                  # Tipo de unión
)

# Realizar la combinación mediante un left join con results
lap_times_w_pit_stops = pd.merge(
    lap_times_w_pit_stops,
    constructor_results,
    on=['raceId','constructorId'],  # Claves para hacer la unión
    how='left'                  # Tipo de unión
)
```

```
#Renombrar la columna de puntos para mayor claridad
lap_times_w_pit_stops =
lap_times_w_pit_stops.rename(columns={'points': 'points_constructor'})

# 1. Crear las categorías simplificadas basadas en la posición final
def categorize_position(position):
    if position in [1, 2, 3]:
        return '1_Podio'
    elif 4 <= position <= 10:
        return '2_Top 10 sin podio'
    else:
        return '3_Sin puntos'

# Calcular el percentil de cada parada dentro de su carrera
lap_times_w_pit_stops['Percentil_parada'] =
lap_times_w_pit_stops.groupby('raceId')['milliseconds_y'] \
    .transform(lambda x: x.rank(pct=True) - 1/x.count())

# Redondear los percentiles a dos decimales
lap_times_w_pit_stops['Percentil_parada'] =
lap_times_w_pit_stops['Percentil_parada'].round(2)
```

```
<ipython-input-15-ab50b42fe1b7>:97: RuntimeWarning: divide by zero
encountered in scalar divide
  .transform(lambda x: x.rank(pct=True) - 1/x.count())
```

```
lap_times_def = lap_times_w_pit_stops
lap_times_def
```

```
{"type":"dataframe","variable_name":"lap_times_w_pit_stops"}
```

Red neuronal con 3 clases (salida)

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import regularizers

# Asegurarse de que los valores de 'milliseconds_y' y
'Percentil_parada' se llenen adecuadamente con 0
lap_times_def['milliseconds_y'] =
```

```
lap_times_def['milliseconds_y'].fillna(0)  # Usamos 0 para imputar
valores faltantes
lap_times_def['Percentil_parada'] =
lap_times_def['Percentil_parada'].fillna(0)  # Similar a
milliseconds_y

# Separar variables numéricas y categóricas
numerical_columns = ['milliseconds_x','milliseconds_y',
'Percentil_parada']  # Solo variables numéricas
categorical_columns = ['raceId', 'driverId', 'constructorId',
'num_stops', 'grid', 'stop']  # Variables categóricas

# Definir las características (X) y la variable objetivo (y)
X = lap_times_def[numerical_columns + categorical_columns]
y = lap_times_def['positionOrder']

# Codificar la variable objetivo en categorías
y = pd.cut(y, bins=[0, 3, 10, 25], labels=['1_Podio', '2_Top 10 sin
podio', '3_Sin puntos'])

# Convertir las etiquetas a valores enteros
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Preprocesamiento de las características: escalar numéricas y one-hot
codificar categóricas
preprocessor = ColumnTransformer(
    transformers=[
        # Para las variables numéricas, imputar valores faltantes y
escalar
        ('num', Pipeline([
            ('imputer', SimpleImputer(strategy='median')),  #
Imputación de valores faltantes
            ('scaler', StandardScaler())  # Escalado de las variables
numéricas
        ]), numerical_columns),

        # Para las variables categóricas, imputar los valores
faltantes y aplicar one-hot encoding
        ('cat', Pipeline([
            ('imputer', SimpleImputer(strategy='most_frequent')),  #
Imputación de valores faltantes
            ('onehot', OneHotEncoder(handle_unknown='ignore'))  #
Codificación OneHot
        ]), categorical_columns)
    ])

# Dividir el dataset en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
test_size=0.2, random_state=42)
```

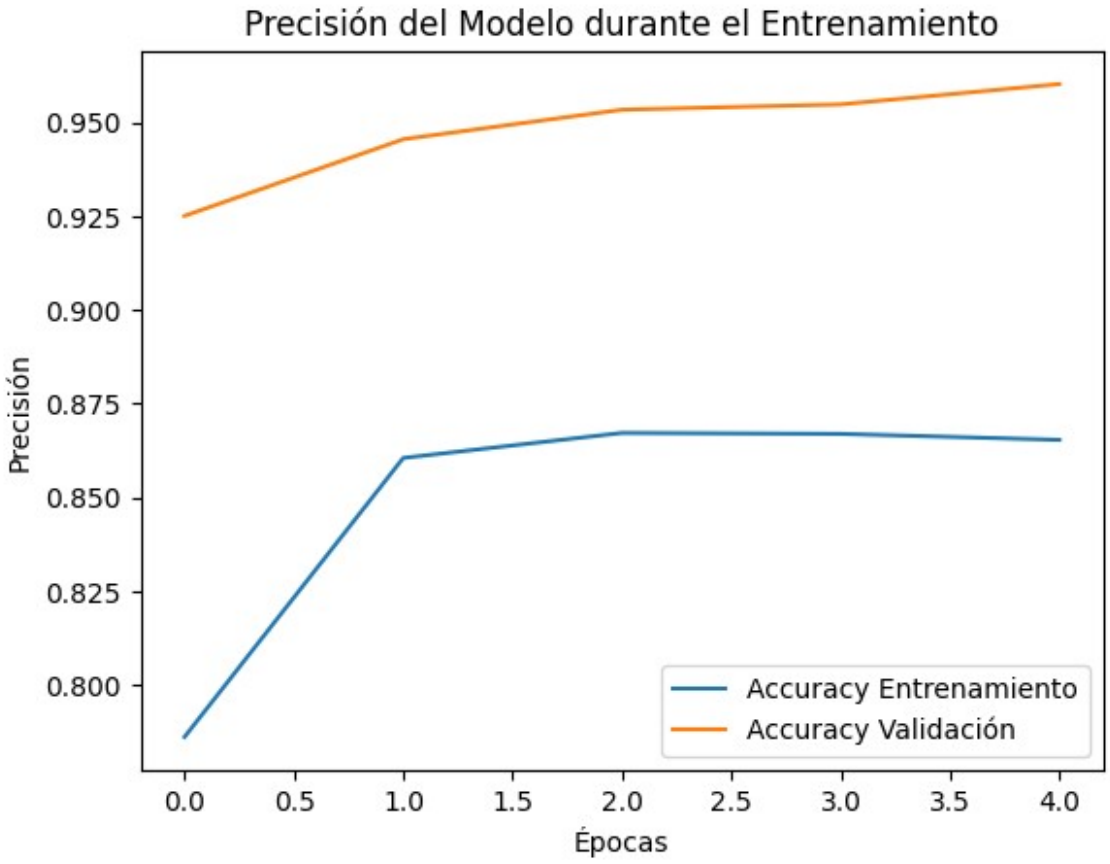
```
# Aplicar el preprocesamiento
X_train = preprocessor.fit_transform(X_train)
X_test = preprocessor.transform(X_test)

# Convertir la variable objetivo a formato categórico (one-hot
encoded)
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Crear el modelo de red neuronal
model = Sequential()

# Capa de entrada con regularización L2
model.add(Dense(64, activation='relu',
input_shape=(X_train.shape[1],),
kernel_regularizer=regularizers.l2(0.001)))

# Dropout para evitar sobreajuste
model.add(Dropout(0.5))

# Capa oculta
model.add(Dense(32, activation='relu',
kernel_regularizer=regularizers.l2(0.001)))

# Capa de salida
model.add(Dense(3, activation='softmax'))

# Compilar el modelo
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Definir el callback de EarlyStopping para evitar sobreajuste
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

# Entrenar el modelo con EarlyStopping
history = model.fit(X_train, y_train, epochs=5, batch_size=32,
validation_data=(X_test, y_test), callbacks=[early_stopping])

# Evaluar el modelo en los datos de prueba
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Accuracy en test: {accuracy*100:.2f}%")

# Obtener predicciones y calcular métricas adicionales
from sklearn.metrics import classification_report
y_pred = np.argmax(model.predict(X_test), axis=1)  # Predicciones del
modelo
y_true = np.argmax(y_test, axis=1)  # Valores reales
```

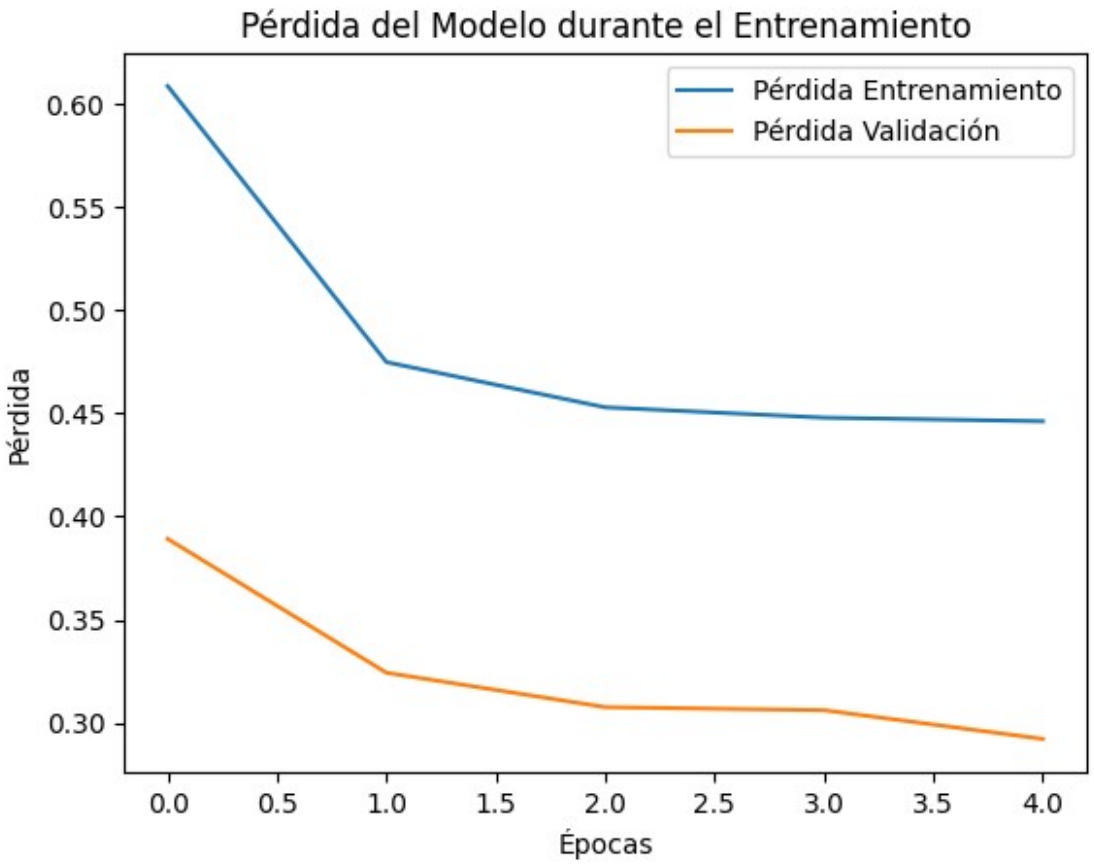
```
# Mostrar el reporte de clasificación
print(classification_report(y_true, y_pred, target_names=['1_Podio',
'2_Top 10 sin podio', '3_Sin puntos']))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 74s 5ms/step - accuracy: 0.7299 -
loss: 0.6923 - val_accuracy: 0.9250 - val_loss: 0.3892
Epoch 2/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 75s 5ms/step - accuracy: 0.8563 -
loss: 0.4875 - val_accuracy: 0.9454 - val_loss: 0.3243
Epoch 3/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 83s 6ms/step - accuracy: 0.8663 -
loss: 0.4560 - val_accuracy: 0.9533 - val_loss: 0.3077
Epoch 4/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 75s 5ms/step - accuracy: 0.8666 -
loss: 0.4498 - val_accuracy: 0.9547 - val_loss: 0.3062
Epoch 5/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 72s 5ms/step - accuracy: 0.8656 -
loss: 0.4467 - val_accuracy: 0.9601 - val_loss: 0.2923
3594/3594 ━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.9595 - loss:
0.2940
Accuracy en test: 96.01%
3594/3594 ━━━━━━━━━━━━━━━━━━━━ 11s 3ms/step
                    precision    recall  f1-score   support

           1_Podio       0.96      0.96      0.96     19515
2_Top 10 sin podio       0.95      0.96      0.96     44962
      3_Sin puntos       0.97      0.96      0.96     50529

          accuracy                           0.96    115006
         macro avg       0.96      0.96      0.96    115006
      weighted avg       0.96      0.96      0.96    115006
```

```
import matplotlib.pyplot as plt

# Graficar la precisión de entrenamiento y validación
plt.plot(history.history['accuracy'], label='Accuracy Entrenamiento')
plt.plot(history.history['val_accuracy'], label='Accuracy Validación')
plt.title('Precisión del Modelo durante el Entrenamiento')
plt.xlabel('Épocas')
plt.ylabel('Precisión')
plt.legend()
```

```
plt.show()

# Graficar la pérdida de entrenamiento y validación
plt.plot(history.history['loss'], label='Pérdida Entrenamiento')
plt.plot(history.history['val_loss'], label='Pérdida Validación')
plt.title('Pérdida del Modelo durante el Entrenamiento')
plt.xlabel('Épocas')
plt.ylabel('Pérdida')
plt.legend()
plt.show()
```





Red neuronal con 5 clases de salida

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import regularizers

# Asegurarse de que los valores de 'milliseconds_y' y
'Percentil_parada' se llenen adecuadamente con 0
lap_times_def['milliseconds_y'] =
lap_times_def['milliseconds_y'].fillna(0)  # Usamos 0 para imputar
valores faltantes
lap_times_def['Percentil_parada'] =
lap_times_def['Percentil_parada'].fillna(0)  # Similar a
```

```
milliseconds_y

# Separar variables numéricas y categóricas
numerical_columns = ['milliseconds_x','milliseconds_y',
'Percentil_parada']  # Solo variables numéricas
categorical_columns = ['raceId', 'driverId', 'constructorId',
'num_stops', 'grid', 'stop']  # Variables categóricas

# Definir las características (X) y la variable objetivo (y)
X = lap_times_def[numerical_columns + categorical_columns]
y = lap_times_def['positionOrder']

# Codificar la variable objetivo en 5 categorías
y = pd.cut(y, bins=[0, 1, 3, 5, 10, np.inf], labels=['1_Campeon',
'2_Podio sin campeonato', '3_Top 5 sin podio', '4_Puntos sin Top 5',
'5_Sin puntos'])

# Convertir las etiquetas a valores enteros
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Preprocesamiento de las características: escalar numéricas y one-hot
codificar categóricas
preprocessor = ColumnTransformer(
    transformers=[
        # Para las variables numéricas, imputar valores faltantes y
escalar
        ('num', Pipeline([
            ('imputer', SimpleImputer(strategy='median')),  #
Imputación de valores faltantes
            ('scaler', StandardScaler())  # Escalado de las variables
numéricas
        ]), numerical_columns),

        # Para las variables categóricas, imputar los valores
faltantes y aplicar one-hot encoding
        ('cat', Pipeline([
            ('imputer', SimpleImputer(strategy='most_frequent')),  #
Imputación de valores faltantes
            ('onehot', OneHotEncoder(handle_unknown='ignore'))  #
Codificación OneHot
        ]), categorical_columns)
    ])

# Dividir el dataset en entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded,
test_size=0.2, random_state=42)

# Aplicar el preprocesamiento
X_train = preprocessor.fit_transform(X_train)
```

```
X_test = preprocessor.transform(X_test)

# Convertir la variable objetivo a formato categórico (one-hot
encoded)
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Crear el modelo de red neuronal
model = Sequential()

# Capa de entrada con regularización L2
model.add(Dense(64, activation='relu',
input_shape=(X_train.shape[1],),
kernel_regularizer=regularizers.l2(0.001)))

# Dropout para evitar sobreajuste
model.add(Dropout(0.5))

# Capa oculta
model.add(Dense(32, activation='relu',
kernel_regularizer=regularizers.l2(0.001)))

# Capa de salida con 5 unidades para las 5 categorías
model.add(Dense(5, activation='softmax'))

# Compilar el modelo
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Definir el callback de EarlyStopping para evitar sobreajuste
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True)

# Entrenar el modelo con EarlyStopping
history = model.fit(X_train, y_train, epochs=5, batch_size=32,
validation_data=(X_test, y_test), callbacks=[early_stopping])

# Evaluar el modelo en los datos de prueba
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Accuracy en test: {accuracy*100:.2f}%")

# Obtener predicciones y calcular métricas adicionales
from sklearn.metrics import classification_report
y_pred = np.argmax(model.predict(X_test), axis=1)  # Predicciones del
modelo
y_true = np.argmax(y_test, axis=1)  # Valores reales

# Mostrar el reporte de clasificación
print(classification_report(y_true, y_pred, target_names=['1_Campeon',
'2_Podio sin campeonato', '3_Top 5 sin podio', '4_Puntos sin Top 5',
```

```
'5_Sin puntos']))

# Convertir las predicciones y los valores reales a sus etiquetas
originales
y_pred_labels = label_encoder.inverse_transform(y_pred)  #
Predicciones del modelo
y_true_labels = label_encoder.inverse_transform(y_true)  # Valores
reales del dataset

# Crear un DataFrame con las predicciones y los valores reales
predictions_df = pd.DataFrame({
    'Predicción': y_pred_labels,  # Columna con las predicciones
    'Valor Real': y_true_labels   # Columna con los valores reales
})
```

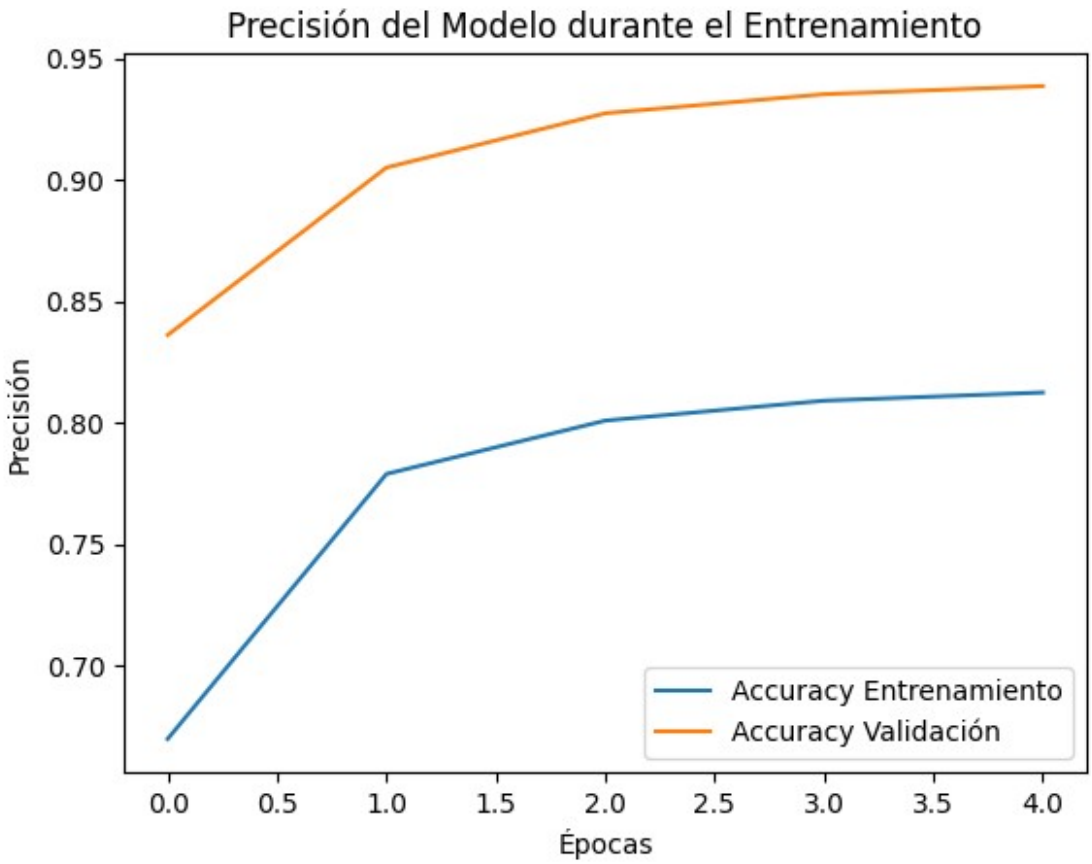
```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/
dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim`
argument to a layer. When using Sequential models, prefer using an
`Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

Epoch 1/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 78s 5ms/step - accuracy: 0.6140 -
loss: 0.9909 - val_accuracy: 0.8362 - val_loss: 0.6652
Epoch 2/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 73s 5ms/step - accuracy: 0.7691 -
loss: 0.7517 - val_accuracy: 0.9051 - val_loss: 0.5362
Epoch 3/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 81s 5ms/step - accuracy: 0.7984 -
loss: 0.6852 - val_accuracy: 0.9275 - val_loss: 0.4727
Epoch 4/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 84s 5ms/step - accuracy: 0.8074 -
loss: 0.6595 - val_accuracy: 0.9353 - val_loss: 0.4466
Epoch 5/5
14376/14376 ━━━━━━━━━━━━━━━━━━━━ 77s 5ms/step - accuracy: 0.8128 -
loss: 0.6448 - val_accuracy: 0.9387 - val_loss: 0.4291
3594/3594 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.9378 - loss:
0.4316
Accuracy en test: 93.87%
3594/3594 ━━━━━━━━━━━━━━━━━━━━ 12s 3ms/step
                        precision    recall  f1-score   support

             1_Campeon       0.85      0.92      0.88      6523
2_Podio sin campeonato       0.91      0.93      0.92     12992
     3_Top 5 sin podio       0.91      0.94      0.92     13105
    4_Puntos sin Top 5       0.93      0.95      0.94     31857
          5_Sin puntos       0.97      0.94      0.96     50529

              accuracy                           0.94    115006
```

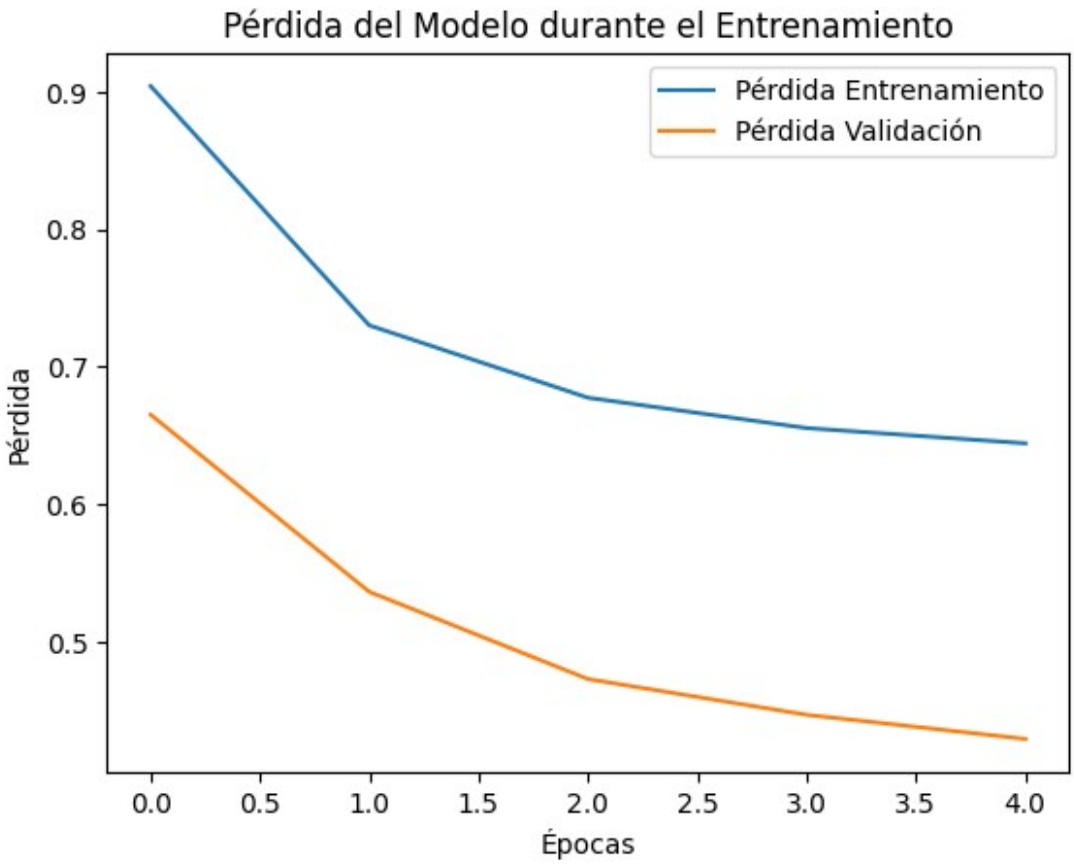
```
            macro avg       0.91      0.93      0.92    115006
         weighted avg       0.94      0.94      0.94    115006
```

```
# Mostrar las primeras filas del DataFrame con las predicciones y
valores reales
predictions_df
```

```
{"type":"dataframe","variable_name":"predictions_df"}
```

```
import matplotlib.pyplot as plt

# Graficar la precisión de entrenamiento y validación
plt.plot(history.history['accuracy'], label='Accuracy Entrenamiento')
plt.plot(history.history['val_accuracy'], label='Accuracy Validación')
plt.title('Precisión del Modelo durante el Entrenamiento')
plt.xlabel('Épocas')
plt.ylabel('Precisión')
plt.legend()
plt.show()

# Graficar la pérdida de entrenamiento y validación
plt.plot(history.history['loss'], label='Pérdida Entrenamiento')
plt.plot(history.history['val_loss'], label='Pérdida Validación')
plt.title('Pérdida del Modelo durante el Entrenamiento')
plt.xlabel('Épocas')
plt.ylabel('Pérdida')
plt.legend()
plt.show()
```


Precisión del Modelo durante el Entrenamiento
0.95
0.90
0.85
0.80
0.75
0.70
Precisión
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Épocas
Accuracy Entrenamiento
Accuracy Validación


Pérdida del Modelo durante el Entrenamiento
Pérdida Entrenamiento
Pérdida Validación
Pérdida
Épocas
0.9
0.8
0.7
0.6
0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

ANEXO 3

Scrapping Comentarios

En este fichero se utiliza la API oficial de reddit para extraer comentarios de las discusiones postcarrera del subreddit oficial de fórmula 1

Se comienza definiendo las funciones para extraer los comentarios e importando la API de reddit

```
import praw

def create_reddit_instance():
    # Crear una instancia de Reddit
    return praw.Reddit(
    client_id="DTYS-7zYELk-n8IFQs1DOQ",
    client_secret="9Hktex2aT_2gWyrUsMcGbSmhCfLbnQ",
    user_agent="RedditCommentScraper/1.0 by u/Emergency_Athlete906"
)

def extract_comments(post_url, num_comments=10):
    # Crear instancia de Reddit
    reddit = create_reddit_instance()

    # Cargar el post
    submission = reddit.submission(url=post_url)

    # Limitar el número de comentarios cargados
    submission.comment_limit = num_comments

    # Actualizar comentarios
    submission.comments.replace_more(limit=0)  # No cargar ramas de
comentarios adicionales

    # Extraer comentarios principales
    comments = [comment.body for comment in
submission.comments.list()]
    return comments
```

A partir de aquí se recrea el mismo formato para cada una de las carreras de la temporada 2023. El fichero con la extracción de 2022 sigue la misma estructura

Se define el enlace a la publicación de reddit y se extraen los comentarios

```
#1-BAREIN

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/11j1huz/2023_bahrain_grand
_prix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
```

```
num_comments = 253

# Extraer comentarios
comments_barein = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_barein)} comentarios:")
for i, comment in enumerate(comments_barein[:10], 1):  # Mostrar los
primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. A shame that Lando’s six-stop strategy didn’t work out

2. what an interesting day for McLaren lmao

McLaren Race Summary:

- McLaren.exe literally stopped working for Piastri
- Lando got the full Ricciardo McLaren experience just vibing around
at the back of the pack

3. Guys, I think the FIA might have executed Esteban Ocon. Prayers and
good vibes, etc.

4. Max just casually destroying the floor on the ramp

5. Ocon’s taxi to the airport gets a penalty

6. The hardest part of Max’s race was parking on that podium.

7. Holy fuck that ramp almost fucked Max car lmao

8. Happy for williams. With years of pain as a backmarker, it looks
like they might have a midfield car now.

9. For anyone not counting Lando pitted 6 times this race.

10. The only serious opponent for RedBull was the ramp onto the
display
```

```
#2-ARABIA SAUDI

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/11vt7ui/2023_saudi_arabian
_grand_prix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
```

```
num_comments = 255

# Extraer comentarios
comments_arabia_saudi = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_arabia_saudi)} comentarios:")
for i, comment in enumerate(comments_arabia_saudi[:10], 1):  # Mostrar
los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. Does McLaren have a Talent Killing program?

2. 2% of Alonso's podiums have been with Aston Martin, while 67% of
Aston Martin's official podiums have been with Alonso...

EDIT: well, I guess it's about 1% again personally, and 50% for the
team ☹

EDIT 2: OK, looks like we're back to 2% for FA, and 67% of AM. Crazy
stuff happening, I could use a drink...

3. Nice of checo to celebrate with Aston as well, his old team! He did
the same in Baku as well IIRC

4. We’re about to get another “Letter from Zak Brown / Letter to the
fans” from McLaren

5. Alonso with 100 podiums … amazing performance and career □□□

Edit: Back to 99 now … curse you penalties

Edit 2: Alonso is reinstated. Back to P3 😢

**what a shitshow by the FIA

6. Incredible that it took the FIA the **whole fucking race** to find
out that AM didn't serve the penalty correctly.

7. Wish they’d stop filming Jos, such a waste of space with that
attitude :(

8. This is Alonso's first back-to-back podium since Monza-Singapore in
**2013**

(Hopefully no other penalties)

9. As much as it pains me to say it, today's race was proof of two
things for this season. 1) Red Bull's car is on a different level than
```

```
the rest of the grid and; 2) Max Verstappen is likely going to run
away with the Driver's Cup.

Dude went from P-15 to P-2 before the half way point of the race and
then stayed there. Abandon all hope boys.

10. Lmao. Did Max just ran back to the pits instead of the track?!
```

```
#3-AUSTRALIA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/129dvvg/2023_australian_gr
and_prix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 250

# Extraer comentarios
comments_australia = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_australia)} comentarios:")
for i, comment in enumerate(comments_australia[:10], 1):  # Mostrar
los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. Look what all you bandwagoners achieved. An unbelievable P3 for
Alonso even after being taken out on the restart. Could you manifest
me some lottery numbers too?

2. Well, at lap 46 I though, "It's late, I'm not gonna miss anything
if I sleep now."  Then I thought, "It's only 12 more laps."

3. Lance Stroll at the restart managed to go from 6th to 3rd to 12th
to 5th to 4th. Incredible, what a drive hahaha

4. Imagine if we used the Hulk P4 order, Sainz got his penalty, and
then Hulk is DSQd from P3 for fuel

5. Hulk is tied with Leclerc in the WDC table. Do with that
information what you will

6. All ima say is from Baku 2017-Italy 2020 there were no red flags
and in this race we had 3 alone.

7. Sargeant had one car to avoid and still smacked DeVries flush lmao
```

8. There's another red flag because Nico broke down on the cool down lap lmao

9. Jenson Button was an absolute joy to listen to. Hope we have him at more races going forward!

10. "Rolex Ambassador Jackie Stewart". Sure, that's what Jackie Stewart is best known for.

```
#4-Azerbaiyan

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/133o178/2023_azerbaijan_grand_prix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 252

# Extraer comentarios
comments_azerbaiyan = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_azerbaiyan )} comentarios:")
for i, comment in enumerate(comments_azerbaiyan[:10], 1):  # Mostrar
los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

Se han extraído 250 comentarios:
1. Commentators were genuinely angry

2. I wonder if they would have kept Crowdstrike as a sponsor if the photographers got hit?

3. We waited a month for that 😭

4. Checo: “You got unlucky there. “ (Safety car replay being shown)

Max: “Yeah, it happens, you had it last year in Jeddah. .”

5. not even the bot was able to wait much longer and just wanted the race to end

6. Nyck De Vries is turning out to be quite disappointing.

7. Never heard Ted so fucking furious like this before about Ocon's pitstop, rightly so.

8. Omg Checo getting eaten up by a gaggle of rabid fans.

```
You could hear Naomi asking "where's security?" as the camera cut
away.

Gross.

9. Ocon’s pit stop the most exciting part of the race lol

10. Charles went from suicidal to slightly hopeful so that’s nice


#5-MIAMI

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/13b2sc8/2023_miami_grand_p
rix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 252

# Extraer comentarios
comments_miami = extract_comments(post_url, num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_miami )} comentarios:")
for i, comment in enumerate(comments_miami[:10], 1):  # Mostrar los
primeros 10 comentarios
    print(f"{i}. {comment}\n")

Se han extraído 250 comentarios:
1. All 20 cars finish and not a single yellow flag.

2. All 50 Mclaren sponsors got like 5 seconds combined of airtime this
race.

3. KMAG bringing the point home for HAAS

4. 12th ever no dnf race

5. [deleted]

6. How did Alonso know when his teammate made a pass? That's easy.
He's a Sky Q or Sky Glass customer and pressed the red button to see
highlights alongside the live action.

7. I love that Fernando celebrates Every podium as a win

8. That Magnussen-Leclerc fight was entertaining as hell tho

9. Y’all better look at some of the overtakes Yuki made! They were
```

```
godly

10. [deleted]


#6-MONACO

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/13u2g56/2023_monaco_grand_
prix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 252

# Extraer comentarios
comments_monaco = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_monaco)} comentarios:")
for i, comment in enumerate(comments_monaco[:10], 1):  # Mostrar los
primeros 10 comentarios
    print(f"{i}. {comment}\n")

Se han extraído 250 comentarios:
1. The helicopter shots really made this more interesting.  The
elevation changes were more evident.

2. Ok what the fuck were the cars made of today? So much ping ponging
across the walls and no suspension damages or punctures.

3. Ocon on the podium with his biggest F1-friends Max and Nando.

4. I bet the band only practiced the Dutch and Austrian anthems

5. For a brief moment there we had complete and utter chaos and it was
glorious

6. My favourite part was when Max's tires pulled a Jesus Christ and
resurrected themselves.

7. That pitstop from AM to put on medium when it was raining was
painful to watch....felt bad for Alonso

8. Tom Holland waving that flag like a pro.

9. Love the human chain of love trying to stop the people from getting
to the cars lmao

10. Absolute shocker of a weekend for Perez.  of course the crash in
```

```
Q1 that made him start at the back set it all up.  But also today he
had so many colisions from desperate dives, 5 pit stops and lapped
twice.  He was struggling all race among the last 5 positions.


#7-ESPAÑA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/140esk5/2023_spanish_grand
_prix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 256

# Extraer comentarios
comments_españa = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_españa)} comentarios:")
for i, comment in enumerate(comments_españa[:10], 1):  # Mostrar los
primeros 10 comentarios
    print(f"{i}. {comment}\n")

Se han extraído 250 comentarios:
1. Max welcomes Lewis and George to his post race cooldown room
podcast

2. once charles pitted for the last time, he went radio silent and
even ignored xavi asking for tyre updates lmfao

3. Expection : Rain on a raceday.

Reality : Sweat on turn 5

4. VER will retire due to boredom at this rate

5. Max shitting out a fastest lap for the lolz.

Mercedes updates seem to work a bit.

Ferrari sucks.

6. Sainz: Find out how to beat Checo

Ferrari: You on slow hards, Checo on fresh softs. Good luck

7. Lewis Hamilton: I'd like to take my hat off to the team
Also Lewis Hamilton: Clearly wearing a hat
```

8. You could feel on the radio that Max was pissed at his pitwall when they kept bothering him about track limits as it was pausing his podcast.

9. Lewis and Nico acting like they don’t know each other I kind of respect it

10. Highlight of the race had to be Russell sweating so much he thought it was raining.

```
#8-CANADA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/14ctc0l/2023_canadian_grand_prix_post_race_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 253

# Extraer comentarios
comments_canada = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_canada)} comentarios:")
for i, comment in enumerate(comments_canada[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

Se han extraído 250 comentarios:
1. “Your rear end is insane”

“Mine?”

“No, his”

□

2. Can we talk about how Perez LOST time to the Ferraris while on faster fresher tyres? Like what?

3. Bruh why cut away from cooldown room

4. Albon 58 laps on the hards O_o

5. This podium has won 17% of all F1 races ever

6. Devries is sweating buckets rn

7. What a podium ladies and gentlemen, take it in, that’s F1 royalty for the past 20 years.

8. Max briefing Lewis on what happened to George in the media room lol

9. “I almost knocked myself out in that kerb, haha!”

That haha carried a lot of weight

10. As a Ferrari fan, this was actually a pretty good race. No drama, strategy that actually worked (even though Sainz had to overrule the team once again) and tire deg was actually pretty damn good. This is what the team needs. Sad with no podium, but it wasn't there today.

As a Haas fan, it was absolutely horrible. The car is just shit in a race. It really is 2019 all over again. Didn't help that DeVries fucked KMag who was trying to do a one stop, but there probably wasn't any points today no matter what.

```
#9-AUSTRIA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/14oprcg/2023_austrian_grand_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 251

# Extraer comentarios
comments_austria = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_austria)} comentarios:")
for i, comment in enumerate(comments_austria[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

Se han extraído 250 comentarios:
1. Unimpressive from Aston. Decentish pace. Terrible strategy

2. for participating in this post race discussion, believe it or not, you too are receiving a 5 second penalty for track limits

3. Max was so bored that he decided to have a Q4 session at the end of the race.

4. >Lewis, the car is bad, we know, please drive it.

```
This legendary radio message needs to be in this thread for posterity.

5. What a defence from Sainz!

6. Medium hard medium SOFT

7. Charles is just happy to talk strategy with somebody halfway competent.

8. Hulkenberg getting a Black and White flag for exceeding track limits *after* he had already dropped out of the race is one of the funniest things I've seen in this sport

Honestly, how the fuck is the system for reviewing track limits still so slow like this? Being 10 laps late is a fucking joke

9. McLaren with a very hopeful surprise this weekend. Hopefully it isn't a one-off for them.

10. Verstappen only winning by 5 seconds, at least it was closer this time! /s


#10-UK

# URL del post que deseas analizar
post_url = "https://www.reddit.com/r/formula1/comments/14v1mc4/2023_british_grand_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 255

# Extraer comentarios
comments_uk = extract_comments(post_url, num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_uk)} comentarios:")
for i, comment in enumerate(comments_uk[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")

Se han extraído 250 comentarios:
1. From p4 p5 to p9 p10. Ferrari masterclass.

2. Lando complimenting Oscar in the cool down room, you love to see it.

3. That Lando huge smile while Max and Lewis compliment his race

4. Piastri got screwed by the safety car and only lost one position.
```

```
Could've been much worse

5. Lando trying to fight a bee whilst giving an interview is just a typical Norris scenario.

6. Lando punching a bee on live TV is peak British summer.

7. Max and Lando both hyping Oscar. You love to hear it

8. “I understand that, without my agreement, Alpine F1 have put out a press release late this afternoon that I am driving for them today. This is wrong and I have not signed a contract with Alpine at Silverstone. I will not be driving for Alpine today.”

— The Renault A523

9. Albon clears both Ferraris 😎😎😎

10. Chrome got sponsorship value for money this weekend


#11-HUNGRIA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/157gktr/2023_hungarian_grand_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 254

# Extraer comentarios
comments_hungria = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_hungria)} comentarios:")
for i, comment in enumerate(comments_hungria[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")

Se han extraído 250 comentarios:
1. My neighbors probably confused as hell and wondering why I wake up at 3am to listen to the Dutch national anthem each Sunday

2. What a weird race from Mercedes. From being slow to having a rocket ship on the last 10 laps

3. Russell from 18th, starting on hards, still beats both Ferrari and Aston’s.
```

4. I can’t wait to see Oscar as he gains more experience. The kid is incredibly fast and comfortable and the fact he is a rookie is mind blowing.

5. Ricciardo did 40 laps on mediums after getting hit and dropping to dead last on turn 1 only to regain his original position. I’m actually quite impressed.

6. My 3 personal takeaways:

Verstappen with -33s to Lando is insane and there surely was more pace in that car judging by his fastest lap.

What happened to Piastri though, I somewhat doubt he’s just a second per lap slower than lando all of a sudden in the middle of the race. Looked weird

Props to Dani ric, good recovery from the shunt early on and promising drive from him, 15s clear of tsunoda who he outquali‘d but still impressive for his first drive in AT.

7. Lando back to back podiums, McLaren hype train full send

8. Commentators know about the podcast lmfao

9. The Verstappen Podcast is getting its co-host back today

10. Ric going 40 laps on mediums is insane

```
#12-BELGICA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/15dmykn/2023_belgian_grand
_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 250

# Extraer comentarios
comments_belgica = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_belgica)} comentarios:")
for i, comment in enumerate(comments_belgica[:10], 1):  # Mostrar los
```

```
primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. At this point, I feel like I only watch F1 for Depression Leclerc. His stoic silence alone in the break room was just too funny

2. Driver of the day - Gianpiero Lambiase.

3. No drama, no bad strategy calls. A rarity for Ferrari & Leclerc.

4. "Congratulations Charles, do you have anything to say about your race there"

"Not really, not much happened, back to you Martin"

5. I have no idea how Norris suddenly jumped from P17 to P7 and that Mclaren is one of man’s greatest mysteries

6. Perez: Lewis couldn’t get you right?


Leclerc: Uuuuhmm.. Nope.

7. Babe wake up, it's time to listen to the Dutch National Anthem.

8. Max didn’t have a 34 point weekend?  Is he washed?

9. Piastri such a good add to F1

10. Really enjoyed some of those non-DRS passes today. Albon was fun to watch today…wish he had a better result.
```

```
#13-HOLANDA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/162tdrr/2023_dutch_grand_prix_postrace_discussion/?sort=confidence"
# Número de comentarios que deseas cargar
num_comments = 255

# Extraer comentarios
comments_holanda = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_holanda)} comentarios:")
for i, comment in enumerate(comments_holanda[:10], 1):  # Mostrar los
```

```
primeros 10 comentarios
    print(f"{i}. {comment}\n")

Se han extraído 250 comentarios:
1. "Maybe I cannot exit the circuit"


Lmao, also the crowd going "OOOOH FERNANDO ALONSO". Good stuff :)

2. Max: cool cool

Alonso: wasn't my best but I'll take it

Pierre: Let's fucking gooooo!

Lol

3. They knew the Dutch anthem would be played so hired a live
performance lol

4. Usually it's a "haha yes" but this time it was a "yes, haha"

5. Every driver dreams of getting onto the podium so that they can
watch ads for Paramount+ in the cooldown room.

6. To be honest, this was one of the better races this year.

7. The real hero today is soggy lion man

8. Trophy is safe from Norris today

9. Everyone discussing the race


Pierre: "Is that Call of Duty?"

10. absolute blast this race was


#14-ITALIA

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/168yodr/2023_italian_grand
_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 250

# Extraer comentarios
comments_italia = extract_comments(post_url,
num_comments=num_comments)
```

```python
# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_italia)} comentarios:")
for i, comment in enumerate(comments_italia[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. Xavi: no risk please. Charles: I’m gonna pretend I didn’t hear that.

2. Charles behind other drivers : I sleep

Charles behind his own teammate : REAL SHIT

3. [deleted]

4. Ferrari single handedly made this race interesting, Sainz had to fight both the Red Bulls and Leclerc, imo he deserves the podium.

5. I can never get over Sainz’s movie star hair. Despite the helmet and all that pressure.

6. You know what? Many people might call it stupid and risky...but watching the two Ferrari's fight was great...this is racing in the end and it's all about the drivers and their need to beat each other.

7. This is without a doubt one of the best race weekends from Sainz I have ever witnessed, guy was first in FP2 and FP3,qualified on pole, defended extremely well against the likes of Max, Checo and Leclerc in the race and secured a podium place.

8. Verstappen was slightly inconvenienced on his way to another routine win whilst Ferrari were battling with each other with Leclerc almost sending his Sainz to the moon lmao

9. Alex Albonso. The rolling roadblock.

10. I can’t believe we almost witnessed the end of Ferrari. If they both crashed there at home after the weekend they had.
```

```python
#15-SINGAPUR

# URL del post que deseas analizar
post_url = "https://www.reddit.com/r/formula1/comments/16l17fx/2023_singapore_grand_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 250
```

```
# Extraer comentarios
comments_singapur = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_singapur)} comentarios:")
for i, comment in enumerate(comments_singapur[:10], 1):  # Mostrar los
primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. “Norris 0.8s behind in your DRS”
“YES IT’S ON PURPOSE”

2. Don’t let Red Bulls performance distract you from the absolute
nightmare of a weekend this was for Aston Martin.

3. All the other drivers now passing Russell like "wtf did you do"

4. Without a shadow of a doubt, Sainz’ best weekend of his career. Not
a single wrong step. Pulled off a strategy masterclass.

5. Sainz masterclass

6. Sainz using Norris as bait but also towing him along for DRS

I'm gonna do what's called a pro gamer move - Sainz, probably

7. Carlos Sainz, take a fucking bow. This is one of the smartest
drives I have ever seen, one of the smartest ways to win, especially
after a strategy that put him on the back foot. Incredible.

Edit: Also saved NOR a podium place, gave him a P2 with that DRS
genius.

8. The teamwork between Sainz and Lando to hold off Mercedes was
incredible. “Yeah. That is on purpose” was such a cold response when
being told Norris has DRS on him!

9. Congratulations to Liam Lawson for getting his first points in F1!
Heck, he did it in Singapore of all places. Dude deserves a full time
seat.

10. Lawson you beauty

P9 is exceptional
```

```
#16-JAPON
```

```python
# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/16qqikt/2023_japanese_grand_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 251

# Extraer comentarios
comments_japon = extract_comments(post_url, num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_japon)} comentarios:")
for i, comment in enumerate(comments_japon[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. Lmao, such an Oscar level celebration

*silence*

... Woo.

2. Maybe the worst driver performance I’ve ever seen today from Checo.

So happy for Oscar.

3. "Interesting" at Checo's crash.. so much said with so little

4. I'm convinced that if Mercedes builds a competitive car before Lewis retires, he and George will hate each other after approximately 5 minutes lol

5. "Interesting. . . " **looks at the camera**

6. Horner eyeing Norris and Piastri on the podium like, "well I need one of yous"

7. Red Bull are the first team to get a race win and double DNF in the same race

8. "Interesting" fucking lol. It's even funnier cause they know this is being broadcasted.

9. Does it drive anyone else *NUTS* when they highlight the driver's names in a battle and it gets rid of the deltas??

10. "interesting" watching Perez breaking his front wing... Max I can't
```

```python
#17-CATAR

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/1736aot/2023_qatar_grand_prix_postrace_thread/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 251

# Extraer comentarios
comments_catar = extract_comments(post_url, num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_catar)} comentarios:")
for i, comment in enumerate(comments_catar[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. "see you tomorrow"

"god no"

LMAO

2. Piastri: "Was it George and Lewis that crashed? Wow! Thank you Mercedes."

LMAO

3. Considering Russell and Perez were down the back after the first few laps and George ended up 4th, that's a fucking embarrassing showing from Perez

4. Best cool-down room in years. Oscar just lying there saying “Thank you Mercedes” was fucking magical

5. Crazy how much of a complete write-off Checo Perez has become. What a shame.

6. Oscar is brilliant

“Was it Lewis and George who crashed? Haaaaa”

7. Oscar rolling around on the floor. Max in the wrong chair.

This cooldown room is really something

8. Wow. Ocon vomited, stroll in a wheelchair, Sargeant retired due to feeling unwell, this is def worse than Singapore
```

```
9. Apparently only three drivers did not exceed track limits today.

Max

Lewis

Sainz

lmao

10. I AM LOVING THIS COOLDOWN ROOM


#18-TEXAS

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/17e2qb7/2023_united_states
_grand_prix_postrace_discussion/?sort=confidence"
# Número de comentarios que deseas cargar
num_comments = 254

# Extraer comentarios
comments_texas = extract_comments(post_url, num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_texas)} comentarios:")
for i, comment in enumerate(comments_texas[:10], 1):  # Mostrar los
primeros 10 comentarios
    print(f"{i}. {comment}\n")

Se han extraído 250 comentarios:
1. Charles Leclerc post-race team radio (P6):

"Couldn‘t do any better with that strategy. I think I maximised
everything with that strategy."

Xavi: You managed the tyres well.

Charles: "Doesn‘t matter when we have such a shit result."

You tell em Charles

2. Stroll finishing in P9 after starting from the pit lane is actually
pretty good. Especially considering the car's pace at this point

3. AlphaTauri just wanted to show everyone they don't discriminate
when it comes to shit strategy calls

Doesn't matter if you're Yuki, Lawson or Ricciardo - must be
```

```
AlphaTauri lore to have one driver on a shit strategy at all times lmao

4. Verstappen : 6th -> 1st

Leclerc: 1st -> 6th

Certified Ferrari moment

5. Leclerc's race choices:

* Fucked by the slow button

* Fucked by the strategy

* Fucked by Xavi

* Fucked by a red tractor

6. GP getting out of there “enjoy the celebrations and see you Friday” lmao

7. If Mercedes were ACTUALLY GOOD STRATEGISTS they would have hacked Max’s radio and annoyed him during the braking zones. Can’t believe they’re so shit /s

8. Well I’m glad Lewis didn’t actually catch Max now. Imagine losing the win hours later

9. Wow that 3.6 second merc pitstop ended up being shockingly relevant. If they'd gotten it down to a 2.6 there might have been a real battle the last 2 laps

10. Yet another Sainz podium he doesn't get the opportunity to celebrate


#19-MEXICO

# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/17je0y4/2023_mexico_city_grand_prix_postrace_discussion/?sort=confidence"
# Número de comentarios que deseas cargar
num_comments = 251

# Extraer comentarios
comments_mexico = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
```

```
print(f"Se han extraído {len(comments_mexico)} comentarios:")
for i, comment in enumerate(comments_mexico[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. Max shooting Lewis that look over that overtake.

"Oooh, nice one!"

2. I can't begin to imagine how Checo is feeling right now: under pressure with Red Bull after a disappointing season despite being 2nd, but has a chance to make a statement result at his home race where the crowd love him.

And instead, he pushes his luck into turn 1 & gets launched off the track incurring terminal damage whilst Danny Ric puts in a great drive in an Alpha Tauri to finish 7th.

Stressed doesn't begin to describe it.

3. Lewis watching the Perez crash: “Oh! He did a me in Qatar!” 😂

4. well done Danny Ric, very solid P7 finish and was running a comfortable P5 before the red flag

5. ‘He did a me in Qatar’ that’s hilarious Lewis hahahhaha

6. How hard is it to do a fucking split screen so we can see multiple things at once? ffs we missed ricciardo getting to .057s of russell

7. Booing Leclerc is like kicking a puppy.

8. +0.056 and they fucking cut away???

9.
Embarrassing final stint for George. Can't get past Sainz with a tire advantage at the start, then Norris blows by him on the same tires, and Daniel almost even gets him at the end (lol TV director). Meanwhile Lewis has no major issues clearing Leclerc at the start and putting a 9 second gap on him by the end for an easy P2 and even fastest lap on the last lap

10. Lmao booing Charles for something that was 100% not his fault. Its obvious he's not used to being boo'd
```

```
#20-BRASIL

# URL del post que deseas analizar
post_url =
```

```
"https://www.reddit.com/r/formula1/comments/17oizvd/2023_são_paulo_grand_prix_postrace_discussion/?sort=confidence"
# Número de comentarios que deseas cargar
num_comments = 254

# Extraer comentarios
comments_brasil = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_brasil)} comentarios:")
for i, comment in enumerate(comments_brasil[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. What an ending.  I want to see a photo finish of that one

2. Alonso is a beast

3. Lewis: Haha, yeah, can’t wait to get rid of this car

Toto: It’s awful, terrible, horrific. We will burn this car at the end of the season. The devil has cursed us all.

4. Love how to this day, they refuse to sit in the correct order at the Max Verstappen podcast.

5. Holy fuck Alonso you absolute legend never retire

6. Alonso isn't a man, he's just driving bulwark incarnate.

7. Holy hell that ALO vs PER battle was Intense

8. Drives like that are why people say Fernando is one of the GOATs despite only having two championships

9. Stroll finished P5. A silent race for a solid result. Way to go!

10. "two more finishes with the W14 and it's over. Does that sound good?"

"Yes" *giggles*

wow, you can tell Lewis did not like this season's perfomance from Mercedes
```

```
#21-LAS VEGAS

# URL del post que deseas analizar
```

```
post_url =
"https://www.reddit.com/r/formula1/comments/17yrso4/2023_las_vegas_grand_prix_postrace_discussion/?sort=confidence"

# Número de comentarios que deseas cargar
num_comments = 252

# Extraer comentarios
comments_vegas = extract_comments(post_url, num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_vegas)} comentarios:")
for i, comment in enumerate(comments_vegas[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. Max asking “here?” Like he’s in an Uber has me dead ahahaha

2. What a fucking ending

3. This had the potential to be the most awkward car ride but is just a normal cool down room but just small and cramped

4. Worried F1 Fans: No cooldown room this GP



F1: * Cooldown car with extended chat time and better audio *

5. LMAO Max acting like the kid with his mom driving him and his buddies home after karting all day

6. Lmao I need a cooldown car with some rivals. Imagine a Lewis/Nico Rosberg cooldown car in 2016□□□

7. These three dudes just having a chat squished together in the back of a Rolls is honestly just fucking amazing.

8. "I wanted that win so bad"


:(

9. We went from no cool down room to a cool down car 4x as long lol

10. Charles is an animal insane overtake


#22-ABU DABI
```

```python
# URL del post que deseas analizar
post_url =
"https://www.reddit.com/r/formula1/comments/184byfj/2023_abu_dhabi_grand_prix_postrace_thread/?sort_confidence"
# Número de comentarios que deseas cargar
num_comments = 253

# Extraer comentarios
comments_abudabi = extract_comments(post_url,
num_comments=num_comments)

# Mostrar los comentarios extraídos
print(f"Se han extraído {len(comments_abudabi)} comentarios:")
for i, comment in enumerate(comments_abudabi[:10], 1):  # Mostrar los primeros 10 comentarios
    print(f"{i}. {comment}\n")
```

```
Se han extraído 250 comentarios:
1. Max just casually getting a 7th win in a row but nobody bats an eye because he had 10 earlier in the same season

2. The fact that Charles seemed to be considering constructor championship strategy more from the cockpit than the entire Ferrari strategy team on the pit wall says a lot about the dire state Ferrari are still in

3. Leclerc has become peak strategist Vettel.

4. ferrari doesn't need strategists. Charles doing their job while driving lol

5. [deleted]

6. Valiant effort Charles. Team player through and through

7. Another race where the Ferrari drivers outthink their wall

8. Wow... Fred complaining that Leclerc could have backed up Russell a bit more

Had Charles not thought of it, nothing would have changed either

9. Charles absolute legend for trying to make that work!

If they had done it a lap earlier that could've worked. Damn that was nice to see.

10. [deleted]
```

Juntamos todos los comentarios y los etiquetamos con su Id de carrera

```
comentarios=[comments_barein,comments_arabia_saudi,comments_australia,
             comments_azerbaiyan,comments_miami,comments_monaco,
             comments_españa,comments_canada,comments_austria,
             comments_uk,comments_hungria,comments_belgica,
             comments_holanda,comments_italia,comments_singapur,
             comments_japon,comments_catar,comments_texas,
             comments_mexico,comments_brasil,comments_vegas,
             comments_abudabi]

# Crear la lista de carreras
lista_id = [str(i) for i in range(1098, 1121) for _ in range(250)]

lista_id = [carrera for carrera in lista_id if carrera != '1103']

import pandas as pd

comentarios = [comentario for sublist in comentarios for comentario in
sublist]
df = pd.DataFrame({
    'Comentario': comentarios,
    'raceId': lista_id
})

print(type(df))
display(df)
print(len(df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

	Comentario	raceId
0	A shame that Lando’s six-stop strategy didn’t ...	1098
1	what an interesting day for McLaren lmao\n\nMc...	1098
2	Guys, I think the FIA might have executed Este...	1098
3	Max just casually destroying the floor on the ...	1098
4	Ocon’s taxi to the airport gets a penalty	1098
...	...	...
5495	Max: Understood, 100% is not anywhere near 86%	1120
5496	[deleted]	1120
5497	Is it a gift? Yes, David. Remember when you al...	1120
5498	Lando should have known Checo was out of contr...	1120
5499	Same vibe as when I'm in CoD or Halo with a ma...	1120

```
[5500 rows x 2 columns]

5500
```

Exportamos los resultados

```
df.to_csv(r'C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\comentarios_2023.csv',index=False)
```

ANEXO 4

Análisis Sentimientos

En este fichero se utilizarán los comentarios extraidos de la temporada 2022 y 2023 para realizar el análisis de sentimientos. La extracción de los comentarios de 2020 ha sido realizado enun ficher igual al anterior, únicamente modificando la temporada.

A continuación juntamos ambos ficheros de comentarios para comenzar con el análisis

```
import pandas as pd

df=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\4to
curso\MINERIA DE DATOS\TRABAJO FINAL\DATOS EXTRAIDOS\
comentarios_2023.csv")
df2=pd.read_csv(r"C:\Users\gonza\Downloads\comentarios_2022.csv")

df=pd.concat([df,df2], axis=0, ignore_index=True)
```

Importamos todas las librerías y recursos necesarios

```
import pandas as pd
from nltk.sentiment import SentimentIntensityAnalyzer
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt


# Descargar recursos necesarios
nltk.download('vader_lexicon')
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\gonza\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\gonza\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\gonza\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\gonza\AppData\Roaming\nltk_data...
```

```
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to
[nltk_data]     C:\Users\gonza\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!

True
```

Definimos la función para preprocesar el texto y la aplicamos

```
# Paso 1: Preprocesar comentarios
def preprocesar_texto(texto):
    tokens = word_tokenize(texto.lower())
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word.isalpha() and word not
in stop_words]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return ' '.join(tokens)



df['comentario_procesado'] = df['Comentario'].apply(preprocesar_texto)
```

Analizamos el sentimiento con VADER

```
from textblob import TextBlob
# Paso 2: Análisis de sentimientos con VADER
analyzer = SentimentIntensityAnalyzer()

def analizar_sentimiento_vader(texto):
    sentimiento = analyzer.polarity_scores(texto)
    return sentimiento['compound']  # Devuelve la polaridad compuesta:
Rango de -1 (negativo) a 1 (positivo)

df['sentimiento'] =
df['comentario_procesado'].apply(analizar_sentimiento_vader)
```

Agrupamos el resultado por carrera para extraer sentimientos por carrera

```
# Paso 3: Agregar resultados por carrera
resumen_carreras = df.groupby('raceId')['sentimiento'].agg(
    promedio='mean',
    mediana='median',
    desviacion_std='std',
    comentarios_totales='count'
).reset_index()
```

Clasificamos las carreras en entretenidas o neutrales

```
# Paso 4: Clasificación de carreras
def clasificar_carrera(promedio_sentimiento):
    if promedio_sentimiento > 0.2:
        return 'Entretenida'
    else:
        return 'Neutral'

resumen_carreras['clasificacion'] =
resumen_carreras['promedio'].apply(clasificar_carrera)

# Agregar clasificación al DataFrame principal
df = df.merge(resumen_carreras[['raceId', 'clasificacion']],
on='raceId', how='left')
```

Generamos una nube de palabras para las categorías

```
# Paso 5: Generar nubes de palabras para cada clasificación
palabras_exc=['max','race','lewis','alonso','driver','perez','lando']
def generar_nube_palabras(dataframe, clasificacion, titulo):
    texto = ' '.join(dataframe[dataframe['clasificacion'] ==
clasificacion]['comentario_procesado'])
    wordcloud = WordCloud(width=800, height=400,
background_color='white',
colormap='viridis',stopwords=palabras_exc).generate(texto)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(titulo, fontsize=16)
    plt.show()

# Generar nubes de palabras
generar_nube_palabras(df, 'Entretenida', 'Nube de palabras: Carreras
Entretenidas')
```


```
generar_nube_palabras(df, 'Neutral', 'Nube de palabras: Carreras
Neutrales')
```

Exportamos los resultados a un fichero csv

```
resumen_carreras.to_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\
sentimientos_carreras.csv",index=False)
```

ANEXO 5

INFLUENCIA DE CARACTERÍSTICAS

En este fichero se extrae información de las carreras para ver qué características tienen más peso en si una carrera es entretenida o no. Se hace web scrapping de las url de las bases de datos y se extrae alguna información manualmente. Tras eso se hace un modelo de random forest para evaluar la influencia de las características.

Se importa la base de datos de interes, eliminando las columnas inservibles para el análisis

```
import pandas as pd
import numpy as np
from IPython.display import display


races=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\4to
curso\MINERIA DE DATOS\TRABAJO FINAL\archive\races.csv",na_values="\\
N")
races_2023=races[races['year']==2023]
columnas=['round','name','date','time','fp1_date','fp1_time','fp2_date
','fp2_time','fp3_date','fp3_time','quali_date','quali_time',
          'sprint_date','sprint_time']
races_2023=races_2023.drop(columns=columnas)

display(races_2023)
```

```
      raceId  year  circuitId  \
1079    1098  2023          3
1080    1099  2023         77
1081    1100  2023          1
1082    1101  2023         73
1083    1102  2023         79
1084    1104  2023          6
1085    1105  2023          4
1086    1106  2023          7
1087    1107  2023         70
1088    1108  2023          9
1089    1109  2023         11
1090    1110  2023         13
1091    1111  2023         39
1092    1112  2023         14
1093    1113  2023         15
1094    1114  2023         22
1095    1115  2023         78
1096    1116  2023         69
1097    1117  2023         32
1098    1118  2023         18
1099    1119  2023         80
1100    1120  2023         24

                                                    url
```

```
1079  https://en.wikipedia.org/wiki/2023_Bahrain_Gra...
1080  https://en.wikipedia.org/wiki/2023_Saudi_Arabi...
1081  https://en.wikipedia.org/wiki/2023_Australian_...
1082  https://en.wikipedia.org/wiki/2023_Azerbaijan_...
1083  https://en.wikipedia.org/wiki/2023_Miami_Grand...
1084  https://en.wikipedia.org/wiki/2023_Monaco_Gran...
1085  https://en.wikipedia.org/wiki/2023_Spanish_Gra...
1086  https://en.wikipedia.org/wiki/2023_Canadian_Gr...
1087  https://en.wikipedia.org/wiki/2023_Austrian_Gr...
1088  https://en.wikipedia.org/wiki/2023_British_Gra...
1089  https://en.wikipedia.org/wiki/2023_Hungarian_G...
1090  https://en.wikipedia.org/wiki/2023_Belgian_Gra...
1091  https://en.wikipedia.org/wiki/2023_Dutch_Grand...
1092  https://en.wikipedia.org/wiki/2023_Italian_Gra...
1093  https://en.wikipedia.org/wiki/2023_Singapore_G...
1094  https://en.wikipedia.org/wiki/2023_Japanese_Gr...
1095  https://en.wikipedia.org/wiki/2023_Qatar_Grand...
1096  https://en.wikipedia.org/wiki/2023_United_Stat...
1097  https://en.wikipedia.org/wiki/2023_Mexico_City...
1098  https://en.wikipedia.org/wiki/2023_S%C3%A3o_Pa...
1099  https://en.wikipedia.org/wiki/2023_Las_Vegas_G...
1100  https://en.wikipedia.org/wiki/2023_Abu_Dhabi_G...
```

Se define la función para extraer la información

```
import requests
from bs4 import BeautifulSoup

# Función para extraer datos con la estructura HTML proporcionada
def extract_info(url):
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'html.parser')

        # Extraer "Course length"
        course_length = soup.find('th', string='Course length')
        course_length = course_length.find_next('td').text.strip() if
course_length else None

        # Extraer "Distance"
        distance = soup.find('th', string='Distance')
        distance = distance.find_next('td').text.strip() if distance
else None

        # Extraer "Weather"
        weather = soup.find('th', string='Weather')
        weather = weather.find_next('td').text.strip() if weather else
None
```

```
        return course_length, distance, weather

    except Exception as e:
        print(f"Error al procesar {url}: {e}")
        return None, None, None

# Aplicar la función a todas las URLs y añadir las columnas al
DataFrame
races_2023[['course_length', 'distance', 'weather']] =
races_2023['url'].apply(lambda url: pd.Series(extract_info(url)))

display(races_2023)
```

```
      raceId  year  circuitId  \
1079    1098  2023          3
1080    1099  2023         77
1081    1100  2023          1
1082    1101  2023         73
1083    1102  2023         79
1084    1104  2023          6
1085    1105  2023          4
1086    1106  2023          7
1087    1107  2023         70
1088    1108  2023          9
1089    1109  2023         11
1090    1110  2023         13
1091    1111  2023         39
1092    1112  2023         14
1093    1113  2023         15
1094    1114  2023         22
1095    1115  2023         78
1096    1116  2023         69
1097    1117  2023         32
1098    1118  2023         18
1099    1119  2023         80
1100    1120  2023         24

                                                    url  \
1079  https://en.wikipedia.org/wiki/2023_Bahrain_Gra...
1080  https://en.wikipedia.org/wiki/2023_Saudi_Arabi...
1081  https://en.wikipedia.org/wiki/2023_Australian_...
1082  https://en.wikipedia.org/wiki/2023_Azerbaijan_...
1083  https://en.wikipedia.org/wiki/2023_Miami_Grand...
1084  https://en.wikipedia.org/wiki/2023_Monaco_Gran...
1085  https://en.wikipedia.org/wiki/2023_Spanish_Gra...
1086  https://en.wikipedia.org/wiki/2023_Canadian_Gr...
1087  https://en.wikipedia.org/wiki/2023_Austrian_Gr...
1088  https://en.wikipedia.org/wiki/2023_British_Gra...
1089  https://en.wikipedia.org/wiki/2023_Hungarian_G...
1090  https://en.wikipedia.org/wiki/2023_Belgian_Gra...
```

```
1091  https://en.wikipedia.org/wiki/2023_Dutch_Grand...
1092  https://en.wikipedia.org/wiki/2023_Italian_Gra...
1093  https://en.wikipedia.org/wiki/2023_Singapore_G...
1094  https://en.wikipedia.org/wiki/2023_Japanese_Gr...
1095  https://en.wikipedia.org/wiki/2023_Qatar_Grand...
1096  https://en.wikipedia.org/wiki/2023_United_Stat...
1097  https://en.wikipedia.org/wiki/2023_Mexico_City...
1098  https://en.wikipedia.org/wiki/2023_S%C3%A3o_Pa...
1099  https://en.wikipedia.org/wiki/2023_Las_Vegas_G...
1100  https://en.wikipedia.org/wiki/2023_Abu_Dhabi_G...
```

```
                course_length                                  distance  \
1079  5.412 km (3.363 miles)  57 laps, 308.238 km (190.253 miles)
1080  6.174 km (3.836 miles)  50 laps, 308.450 km (191.662 miles)
1081  5.278 km (3.280 miles)  58 laps, 306.124 km (190.217 miles)
1082  6.003 km (3.730 miles)  51 laps, 306.049 km (190.170 miles)
1083  5.412 km (3.363 miles)  57 laps, 308.326 km (191.584 miles)
1084  3.337 km (2.074 miles)  78 laps, 260.286 km (161.772 miles)
1085  4.657 km (2.894 miles)  66 laps, 307.236 km (190.908 miles)
1086  4.361 km (2.710 miles)  70 laps, 305.270 km (189.686 miles)
1087  4.318 km (2.683 miles)  71 laps, 306.452 km (190.420 miles)
1088  5.891 km (3.660 miles)  52 laps, 306.198 km (190.263 miles)
1089  4.381 km (2.722 miles)  70 laps, 306.630 km (190.531 miles)
1090  7.004 km (4.352 miles)  44 laps, 308.052 km (191.415 miles)
1091  4.259 km (2.646 miles)  72 laps, 306.587 km (190.504 miles)
1092  5.793 km (3.599 miles)  51 laps, 295.134 km (183.388 miles)
1093  4.940 km (3.070 miles)  62 laps, 306.143 km (190.228 miles)
1094  5.807 km (3.608 miles)  53 laps, 307.471 km (191.054 miles)
1095  5.419 km (3.367 miles)  57 laps, 308.611 km (191.762 miles)
1096  5.513 km (3.426 miles)  56 laps, 308.405 km (191.634 miles)
1097  4.304 km (2.674 miles)  71 laps, 305.354 km (189.738 miles)
1098  4.309 km (2.677 miles)  71 laps, 305.879 km (190.064 miles)
1099  6.201 km (3.853 miles)  50 laps, 309.958 km (192.599 miles)
1100  5.281 km (3.281 miles)  58 laps, 306.183 km (190.253 miles)
```

```
                                  weather
1079                                Clear
1080                                Clear
1081                                Sunny
1082                        Partly cloudy
1083                        Partly cloudy
1084    Cloudy, with some rain intervals
1085                               Cloudy
1086                        Partly cloudy
1087                               Cloudy
1088                        Partly cloudy
1089                                Sunny
1090  Partly cloudy, with a rain interval
1091                      Cloudy and rainy
```

```
1092                                  Sunny
1093                                  Clear
1094                                  Sunny
1095                                  Clear
1096                                  Sunny
1097                                  Sunny
1098                           Partly cloudy
1099                                  Clear
1100                                  Clear
```

La información extraida no está en un formato que nos venga bien así que le aplicamos funciones para llegar al formato deseado

```
import re

# Función para extraer el primer número de 'Course Length'
def extract_course_length(value):
    match = re.search(r'\d+\.\d+', value)
    return float(match.group()) * 1000 if match else None  # Convertir
a metros

# Función para extraer el número de vueltas de 'Distance'
def extract_laps(value):
    match = re.search(r'\d+ laps', value)
    return int(match.group().split()[0]) if match else None

# Aplicar las transformaciones
races_2023['Course Length (m)'] =
races_2023['course_length'].apply(extract_course_length)
races_2023['Laps'] = races_2023['distance'].apply(extract_laps)


races_2023=races_2023.drop(columns=['course_length','distance'])
display(races_2023)
```

```
      raceId  year  circuitId  \
1079    1098  2023          3
1080    1099  2023         77
1081    1100  2023          1
1082    1101  2023         73
1083    1102  2023         79
1084    1104  2023          6
1085    1105  2023          4
1086    1106  2023          7
1087    1107  2023         70
1088    1108  2023          9
1089    1109  2023         11
1090    1110  2023         13
1091    1111  2023         39
1092    1112  2023         14
```

1093	1113	2023	15
1094	1114	2023	22
1095	1115	2023	78
1096	1116	2023	69
1097	1117	2023	32
1098	1118	2023	18
1099	1119	2023	80
1100	1120	2023	24

	url
1079	https://en.wikipedia.org/wiki/2023_Bahrain_Gra...
1080	https://en.wikipedia.org/wiki/2023_Saudi_Arabi...
1081	https://en.wikipedia.org/wiki/2023_Australian_...
1082	https://en.wikipedia.org/wiki/2023_Azerbaijan_...
1083	https://en.wikipedia.org/wiki/2023_Miami_Grand...
1084	https://en.wikipedia.org/wiki/2023_Monaco_Gran...
1085	https://en.wikipedia.org/wiki/2023_Spanish_Gra...
1086	https://en.wikipedia.org/wiki/2023_Canadian_Gr...
1087	https://en.wikipedia.org/wiki/2023_Austrian_Gr...
1088	https://en.wikipedia.org/wiki/2023_British_Gra...
1089	https://en.wikipedia.org/wiki/2023_Hungarian_G...
1090	https://en.wikipedia.org/wiki/2023_Belgian_Gra...
1091	https://en.wikipedia.org/wiki/2023_Dutch_Grand...
1092	https://en.wikipedia.org/wiki/2023_Italian_Gra...
1093	https://en.wikipedia.org/wiki/2023_Singapore_G...
1094	https://en.wikipedia.org/wiki/2023_Japanese_Gr...
1095	https://en.wikipedia.org/wiki/2023_Qatar_Grand...
1096	https://en.wikipedia.org/wiki/2023_United_Stat...
1097	https://en.wikipedia.org/wiki/2023_Mexico_City...
1098	https://en.wikipedia.org/wiki/2023_S%C3%A3o_Pa...
1099	https://en.wikipedia.org/wiki/2023_Las_Vegas_G...
1100	https://en.wikipedia.org/wiki/2023_Abu_Dhabi_G...

	weather	Course Length (m)	Laps
1079	Clear	5412.0	57
1080	Clear	6174.0	50
1081	Sunny	5278.0	58
1082	Partly cloudy	6003.0	51
1083	Partly cloudy	5412.0	57
1084	Cloudy, with some rain intervals	3337.0	78
1085	Cloudy	4657.0	66
1086	Partly cloudy	4361.0	70
1087	Cloudy	4318.0	71
1088	Partly cloudy	5891.0	52
1089	Sunny	4381.0	70
1090	Partly cloudy, with a rain interval	7004.0	44
1091	Cloudy and rainy	4259.0	72
1092	Sunny	5793.0	51
1093	Clear	4940.0	62

```
1094                                Sunny             5807.0    53
1095                                Clear             5419.0    57
1096                                Sunny             5513.0    56
1097                                Sunny             4304.0    71
1098                        Partly cloudy             4309.0    71
1099                                Clear             6201.0    50
1100                                Clear             5281.0    58
```

Juntamos nuestro dataframe con el de los circuitos para obtener información de estos

```
circuits=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\archive\
circuits.csv",na_values="\\N")
circuits['url_c']=circuits['url']
columnas_c=['circuitRef','name','location','country','lat','lng','alt'
,'url']
circuits=circuits.drop(columns=columnas_c)
races_2023=pd.merge(races_2023,circuits,on='circuitId', how='left')
display(races_2023)
```

```
    raceId  year  circuitId  \
0     1098  2023          3
1     1099  2023         77
2     1100  2023          1
3     1101  2023         73
4     1102  2023         79
5     1104  2023          6
6     1105  2023          4
7     1106  2023          7
8     1107  2023         70
9     1108  2023          9
10    1109  2023         11
11    1110  2023         13
12    1111  2023         39
13    1112  2023         14
14    1113  2023         15
15    1114  2023         22
16    1115  2023         78
17    1116  2023         69
18    1117  2023         32
19    1118  2023         18
20    1119  2023         80
21    1120  2023         24

                                                  url  \
0   https://en.wikipedia.org/wiki/2023_Bahrain_Gra...
1   https://en.wikipedia.org/wiki/2023_Saudi_Arabi...
2   https://en.wikipedia.org/wiki/2023_Australian_...
3   https://en.wikipedia.org/wiki/2023_Azerbaijan_...
```

```
4   https://en.wikipedia.org/wiki/2023_Miami_Grand...
5   https://en.wikipedia.org/wiki/2023_Monaco_Gran...
6   https://en.wikipedia.org/wiki/2023_Spanish_Gra...
7   https://en.wikipedia.org/wiki/2023_Canadian_Gr...
8   https://en.wikipedia.org/wiki/2023_Austrian_Gr...
9   https://en.wikipedia.org/wiki/2023_British_Gra...
10  https://en.wikipedia.org/wiki/2023_Hungarian_G...
11  https://en.wikipedia.org/wiki/2023_Belgian_Gra...
12  https://en.wikipedia.org/wiki/2023_Dutch_Grand...
13  https://en.wikipedia.org/wiki/2023_Italian_Gra...
14  https://en.wikipedia.org/wiki/2023_Singapore_G...
15  https://en.wikipedia.org/wiki/2023_Japanese_Gr...
16  https://en.wikipedia.org/wiki/2023_Qatar_Grand...
17  https://en.wikipedia.org/wiki/2023_United_Stat...
18  https://en.wikipedia.org/wiki/2023_Mexico_City...
19  https://en.wikipedia.org/wiki/2023_S%C3%A3o_Pa...
20  https://en.wikipedia.org/wiki/2023_Las_Vegas_G...
21  https://en.wikipedia.org/wiki/2023_Abu_Dhabi_G...

                                weather  Course Length (m)  Laps  \
0                                 Clear             5412.0    57
1                                 Clear             6174.0    50
2                                 Sunny             5278.0    58
3                         Partly cloudy             6003.0    51
4                         Partly cloudy             5412.0    57
5      Cloudy, with some rain intervals             3337.0    78
6                                Cloudy             4657.0    66
7                         Partly cloudy             4361.0    70
8                                Cloudy             4318.0    71
9                         Partly cloudy             5891.0    52
10                                Sunny             4381.0    70
11  Partly cloudy, with a rain interval             7004.0    44
12                      Cloudy and rainy             4259.0    72
13                                Sunny             5793.0    51
14                                Clear             4940.0    62
15                                Sunny             5807.0    53
16                                Clear             5419.0    57
17                                Sunny             5513.0    56
18                                Sunny             4304.0    71
19                        Partly cloudy             4309.0    71
20                                Clear             6201.0    50
21                                Clear             5281.0    58

                                                  url_c
0   http://en.wikipedia.org/wiki/Bahrain_Internati...
1   http://en.wikipedia.org/wiki/Jeddah_Street_Cir...
2   http://en.wikipedia.org/wiki/Melbourne_Grand_P...
3      http://en.wikipedia.org/wiki/Baku_City_Circuit
4   http://en.wikipedia.org/wiki/Miami_Internation...
```

```
5      http://en.wikipedia.org/wiki/Circuit_de_Monaco
6   http://en.wikipedia.org/wiki/Circuit_de_Barcel...
7   http://en.wikipedia.org/wiki/Circuit_Gilles_Vi...
8          http://en.wikipedia.org/wiki/Red_Bull_Ring
9    http://en.wikipedia.org/wiki/Silverstone_Circuit
10          http://en.wikipedia.org/wiki/Hungaroring
11  http://en.wikipedia.org/wiki/Circuit_de_Spa-Fr...
12     http://en.wikipedia.org/wiki/Circuit_Zandvoort
13  http://en.wikipedia.org/wiki/Autodromo_Naziona...
14  http://en.wikipedia.org/wiki/Marina_Bay_Street...
15       http://en.wikipedia.org/wiki/Suzuka_Circuit
16  http://en.wikipedia.org/wiki/Losail_Internatio...
17  http://en.wikipedia.org/wiki/Circuit_of_the_Am...
18  http://en.wikipedia.org/wiki/Aut%C3%B3dromo_He...
19  http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Jo...
20  https://en.wikipedia.org/wiki/Las_Vegas_Grand_...
21    http://en.wikipedia.org/wiki/Yas_Marina_Circuit
```

Extraemos las curvas de cada carrera

```
def extract_turns(url):
    try:
        # Paso 1: Hacer una solicitud al enlace principal
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.text, 'html.parser')

        # Paso 2: Extraer información del elemento 'Turns'
        turns_element = soup.find('th', class_='infobox-label',
text='Turns')  # Etiqueta con clase y texto específico
        if not turns_element:
            return None  # No se encontró el elemento 'Turns'

        # Paso 3: Buscar el valor asociado a 'Turns' en la siguiente
columna (td)
        turns_value = turns_element.find_next('td', class_='infobox-
data').text.strip()
        return turns_value

    except requests.RequestException as e:
        print(f"Error al procesar {url}: {e}")
        return None

races_2023['turns'] = races_2023['url_c'].apply(extract_turns)
```

```
C:\Users\gonza\AppData\Local\Temp\ipykernel_24408\2643165226.py:9:
DeprecationWarning: The 'text' argument to find()-type methods is
deprecated. Use 'string' instead.
  turns_element = soup.find('th', class_='infobox-label',
text='Turns')  # Etiqueta con clase y texto específico
```

```
display(races_2023)
```

	raceId	year	circuitId	\
0	1098	2023	3	
1	1099	2023	77	
2	1100	2023	1	
3	1101	2023	73	
4	1102	2023	79	
5	1104	2023	6	
6	1105	2023	4	
7	1106	2023	7	
8	1107	2023	70	
9	1108	2023	9	
10	1109	2023	11	
11	1110	2023	13	
12	1111	2023	39	
13	1112	2023	14	
14	1113	2023	15	
15	1114	2023	22	
16	1115	2023	78	
17	1116	2023	69	
18	1117	2023	32	
19	1118	2023	18	
20	1119	2023	80	
21	1120	2023	24	

	url	\
0	https://en.wikipedia.org/wiki/2023_Bahrain_Gra...	
1	https://en.wikipedia.org/wiki/2023_Saudi_Arabi...	
2	https://en.wikipedia.org/wiki/2023_Australian_...	
3	https://en.wikipedia.org/wiki/2023_Azerbaijan_...	
4	https://en.wikipedia.org/wiki/2023_Miami_Grand...	
5	https://en.wikipedia.org/wiki/2023_Monaco_Gran...	
6	https://en.wikipedia.org/wiki/2023_Spanish_Gra...	
7	https://en.wikipedia.org/wiki/2023_Canadian_Gr...	
8	https://en.wikipedia.org/wiki/2023_Austrian_Gr...	
9	https://en.wikipedia.org/wiki/2023_British_Gra...	
10	https://en.wikipedia.org/wiki/2023_Hungarian_G...	
11	https://en.wikipedia.org/wiki/2023_Belgian_Gra...	
12	https://en.wikipedia.org/wiki/2023_Dutch_Grand...	
13	https://en.wikipedia.org/wiki/2023_Italian_Gra...	
14	https://en.wikipedia.org/wiki/2023_Singapore_G...	
15	https://en.wikipedia.org/wiki/2023_Japanese_Gr...	
16	https://en.wikipedia.org/wiki/2023_Qatar_Grand...	
17	https://en.wikipedia.org/wiki/2023_United_Stat...	
18	https://en.wikipedia.org/wiki/2023_Mexico_City...	
19	https://en.wikipedia.org/wiki/2023_S%C3%A3o_Pa...	
20	https://en.wikipedia.org/wiki/2023_Las_Vegas_G...	
21	https://en.wikipedia.org/wiki/2023_Abu_Dhabi_G...	

```
                                weather  Course Length (m)  Laps  \
0                                 Clear             5412.0    57
1                                 Clear             6174.0    50
2                                 Sunny             5278.0    58
3                         Partly cloudy             6003.0    51
4                         Partly cloudy             5412.0    57
5      Cloudy, with some rain intervals             3337.0    78
6                                Cloudy             4657.0    66
7                         Partly cloudy             4361.0    70
8                                Cloudy             4318.0    71
9                         Partly cloudy             5891.0    52
10                                Sunny             4381.0    70
11  Partly cloudy, with a rain interval             7004.0    44
12                      Cloudy and rainy             4259.0    72
13                                Sunny             5793.0    51
14                                Clear             4940.0    62
15                                Sunny             5807.0    53
16                                Clear             5419.0    57
17                                Sunny             5513.0    56
18                                Sunny             4304.0    71
19                        Partly cloudy             4309.0    71
20                                Clear             6201.0    50
21                                Clear             5281.0    58

                                                   url_c turns
duracion  \
0   http://en.wikipedia.org/wiki/Bahrain_Internati...    15
01:33:56.736
1   http://en.wikipedia.org/wiki/Jeddah_Street_Cir...    27
01:21:14.894
2   http://en.wikipedia.org/wiki/Melbourne_Grand_P...    14
02:32:38.371
3      http://en.wikipedia.org/wiki/Baku_City_Circuit    20
01:32:42.436
4   http://en.wikipedia.org/wiki/Miami_Internation...    19
01:27:38.241
5      http://en.wikipedia.org/wiki/Circuit_de_Monaco    19
01:48:51.980
6   http://en.wikipedia.org/wiki/Circuit_de_Barcel...    14
01:27:57.940
7   http://en.wikipedia.org/wiki/Circuit_Gilles_Vi...    14
01:33:58.348
8          http://en.wikipedia.org/wiki/Red_Bull_Ring    10
01:25:33.607
9    http://en.wikipedia.org/wiki/Silverstone_Circuit    18
01:25:16.938
10           http://en.wikipedia.org/wiki/Hungaroring    14
01:38:08.634
11  http://en.wikipedia.org/wiki/Circuit_de_Spa-Fr...    19
```

```
01:22:30.450
12     http://en.wikipedia.org/wiki/Circuit_Zandvoort    14
02:24:04.411
13  http://en.wikipedia.org/wiki/Autodromo_Naziona...    11
01:13:41.143
14  http://en.wikipedia.org/wiki/Marina_Bay_Street...    19
01:46.37.418
15        http://en.wikipedia.org/wiki/Suzuka_Circuit    18
01:30:58.421
16  http://en.wikipedia.org/wiki/Losail_Internatio...    16
01:27:39.168
17  http://en.wikipedia.org/wiki/Circuit_of_the_Am...    20
01:35:21.362
18  http://en.wikipedia.org/wiki/Aut%C3%B3dromo_He...    17
02:02:30.814
19  http://en.wikipedia.org/wiki/Aut%C3%B3dromo_Jo...    15
01:56:48.894
20  https://en.wikipedia.org/wiki/Las_Vegas_Grand_...    17
01:29:08.289
21    http://en.wikipedia.org/wiki/Yas_Marina_Circuit    16
01:27:02.624

    adelantamientos
0                37
1                36
2                29
3                23
4                62
5                22
6                65
7                20
8                63
9                26
10               16
11               75
12              188
13               31
14               42
15               29
16               48
17               46
18               48
19               25
20               99
21               70
```

Algunos datos no se han extraido en el formato deseado, los cambiamos manualmente

```
races_2023.at[8,'turns']=10
races_2023.at[18,'turns']=17
races_2023.at[20,'turns']=17
```

Incluimos también información extraida a mano

```
duracion=['01:33:56.736','01:21:14.894','02:32:38.371','01:32:42.436',
'01:27:38.241','01:48:51.980','01:27:57.940','01:33:58.348','01:25:33.
607','01:25:16.938','01:38:08.634','01:22:30.450','02:24:04.411','01:1
3:41.143','01:46:37.418','01:30:58.421','01:27:39.168','01:35:21.362',
'02:02:30.814','01:56:48.894','01:29:08.289','01:27:02.624']
adelantamientos=[37,36,29,23,62,22,65,20,63,26,16,75,188,31,42,29,48,4
6,48,25,99,70]
races_2023['duracion']=duracion
races_2023['adelantamientos']=adelantamientos
```

Cambiamos el tiempo extraido a milisegundos

```
def tiempo_a_milisegundos(tiempo):
    horas, minutos, segundos = tiempo.split(':')
    segundos, milisegundos = segundos.split('.')

    # Convertir a milisegundos
    total_milisegundos = (int(horas) * 3600 + int(minutos) * 60 +
int(segundos)) * 1000 + int(milisegundos)
    return total_milisegundos

races_2023['duracion'] =
races_2023['duracion'].apply(tiempo_a_milisegundos)
```

Cargamos el dataframe resultante del análisis de sentimientos

```
sentimientos=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\
sentimientos_carreras.csv")
sentimientos=sentimientos.drop(columns=['promedio','mediana','desviaci
on_std','comentarios_totales'])
races_2023=pd.merge(races_2023,sentimientos,on='raceId')
```

Aplicamos un mapeo a la variable categórica del clima

```
# Definir los mapeos
mapeo_clima = {'Sunny': 1, 'Clear': 2, 'Partly
cloudy':3,'Cloudy':4,'Partly cloudy, with a rain interval':5,'Cloudy,
with some rain intervals':6,'Cloudy and rainy':7}
```

```
# Aplicar map en ambas columnas
races_2023['weather'] = races_2023['weather'].map(mapeo_clima)

display(races_2023)
```

```
    raceId  year  circuitId  weather  Course Length (m)  Laps turns
duracion  \
0     1098  2023          3        2             5412.0    57    15
5636736
1     1099  2023         77        2             6174.0    50    27
4874894
2     1100  2023          1        1             5278.0    58    14
9158371
3     1101  2023         73        3             6003.0    51    20
5562436
4     1102  2023         79        3             5412.0    57    19
5258241
5     1104  2023          6        6             3337.0    78    19
6531980
6     1105  2023          4        4             4657.0    66    14
5277940
7     1106  2023          7        3             4361.0    70    14
5638348
8     1107  2023         70        4             4318.0    71    10
5133607
9     1108  2023          9        3             5891.0    52    18
5116938
10    1109  2023         11        1             4381.0    70    14
5888634
11    1110  2023         13        5             7004.0    44    19
4950450
12    1111  2023         39        7             4259.0    72    14
8644411
13    1112  2023         14        1             5793.0    51    11
4421143
14    1113  2023         15        2             4940.0    62    19
6397418
15    1114  2023         22        1             5807.0    53    18
5458421
16    1115  2023         78        2             5419.0    57    16
5259168
17    1116  2023         69        1             5513.0    56    20
5721362
18    1117  2023         32        1             4304.0    71    17
7350814
19    1118  2023         18        3             4309.0    71    15
7008894
20    1119  2023         80        2             6201.0    50    17
5348289
21    1120  2023         24        2             5281.0    58    16
```

```
5222624
```

	adelantamientos	clasificacion
0	37	0
1	36	0
2	29	0
3	23	0
4	62	0
5	22	0
6	65	0
7	20	1
8	63	0
9	26	1
10	16	0
11	75	1
12	188	1
13	31	0
14	42	1
15	29	1
16	48	0
17	46	0
18	48	0
19	25	0
20	99	1
21	70	0

Eliminamos las columnas de las urls y cargamos el dataframe con el mismo formato de 2022

```
races_2023=races_2023.drop(columns=['url','url_c'])
races_2022=pd.read_csv(r"C:\Users\gonza\Downloads\races_2022.csv")
races_2023=pd.concat([races_2023,races_2022],axis=0,
ignore_index=True)
```

Definimos las variables explicativas y la etiqueta para entrenar un modelo de RandomForest del que poder extraer la influencia de características

```
X=races_2023.drop(columns=['raceId','year','circuitId','clasificacion'
])
y=races_2023['clasificacion']

from sklearn.ensemble import RandomForestClassifier

# Entrenar el modelo
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X, y)

# Obtener la importancia de las características
importancia_rf = rf_model.feature_importances_
```

```python
# Mostrar las características más importantes
feature_importance_rf = pd.DataFrame({
    'Característica': X.columns,
    'Importancia': importancia_rf
})
feature_importance_rf =
feature_importance_rf.sort_values(by='Importancia', ascending=False)
print(feature_importance_rf)
```

```
      Característica  Importancia
1  Course Length (m)     0.238619
5    adelantamientos     0.236518
2               Laps     0.140671
3              turns     0.138579
4           duracion     0.138473
0            weather     0.107140
```

ANEXO 6

Predicción dominancia Multilayer Perceptrón

En este fichero se hace una preparación de los datos que se le va a meter a la red nueronal y luego se entrena para que prediga la probabilidad de que los equipos se encuentren en una época de dominancia

De aquí en adelante es una consecución de preparación de los datos para dar como output de la red neuronal cada resultado de cada piloto en cada carrera

```
import pandas as pd
import numpy as np
from IPython.display import display

circuits=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
circuits.csv",na_values="\\N")
constructor_results=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
constructor_results.csv",na_values="\\N")
constructor_standings=pd.read_csv(r"C:\Users\gonza\OneDrive\
Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE
DE DATOS\constructor_standings.csv",na_values="\\N")
constructors=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
constructors.csv",na_values="\\N")
driver_standings=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
driver_standings.csv",na_values="\\N")
drivers=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
drivers.csv",na_values="\\N")
lap_times=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
lap_times.csv",na_values="\\N")
pit_stops=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
pit_stops.csv",na_values="\\N")
qualifying=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
qualifying.csv",na_values="\\N")
races=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\4to
curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
races.csv",na_values="\\N")
results=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
results.csv",na_values="\\N")
seasons=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
seasons.csv",na_values="\\N")
```

```
sprint_results=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
sprint_results.csv",na_values="\\N")
status=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
status.csv",na_values="\\N")

from IPython.display import display

prueba=pd.merge(races,circuits, on='circuitId')
display(prueba)
prueba.info()
```

```
      raceId  year  round  circuitId                  name_x
date  \
0          1  2009      1          1   Australian Grand Prix  2009-03-
29
1          2  2009      2          2    Malaysian Grand Prix  2009-04-
05
2          3  2009      3         17      Chinese Grand Prix  2009-04-
19
3          4  2009      4          3      Bahrain Grand Prix  2009-04-
26
4          5  2009      5          4      Spanish Grand Prix  2009-05-
10
...      ...   ...    ...        ...                     ...         .
..
1120    1140  2024     20         32  Mexico City Grand Prix  2024-10-
27
1121    1141  2024     21         18    São Paulo Grand Prix  2024-11-
03
1122    1142  2024     22         80    Las Vegas Grand Prix  2024-11-
23
1123    1143  2024     23         78        Qatar Grand Prix  2024-12-
01
1124    1144  2024     24         24    Abu Dhabi Grand Prix  2024-12-
08

          time                                              url_x
fp1_date  \
0     06:00:00  http://en.wikipedia.org/wiki/2009_Australian_G...
NaN
1     09:00:00  http://en.wikipedia.org/wiki/2009_Malaysian_Gr...
NaN
2     07:00:00  http://en.wikipedia.org/wiki/2009_Chinese_Gran...
NaN
3     12:00:00  http://en.wikipedia.org/wiki/2009_Bahrain_Gran...
NaN
4     12:00:00  http://en.wikipedia.org/wiki/2009_Spanish_Gran...
NaN
```

```
...        ...                                                  ...
...
1120  20:00:00  https://en.wikipedia.org/wiki/2024_Mexico_City...
2024-10-25
1121  17:00:00  https://en.wikipedia.org/wiki/2024_S%C3%A3o_Pa...
2024-11-01
1122  06:00:00  https://en.wikipedia.org/wiki/2024_Las_Vegas_G...
2024-11-21
1123  17:00:00  https://en.wikipedia.org/wiki/2024_Qatar_Grand...
2024-11-29
1124  13:00:00  https://en.wikipedia.org/wiki/2024_Abu_Dhabi_G...
2024-12-06

      fp1_time  ... sprint_date sprint_time   circuitRef  \
0          NaN  ...         NaN         NaN  albert_park
1          NaN  ...         NaN         NaN       sepang
2          NaN  ...         NaN         NaN     shanghai
3          NaN  ...         NaN         NaN      bahrain
4          NaN  ...         NaN         NaN    catalunya
...        ...  ...         ...         ...          ...
1120  18:30:00  ...         NaN         NaN    rodriguez
1121  14:30:00  ...  2024-11-02    14:00:00   interlagos
1122  02:30:00  ...         NaN         NaN        vegas
1123  13:30:00  ...  2024-11-30    13:00:00       losail
1124  09:30:00  ...         NaN         NaN   yas_marina

                               name_y      location        country
lat  \
0     Albert Park Grand Prix Circuit     Melbourne      Australia -
37.84970
1       Sepang International Circuit  Kuala Lumpur       Malaysia
2.76083
2     Shanghai International Circuit      Shanghai          China
31.33890
3      Bahrain International Circuit        Sakhir        Bahrain
26.03250
4     Circuit de Barcelona-Catalunya      Montmeló          Spain
41.57000
...                              ...           ...            ...
...
1120    Autódromo Hermanos Rodríguez   Mexico City         Mexico
19.40420
1121      Autódromo José Carlos Pace     São Paulo         Brazil -
23.70360
1122  Las Vegas Strip Street Circuit     Las Vegas  United States
36.11470
1123    Losail International Circuit     Al Daayen          Qatar
25.49000
1124              Yas Marina Circuit     Abu Dhabi            UAE
```

```
24.46720

             lng   alt
url_y
0     144.96800    10
http://en.wikipedia.org/wiki/Melbourne_Grand_P...
1     101.73800    18
http://en.wikipedia.org/wiki/Sepang_Internatio...
2     121.22000     5
http://en.wikipedia.org/wiki/Shanghai_Internat...
3      50.51060     7
http://en.wikipedia.org/wiki/Bahrain_Internati...
4       2.26111   109
http://en.wikipedia.org/wiki/Circuit_de_Barcel...
...         ...   ...                                                 .
..
1120  -99.09070  2227  http://en.wikipedia.org/wiki/Aut
%C3%B3dromo_He...
1121  -46.69970   785  http://en.wikipedia.org/wiki/Aut
%C3%B3dromo_Jo...
1122 -115.17300   642
https://en.wikipedia.org/wiki/Las_Vegas_Grand_...
1123   51.45420    12
http://en.wikipedia.org/wiki/Losail_Internatio...
1124   54.60310     3
http://en.wikipedia.org/wiki/Yas_Marina_Circuit

[1125 rows x 26 columns]

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1125 entries, 0 to 1124
Data columns (total 26 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   raceId       1125 non-null   int64
 1   year         1125 non-null   int64
 2   round        1125 non-null   int64
 3   circuitId    1125 non-null   int64
 4   name_x       1125 non-null   object
 5   date         1125 non-null   object
 6   time         394 non-null    object
 7   url_x        1125 non-null   object
 8   fp1_date     90 non-null     object
 9   fp1_time     68 non-null     object
 10  fp2_date     90 non-null     object
 11  fp2_time     68 non-null     object
 12  fp3_date     72 non-null     object
 13  fp3_time     53 non-null     object
 14  quali_date   90 non-null     object
 15  quali_time   68 non-null     object
```

```
 16  sprint_date  18 non-null     object
 17  sprint_time  15 non-null     object
 18  circuitRef   1125 non-null   object
 19  name_y       1125 non-null   object
 20  location     1125 non-null   object
 21  country      1125 non-null   object
 22  lat          1125 non-null   float64
 23  lng          1125 non-null   float64
 24  alt          1125 non-null   int64
 25  url_y        1125 non-null   object
dtypes: float64(2), int64(5), object(19)
memory usage: 228.6+ KB
```

```
prueba2=pd.merge(results,prueba, on='raceId')
display(prueba2)
prueba2.info()
```

```
       resultId  raceId  driverId  constructorId  number  grid
position  \
0             1      18         1              1    22.0     1
1.0
1             2      18         2              2     3.0     5
2.0
2             3      18         3              3     7.0     7
3.0
3             4      18         4              4     5.0    11
4.0
4             5      18         5              1    23.0     3
5.0
...         ...     ...       ...            ...     ...   ...       .
..
26514     26520    1132       839            214    31.0    18
16.0
26515     26521    1132       815              9    11.0     0
17.0
26516     26522    1132       855             15    24.0    14
18.0
26517     26523    1132       847            131    63.0     1
NaN
26518     26524    1132       842            214    10.0    19
NaN

      positionText  positionOrder  points  ...  sprint_date
sprint_time  \
0                1              1    10.0  ...          NaN
NaN
1                2              2     8.0  ...          NaN
NaN
2                3              3     6.0  ...          NaN
NaN
```

```
3                4              4     5.0  ...          NaN
NaN
4                5              5     4.0  ...          NaN
NaN
...            ...            ...     ...  ...          ...          ..
.
26514           16             16     0.0  ...          NaN
NaN
26515           17             17     0.0  ...          NaN
NaN
26516           18             18     0.0  ...          NaN
NaN
26517            R             19     0.0  ...          NaN
NaN
26518            W             20     0.0  ...          NaN
NaN

        circuitRef                          name_y     location
country  \
0      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia
1      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia
2      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia
3      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia
4      albert_park  Albert Park Grand Prix Circuit    Melbourne
Australia
...            ...                               ...          ...
...
26514  silverstone             Silverstone Circuit  Silverstone
UK
26515  silverstone             Silverstone Circuit  Silverstone
UK
26516  silverstone             Silverstone Circuit  Silverstone
UK
26517  silverstone             Silverstone Circuit  Silverstone
UK
26518  silverstone             Silverstone Circuit  Silverstone
UK

           lat        lng  alt  \
0     -37.8497  144.96800   10
1     -37.8497  144.96800   10
2     -37.8497  144.96800   10
3     -37.8497  144.96800   10
4     -37.8497  144.96800   10
...        ...        ...  ...
```

```
26514  52.0786   -1.01694  153
26515  52.0786   -1.01694  153
26516  52.0786   -1.01694  153
26517  52.0786   -1.01694  153
26518  52.0786   -1.01694  153

                                                   url_y
0      http://en.wikipedia.org/wiki/Melbourne_Grand_P...
1      http://en.wikipedia.org/wiki/Melbourne_Grand_P...
2      http://en.wikipedia.org/wiki/Melbourne_Grand_P...
3      http://en.wikipedia.org/wiki/Melbourne_Grand_P...
4      http://en.wikipedia.org/wiki/Melbourne_Grand_P...
...                                                  ...
26514   http://en.wikipedia.org/wiki/Silverstone_Circuit
26515   http://en.wikipedia.org/wiki/Silverstone_Circuit
26516   http://en.wikipedia.org/wiki/Silverstone_Circuit
26517   http://en.wikipedia.org/wiki/Silverstone_Circuit
26518   http://en.wikipedia.org/wiki/Silverstone_Circuit

[26519 rows x 43 columns]

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26519 entries, 0 to 26518
Data columns (total 43 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   resultId         26519 non-null  int64
 1   raceId           26519 non-null  int64
 2   driverId         26519 non-null  int64
 3   constructorId    26519 non-null  int64
 4   number           26513 non-null  float64
 5   grid             26519 non-null  int64
 6   position         15591 non-null  float64
 7   positionText     26519 non-null  object
 8   positionOrder    26519 non-null  int64
 9   points           26519 non-null  float64
 10  laps             26519 non-null  int64
 11  time_x           7533 non-null   object
 12  milliseconds     7533 non-null   float64
 13  fastestLap       8020 non-null   float64
 14  rank             8270 non-null   float64
 15  fastestLapTime   8020 non-null   object
 16  fastestLapSpeed  8020 non-null   float64
 17  statusId         26519 non-null  int64
 18  year             26519 non-null  int64
 19  round            26519 non-null  int64
 20  circuitId        26519 non-null  int64
 21  name_x           26519 non-null  object
 22  date             26519 non-null  object
 23  time_y           8050 non-null   object
```

```
 24  url_x            26519 non-null  object
 25  fp1_date         1559 non-null   object
 26  fp1_time         1119 non-null   object
 27  fp2_date         1559 non-null   object
 28  fp2_time         1119 non-null   object
 29  fp3_date         1259 non-null   object
 30  fp3_time         879 non-null    object
 31  quali_date       1559 non-null   object
 32  quali_time       1119 non-null   object
 33  sprint_date      300 non-null    object
 34  sprint_time      240 non-null    object
 35  circuitRef       26519 non-null  object
 36  name_y           26519 non-null  object
 37  location         26519 non-null  object
 38  country          26519 non-null  object
 39  lat              26519 non-null  float64
 40  lng              26519 non-null  float64
 41  alt              26519 non-null  int64
 42  url_y            26519 non-null  object
dtypes: float64(9), int64(12), object(22)
memory usage: 8.7+ MB
```

```
prueba3=pit_stops.drop(columns=['stop','lap','time','duration'])

prueba3=(pit_stops.groupby(['raceId', 'driverId'])
    .agg(mean_pit_stop=('milliseconds', 'mean'))  # Calcular la media
    .reset_index())

display(prueba3)
prueba3.info()
```

```
      raceId  driverId  mean_pit_stop
0        841         1        23213.0
1        841         2        24046.0
2        841         3        23716.0
3        841         4        24055.0
4        841         5        24865.0
...      ...       ...            ...
5341    1132       848        32076.0
5342    1132       852        29829.0
5343    1132       855        30306.0
5344    1132       857        29370.5
5345    1132       858        31122.5

[5346 rows x 3 columns]

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5346 entries, 0 to 5345
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype
```

```
---  ------          --------------  -----
 0   raceId          5346 non-null   int64
 1   driverId        5346 non-null   int64
 2   mean_pit_stop   5346 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 125.4 KB
```

```
prueba5=races[['raceId','year']]
display(prueba5)
```

```
      raceId  year
0          1  2009
1          2  2009
2          3  2009
3          4  2009
4          5  2009
...      ...   ...
1120    1140  2024
1121    1141  2024
1122    1142  2024
1123    1143  2024
1124    1144  2024

[1125 rows x 2 columns]
```

```
prueba4=qualifying.drop(columns=['constructorId','number','position'])
prueba4=pd.merge(results,prueba4, on=['raceId','driverId'],how='left')
prueba4=pd.merge(prueba4,prueba3, on=['raceId','driverId'],how='left')
prueba4=pd.merge(prueba4,prueba5, on=['raceId'])
display(prueba4)
prueba4.info()
```

```
       resultId  raceId  driverId  constructorId  number  grid
position  \
0             1      18         1              1    22.0     1
1.0
1             2      18         2              2     3.0     5
2.0
2             3      18         3              3     7.0     7
3.0
3             4      18         4              4     5.0    11
4.0
4             5      18         5              1    23.0     3
5.0
...         ...     ...       ...            ...     ...   ...       .
..
26514     26520    1132       839            214    31.0    18
16.0
26515     26521    1132       815              9    11.0     0
```

```
17.0
26516     26522    1132       855             15    24.0    14
18.0
26517     26523    1132       847            131    63.0     1
NaN
26518     26524    1132       842            214    10.0    19
NaN

      positionText  positionOrder  points  ...  rank fastestLapTime  \
0                1              1    10.0  ...   2.0       1:27.452
1                2              2     8.0  ...   3.0       1:27.739
2                3              3     6.0  ...   5.0       1:28.090
3                4              4     5.0  ...   7.0       1:28.603
4                5              5     4.0  ...   1.0       1:27.418
...            ...            ...     ...  ...   ...            ...
26514           16             16     0.0  ...  16.0       1:30.875
26515           17             17     0.0  ...   6.0       1:29.707
26516           18             18     0.0  ...  17.0       1:31.014
26517            R             19     0.0  ...  19.0       1:31.298
26518            W             20     0.0  ...   0.0            NaN

       fastestLapSpeed  statusId  qualifyId        q1         q2
q3  \
0              218.300         1        1.0  1:26.572  1:25.187
1:26.714
1              217.586         1        5.0  1:25.960  1:25.518
1:27.236
2              216.719         1        7.0  1:26.295  1:26.059
1:28.687
3              215.464         1       12.0  1:26.907  1:26.188
NaN
4              218.385         1        3.0  1:25.664  1:25.452
1:27.079
...                ...       ...        ...       ...       ...
...
26514          233.371        12    10309.0  1:34.557       NaN
NaN
26515          236.409        12    10310.0  1:38.348       NaN
NaN
26516          233.014        12    10305.0  1:31.190  1:27.867
NaN
26517          232.289        34    10292.0  1:30.106  1:26.723
1:25.819
26518              NaN         6    10311.0  1:39.804       NaN
NaN

       mean_pit_stop  year
0                NaN  2008
1                NaN  2008
2                NaN  2008
```

```
3                NaN  2008
4                NaN  2008
...              ...   ...
26514       28786.50  2024
26515       29718.75  2024
26516       30306.00  2024
26517       32045.00  2024
26518            NaN  2024

[26519 rows x 24 columns]

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26519 entries, 0 to 26518
Data columns (total 24 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   resultId         26519 non-null  int64  
 1   raceId           26519 non-null  int64  
 2   driverId         26519 non-null  int64  
 3   constructorId    26519 non-null  int64  
 4   number           26513 non-null  float64
 5   grid             26519 non-null  int64  
 6   position         15591 non-null  float64
 7   positionText     26519 non-null  object 
 8   positionOrder    26519 non-null  int64  
 9   points           26519 non-null  float64
 10  laps             26519 non-null  int64  
 11  time             7533 non-null   object 
 12  milliseconds     7533 non-null   float64
 13  fastestLap       8020 non-null   float64
 14  rank             8270 non-null   float64
 15  fastestLapTime   8020 non-null   object 
 16  fastestLapSpeed  8020 non-null   float64
 17  statusId         26519 non-null  int64  
 18  qualifyId        10254 non-null  float64
 19  q1               10100 non-null  object 
 20  q2               5669 non-null   object 
 21  q3               3514 non-null   object 
 22  mean_pit_stop    5346 non-null   float64
 23  year             26519 non-null  int64  
dtypes: float64(9), int64(9), object(6)
memory usage: 4.9+ MB
```

```
prueba4=prueba4.drop(columns=['positionText','time','fastestLap','qual
ifyId'])

prueba4['fastestLapTime'] = prueba4['fastestLapTime'].apply(str)
prueba4['q1']=prueba4['q1'].apply(str)
prueba4['q2']=prueba4['q2'].apply(str)
prueba4['q3']=prueba4['q3'].apply(str)
```

```
def time_to_milliseconds(time_str):

    if time_str == 'nan' or time_str == 'None' or time_str.strip() ==
'':
        return np.nan  # o podrías retornar un valor por defecto, como
0

    parts = time_str.split(':')

    if len(parts) == 2:  # Formato 'm:ss.sss'
        minutes, seconds = map(float, parts)
        return int((minutes * 60 + seconds) * 1000)

    elif len(parts) == 3:  # Formato 'h:mm:ss.sss'
        hours, minutes, seconds = map(float, parts)
        return int((hours * 3600 + minutes * 60 + seconds) * 1000)

    else:
        raise ValueError(f"Formato desconocido para el tiempo:
{time_str}")

prueba4['fastestLapTime']=prueba4['fastestLapTime'].apply(time_to_mill
iseconds)
prueba4['q1']=prueba4['q1'].apply(time_to_milliseconds)
prueba4['q2']=prueba4['q2'].apply(time_to_milliseconds)
prueba4['q3']=prueba4['q3'].apply(time_to_milliseconds)
display(prueba4)
```

```
       resultId  raceId  driverId  constructorId  number  grid
position  \
0             1      18         1              1    22.0     1
1.0
1             2      18         2              2     3.0     5
2.0
2             3      18         3              3     7.0     7
3.0
3             4      18         4              4     5.0    11
4.0
4             5      18         5              1    23.0     3
5.0
...         ...     ...       ...            ...     ...   ...       .
..
26514     26520    1132       839            214    31.0    18
16.0
26515     26521    1132       815              9    11.0     0
17.0
26516     26522    1132       855             15    24.0    14
18.0
26517     26523    1132       847            131    63.0     1
```

```
NaN
26518     26524    1132        842             214    10.0    19
NaN

       positionOrder  points  laps  milliseconds  rank  fastestLapTime
\
0                  1    10.0    58     5690616.0   2.0         87452.0

1                  2     8.0    58     5696094.0   3.0         87739.0

2                  3     6.0    58     5698779.0   5.0         88090.0

3                  4     5.0    58     5707797.0   7.0         88603.0

4                  5     4.0    58     5708630.0   1.0         87418.0

...              ...     ...   ...           ...   ...             ...

26514             16     0.0    50           NaN  16.0         90875.0

26515             17     0.0    50           NaN   6.0         89707.0

26516             18     0.0    50           NaN  17.0         91014.0

26517             19     0.0    33           NaN  19.0         91298.0

26518             20     0.0     0           NaN   0.0             NaN


       fastestLapSpeed  statusId       q1       q2       q3
mean_pit_stop  \
0              218.300         1  86572.0  85187.0  86714.0
NaN
1              217.586         1  85960.0  85518.0  87236.0
NaN
2              216.719         1  86295.0  86059.0  88687.0
NaN
3              215.464         1  86907.0  86188.0      NaN
NaN
4              218.385         1  85664.0  85452.0  87079.0
NaN
...                ...       ...      ...      ...      ...
...
26514          233.371        12  94557.0      NaN      NaN
28786.50
26515          236.409        12  98348.0      NaN      NaN
29718.75
26516          233.014        12  91190.0  87867.0      NaN
30306.00
26517          232.289        34  90106.0  86723.0  85819.0
```

```
32045.00
26518               NaN         6  99804.0      NaN      NaN
NaN

       year
0      2008
1      2008
2      2008
3      2008
4      2008
...     ...
26514  2024
26515  2024
26516  2024
26517  2024
26518  2024

[26519 rows x 20 columns]

prueba4.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26519 entries, 0 to 26518
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   resultId         26519 non-null  int64
 1   raceId           26519 non-null  int64
 2   driverId         26519 non-null  int64
 3   constructorId    26519 non-null  int64
 4   number           26513 non-null  float64
 5   grid             26519 non-null  int64
 6   position         15591 non-null  float64
 7   positionOrder    26519 non-null  int64
 8   points           26519 non-null  float64
 9   laps             26519 non-null  int64
 10  milliseconds     7533 non-null   float64
 11  rank             8270 non-null   float64
 12  fastestLapTime   8020 non-null   float64
 13  fastestLapSpeed  8020 non-null   float64
 14  statusId         26519 non-null  int64
 15  q1               10100 non-null  float64
 16  q2               5669 non-null   float64
 17  q3               3514 non-null   float64
 18  mean_pit_stop    5346 non-null   float64
 19  year             26519 non-null  int64
dtypes: float64(11), int64(9)
memory usage: 4.0 MB
```

```
prueba4['milliseconds_missing'] =
(((~prueba4['statusId'].isin([1,11,12,13,14,15,16,17,18,19]))) &
prueba4['milliseconds'].isna()).astype(int)
prueba4['fastestLapTime_missing'] =
(((~prueba4['statusId'].isin([1,11,12,13,14,15,16,17,18,19]))) &
prueba4['fastestLapTime'].isna()).astype(int)
prueba4['fastestLapSpeed_missing'] =
(((~prueba4['statusId'].isin([1,11,12,13,14,15,16,17,18,19]))) &
prueba4['fastestLapSpeed'].isna()).astype(int)
prueba4['q1_missing'] =
(((~prueba4['statusId'].isin([1,11,12,13,14,15,16,17,18,19]))) &
prueba4['q1'].isna()).astype(int)
prueba4['q2_missing'] = prueba4['q2'].isna().astype(int)
prueba4['q3_missing'] = prueba4['q3'].isna().astype(int)
prueba4['mean_pit_stop_missing'] =
(((~prueba4['statusId'].isin([1,11,12,13,14,15,16,17,18,19]))) &
prueba4['mean_pit_stop'].isna()).astype(int)
prueba4['position_missing'] =
(((~prueba4['statusId'].isin([1,11,12,13,14,15,16,17,18,19]))) &
prueba4['position'].isna()).astype(int)
```

Aquí se pasa a imputar los valores faltantes con valores exagerados

```
# Paso 1: Calcular los valores por grupo (raceId)
grouped = prueba4.groupby('raceId')

# Calcular máximo y media de 'time' por cada 'raceId'
prueba4['max_time_by_race'] = grouped['milliseconds'].transform('max')
prueba4['mean_time_by_race'] =
grouped['milliseconds'].transform('mean')
prueba4['max_fastest_lap_by_race'] =
grouped['fastestLapTime'].transform('max')
prueba4['mean_fastest_lap_by_race'] =
grouped['fastestLapTime'].transform('mean')
prueba4['min_fastest_lap_speed_by_race'] =
grouped['fastestLapSpeed'].transform('min')
prueba4['mean_fastest_lap_speed_by_race'] =
grouped['fastestLapSpeed'].transform('mean')
prueba4['max_q1_by_race'] = grouped['q1'].transform('max')
prueba4['mean_q1_by_race'] = grouped['q1'].transform('mean')


# Paso 2: Calcular los valores generales
max_time_general = prueba4['milliseconds'].max()
mean_time_general = prueba4['milliseconds'].mean()
max_fastest_lap_general = prueba4['fastestLapTime'].max()
mean_fastest_lap_general = prueba4['fastestLapTime'].mean()
min_lap_speed_general = prueba4['fastestLapSpeed'].min()
mean_lap_speed_general = prueba4['fastestLapSpeed'].mean()
```

```
max_q1_general = prueba4['q1'].max()
mean_q1_general = prueba4['q1'].mean()



# Paso 3: Crear una función para imputar valores
def imputar_tiempo(row):
    if row['statusId'] == 11:
        return row['max_time_by_race'] + 0.1 *
row['mean_time_by_race']
    elif row['statusId'] == 12:
        return row['max_time_by_race'] + 0.2 *
row['mean_time_by_race']
    elif row['statusId'] == 13:
        return row['max_time_by_race'] + 0.3 *
row['mean_time_by_race']
    elif row['statusId'] == 14:
        return row['max_time_by_race'] + 0.4 *
row['mean_time_by_race']
    elif row['statusId'] == 15:
        return row['max_time_by_race'] + 0.5 *
row['mean_time_by_race']
    elif row['statusId'] == 16:
        return row['max_time_by_race'] + 0.6 *
row['mean_time_by_race']
    elif row['statusId'] == 17:
        return row['max_time_by_race'] + 0.7 *
row['mean_time_by_race']
    elif row['statusId'] == 18:
        return row['max_time_by_race'] + 0.8 *
row['mean_time_by_race']
    elif row['statusId'] == 19:
        return row['max_time_by_race'] + 0.9 *
row['mean_time_by_race']
    elif row['statusId'] != 1 and pd.isnull(row['milliseconds']):
        return max_time_general + mean_time_general
    elif pd.isnull(row['milliseconds']):
        return max_time_general + mean_time_general
    else:
        return row['milliseconds']  # Dejar el valor original si no
cumple las condiciones

# Paso 3: Crear una función para imputar valores
def imputar_vuelta_rapida(row):
    if pd.isnull(row['fastestLapTime']) and
pd.notna(row['max_fastest_lap_by_race']):
        return row['max_fastest_lap_by_race'] +
row['mean_fastest_lap_by_race']
    elif pd.isnull(row['fastestLapTime']):
        return max_fastest_lap_general + mean_fastest_lap_general
```

```
    else:
        return row['fastestLapTime']  # Dejar el valor original si no
cumple las condiciones

# Paso 3: Crear una función para imputar valores
def imputar_velocidad(row):
    if pd.isnull(row['fastestLapSpeed']) and
pd.notna(row['min_fastest_lap_speed_by_race']):
        return row['min_fastest_lap_speed_by_race'] -
row['mean_fastest_lap_speed_by_race']
    elif pd.isnull(row['fastestLapSpeed']):
        return min_lap_speed_general + mean_lap_speed_general
    else:
        return row['fastestLapSpeed']  # Dejar el valor original si no
cumple las condiciones

def imputar_q1(row):
    if pd.isnull(row['q1']):
        return max_q1_general + mean_q1_general
    else:
        return row['q1']  # Dejar el valor original si no cumple las
condiciones

def imputar_q2(row):
    if pd.isnull(row['q2']) and pd.notna(row['max_q1_by_race']):
        return row['max_q1_by_race'] + row['mean_q1_by_race']
    elif pd.isnull(row['q2']):
        return max_q1_general + mean_q1_general
    else:
        return row['q2']  # Dejar el valor original si no cumple las
condiciones

def imputar_q3(row):
    if pd.isnull(row['q3']) and pd.notna(row['max_q1_by_race']):
        return row['max_q1_by_race'] + row['mean_q1_by_race']
    elif pd.isnull(row['q3']):
        return max_q1_general + mean_q1_general
    else:
        return row['q3']  # Dejar el valor original si no cumple las
condiciones

# Paso 4: Aplicar la función
prueba4['milliseconds'] = prueba4.apply(imputar_tiempo, axis=1)
prueba4['fastestLapTime']=prueba4.apply(imputar_vuelta_rapida, axis=1)
prueba4['fastestLapSpeed']=prueba4.apply(imputar_velocidad,axis=1)
prueba4['q1']=prueba4.apply(imputar_q1,axis=1)
prueba4['q2']=prueba4.apply(imputar_q2,axis=1)
prueba4['q3']=prueba4.apply(imputar_q3,axis=1)
```

```
# Paso 5: Eliminar columnas intermedias si ya no son necesarias
prueba4.drop(['max_time_by_race',
'mean_time_by_race','max_fastest_lap_by_race','mean_fastest_lap_by_rac
e',

'min_fastest_lap_speed_by_race','mean_fastest_lap_speed_by_race','max_
q1_by_race','mean_q1_by_race','number','rank'], axis=1, inplace=True)

prueba4['mean_pit_stop'] =
prueba4['mean_pit_stop'].fillna(prueba4['mean_pit_stop'].max())
prueba4['position'] = prueba4['position'].fillna(30)
display(prueba4)
```

```
       resultId  raceId  driverId  constructorId  grid  position  \
0             1      18         1              1     1       1.0
1             2      18         2              2     5       2.0
2             3      18         3              3     7       3.0
3             4      18         4              4    11       4.0
4             5      18         5              1     3       5.0
...         ...     ...       ...            ...   ...       ...
26514     26520    1132       839            214    18      16.0
26515     26521    1132       815              9     0      17.0
26516     26522    1132       855             15    14      18.0
26517     26523    1132       847            131     1      30.0
26518     26524    1132       842            214    19      30.0

       positionOrder  points  laps  milliseconds  ...  mean_pit_stop
year  \
0                  1    10.0    58  5.690616e+06  ...     3055732.00
2008
1                  2     8.0    58  5.696094e+06  ...     3055732.00
2008
2                  3     6.0    58  5.698779e+06  ...     3055732.00
2008
3                  4     5.0    58  5.707797e+06  ...     3055732.00
2008
4                  5     4.0    58  5.708630e+06  ...     3055732.00
2008
...              ...     ...   ...           ...  ...            ...
...
26514             16     0.0    50  6.036148e+06  ...       28786.50
2024
26515             17     0.0    50  6.036148e+06  ...       29718.75
2024
26516             18     0.0    50  6.036148e+06  ...       30306.00
2024
26517             19     0.0    33  2.129926e+07  ...       32045.00
2024
26518             20     0.0     0  2.129926e+07  ...     3055732.00
2024
```

```
       milliseconds_missing  fastestLapTime_missing
fastestLapSpeed_missing  \
0                         0                       0
0
1                         0                       0
0
2                         0                       0
0
3                         0                       0
0
4                         0                       0
0
...                     ...                     ...
...
26514                     0                       0
0
26515                     0                       0
0
26516                     0                       0
0
26517                     1                       0
0
26518                     1                       1
1

       q1_missing  q2_missing  q3_missing  mean_pit_stop_missing  \
0               0           0           0                      0
1               0           0           0                      0
2               0           0           0                      0
3               0           0           1                      0
4               0           0           0                      0
...           ...         ...         ...                    ...
26514           0           1           1                      0
26515           0           1           1                      0
26516           0           0           1                      0
26517           0           0           0                      0
26518           0           1           1                      1

       position_missing
0                     0
1                     0
2                     0
3                     0
4                     0
...                 ...
26514                 0
26515                 0
26516                 0
26517                 1
```

```
26518                  1

[26519 rows x 26 columns]
```

```
faltantes = prueba4.isna().sum()

# Mostrar las columnas con valores faltantes
print(faltantes[faltantes > 0])
```

```
Series([], dtype: int64)
```

```
prueba4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26519 entries, 0 to 26518
Data columns (total 26 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   resultId                 26519 non-null  int64  
 1   raceId                   26519 non-null  int64  
 2   driverId                 26519 non-null  int64  
 3   constructorId            26519 non-null  int64  
 4   grid                     26519 non-null  int64  
 5   position                 26519 non-null  float64
 6   positionOrder            26519 non-null  int64  
 7   points                   26519 non-null  float64
 8   laps                     26519 non-null  int64  
 9   milliseconds             26519 non-null  float64
 10  fastestLapTime           26519 non-null  float64
 11  fastestLapSpeed          26519 non-null  float64
 12  statusId                 26519 non-null  int64  
 13  q1                       26519 non-null  float64
 14  q2                       26519 non-null  float64
 15  q3                       26519 non-null  float64
 16  mean_pit_stop            26519 non-null  float64
 17  year                     26519 non-null  int64  
 18  milliseconds_missing     26519 non-null  int64  
 19  fastestLapTime_missing   26519 non-null  int64  
 20  fastestLapSpeed_missing  26519 non-null  int64  
 21  q1_missing               26519 non-null  int64  
 22  q2_missing               26519 non-null  int64  
 23  q3_missing               26519 non-null  int64  
 24  mean_pit_stop_missing    26519 non-null  int64  
 25  position_missing         26519 non-null  int64  
dtypes: float64(9), int64(17)
memory usage: 5.3 MB
```

Estandarizamos nuestro dataframe

```
from sklearn.preprocessing import MinMaxScaler
no_estandarizar=['resultId','raceId','driverId','constructorId','grid'
,'position','positionOrder','points','laps','StatusId','year','millise
conds_missing',

'fastestLapTime_missing','fastestLapSpeed_missing','q1_missing','q2_mi
ssing','q3_missing','mean_pit_stop_missing','position_missing']
si_estandarizar = [col for col in prueba4.columns if col not in
no_estandarizar]
scaler = MinMaxScaler(feature_range=(0,40))
prueba4[si_estandarizar]=
scaler.fit_transform(prueba4[si_estandarizar])
```

Definimos como entrenamiento hasta 2020 y como test desde 2021

```
train=prueba4[prueba4['year']<2021]
test=prueba4[prueba4['year']>=2021]
```

Definimos como épocas de dominancias las tres épocas de dominancia más grandes de la competición

```
condicion1 = (train['year'].between(2014,2020)) &
(train['constructorId'] == 131)
condicion2 = (train['year'].between(2000,2004)) &
(train['constructorId'] == 6)
condicion3 = (train['year'].between(1992,1997)) &
(train['constructorId'] == 3)
condicion= condicion1 | condicion2 | condicion3
train['dominancia']=condicion.astype(int)
```

```
C:\Users\gonza\AppData\Local\Temp\ipykernel_27900\4240786890.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  train['dominancia']=condicion.astype(int)
```

```
prueba4.to_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\4to
curso\MINERIA DE DATOS\TRABAJO FINAL\pruebaMLP.csv",index=False)
```

```
print(prueba4['grid'].isnull().sum())
```

```
0
```

A continuación definimos la red para poder predecir la probabilidad

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split

identificadores=['resultId','driverId','constructorId']
X_train_2020=train.drop(columns=identificadores,axis=1)
X_train_2020=X_train_2020.drop('dominancia',axis=1)
y_train_2020=train['dominancia']
test2=test.drop(columns=identificadores)

# Definir parámetros
batch_size = 64  # Tamaño del batch
epochs = 50  # Número de épocas
learning_rate = 0.001  # Tasa de aprendizaje



X_train, X_val, y_train, y_val = train_test_split(X_train_2020,
y_train_2020, test_size=0.2, random_state=42)

# Definir la arquitectura de la red neuronal (Perceptron Multicapa)
model = Sequential()

# Capa de entrada + primera capa oculta
model.add(Dense(units=64, activation='relu',
input_dim=X_train.shape[1]))

# Capa oculta adicional
model.add(Dense(units=32, activation='relu'))


# Capa de salida
model.add(Dense(units=1, activation='sigmoid'))  # Usamos sigmoid
porque es clasificación binaria

# Compilación del modelo
model.compile(optimizer=Adam(learning_rate=learning_rate),
loss='binary_crossentropy', metrics=['accuracy'])

# Resumen del modelo para verificar la arquitectura
model.summary()

# Entrenamiento del modelo con validación
history = model.fit(X_train, y_train, epochs=epochs,
batch_size=batch_size, validation_data=(X_val, y_val))

# Evaluación del modelo en el conjunto de validación
val_loss, val_accuracy = model.evaluate(X_val, y_val)
print(f"Validation Loss: {val_loss}")
print(f"Validation Accuracy: {val_accuracy}")
```

```python
# Predicción de la probabilidad de que cada equipo esté en una era de dominancia
predictions = model.predict(test2)

# Ahora, agrupamos las predicciones por equipo (asumiendo que 'teamId' es la columna que identifica a los equipos)
test['predicted_prob'] = predictions

# Calculamos la probabilidad media para cada equipo a través de sus carreras en 2021
team_probabilities = test.groupby('constructorId')
['predicted_prob'].mean()

# Crear un diccionario de mapeo desde la tabla 'constructors'
id_to_name = dict(zip(constructors['constructorId'],
constructors['name']))

# Mapear los nombres de los equipos en lugar de sus IDs
team_probabilities_named = {id_to_name[team_id]: prob for team_id,
prob in team_probabilities.items()}

# Mostrar las probabilidades con los nombres de los equipos
for team_name, prob in team_probabilities_named.items():
    print(f"Probabilidad de dominancia para el equipo {team_name} en
toda la época (2021 en adelante): {prob:.4f}")
```

```
c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass
an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first
layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ dense (Dense)                   │ (None, 64)             │
1,536 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 32)             │
```

```
2,080 │
├─────────────────────────────────────┼─────────────────────────────┼─────────
─────┤
│ dense_2 (Dense)                     │ (None, 1)                   │
33 │
└─────────────────────────────────────┴─────────────────────────────┴─────────
─────┘

 Total params: 3,649 (14.25 KB)

 Trainable params: 3,649 (14.25 KB)

 Non-trainable params: 0 (0.00 B)

Epoch 1/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - accuracy: 0.9554 - loss:
1.2112 - val_accuracy: 0.4493 - val_loss: 1.8288
Epoch 2/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9568 - loss:
0.8143 - val_accuracy: 0.9748 - val_loss: 0.0866
Epoch 3/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9603 - loss:
0.2164 - val_accuracy: 0.9778 - val_loss: 0.0773
Epoch 4/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9632 - loss:
0.2482 - val_accuracy: 0.9752 - val_loss: 0.2717
Epoch 5/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9631 - loss:
0.5140 - val_accuracy: 0.8051 - val_loss: 0.7134
Epoch 6/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9587 - loss:
0.4302 - val_accuracy: 0.9700 - val_loss: 0.0926
Epoch 7/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9664 - loss:
0.2107 - val_accuracy: 0.9085 - val_loss: 0.2811
Epoch 8/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9630 - loss:
0.2398 - val_accuracy: 0.9752 - val_loss: 0.3598
Epoch 9/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9709 - loss:
0.1393 - val_accuracy: 0.9625 - val_loss: 0.1238
Epoch 10/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9680 - loss:
0.1931 - val_accuracy: 0.9515 - val_loss: 0.1797
Epoch 11/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9625 - loss:
0.1876 - val_accuracy: 0.8744 - val_loss: 0.5445
Epoch 12/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9634 - loss:
0.2367 - val_accuracy: 0.9752 - val_loss: 0.1259
```

```
Epoch 13/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9702 - loss:
0.1756 - val_accuracy: 0.9333 - val_loss: 0.1880
Epoch 14/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9655 - loss:
0.1891 - val_accuracy: 0.9750 - val_loss: 0.0811
Epoch 15/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9674 - loss:
0.1620 - val_accuracy: 0.9776 - val_loss: 0.0914
Epoch 16/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9680 - loss:
0.1370 - val_accuracy: 0.9780 - val_loss: 0.1813
Epoch 17/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9661 - loss:
0.1890 - val_accuracy: 0.9667 - val_loss: 0.1212
Epoch 18/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9664 - loss:
0.1391 - val_accuracy: 0.9774 - val_loss: 0.0703
Epoch 19/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9657 - loss:
0.1624 - val_accuracy: 0.9768 - val_loss: 0.0730
Epoch 20/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9651 - loss:
0.1596 - val_accuracy: 0.9457 - val_loss: 0.1945
Epoch 21/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9677 - loss:
0.1750 - val_accuracy: 0.9561 - val_loss: 0.1206
Epoch 22/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9686 - loss:
0.1257 - val_accuracy: 0.9567 - val_loss: 0.1178
Epoch 23/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9693 - loss:
0.1190 - val_accuracy: 0.9752 - val_loss: 0.1479
Epoch 24/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9684 - loss:
0.1568 - val_accuracy: 0.9383 - val_loss: 0.1592
Epoch 25/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9657 - loss:
0.2145 - val_accuracy: 0.9641 - val_loss: 0.0906
Epoch 26/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9680 - loss:
0.1188 - val_accuracy: 0.8768 - val_loss: 0.4108
Epoch 27/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9680 - loss:
0.1442 - val_accuracy: 0.9752 - val_loss: 0.1513
Epoch 28/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9684 - loss:
0.1494 - val_accuracy: 0.9756 - val_loss: 0.1165
Epoch 29/50
```

```
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9706 - loss:
0.0942 - val_accuracy: 0.9782 - val_loss: 0.0722
Epoch 30/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9694 - loss:
0.1149 - val_accuracy: 0.9752 - val_loss: 0.2800
Epoch 31/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9721 - loss:
0.1129 - val_accuracy: 0.9615 - val_loss: 0.0959
Epoch 32/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9713 - loss:
0.1011 - val_accuracy: 0.9720 - val_loss: 0.0663
Epoch 33/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9716 - loss:
0.0981 - val_accuracy: 0.9720 - val_loss: 0.0683
Epoch 34/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9718 - loss:
0.1052 - val_accuracy: 0.9629 - val_loss: 0.0994
Epoch 35/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9711 - loss:
0.1057 - val_accuracy: 0.9752 - val_loss: 0.1055
Epoch 36/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9695 - loss:
0.1003 - val_accuracy: 0.9489 - val_loss: 0.1476
Epoch 37/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9729 - loss:
0.1013 - val_accuracy: 0.9784 - val_loss: 0.0529
Epoch 38/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9689 - loss:
0.0920 - val_accuracy: 0.9756 - val_loss: 0.0597
Epoch 39/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9741 - loss:
0.0736 - val_accuracy: 0.9770 - val_loss: 0.0555
Epoch 40/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9689 - loss:
0.1563 - val_accuracy: 0.9764 - val_loss: 0.0620
Epoch 41/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9702 - loss:
0.0954 - val_accuracy: 0.9752 - val_loss: 0.1184
Epoch 42/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9687 - loss:
0.0973 - val_accuracy: 0.9776 - val_loss: 0.0764
Epoch 43/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9704 - loss:
0.1116 - val_accuracy: 0.9794 - val_loss: 0.0516
Epoch 44/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9703 - loss:
0.0866 - val_accuracy: 0.9776 - val_loss: 0.0530
Epoch 45/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9735 - loss:
```

```
0.0764 - val_accuracy: 0.9776 - val_loss: 0.0668
Epoch 46/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9706 - loss:
0.0907 - val_accuracy: 0.9732 - val_loss: 0.0699
Epoch 47/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9704 - loss:
0.0888 - val_accuracy: 0.9780 - val_loss: 0.0572
Epoch 48/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9724 - loss:
0.0737 - val_accuracy: 0.9752 - val_loss: 0.0689
Epoch 49/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9751 - loss:
0.0670 - val_accuracy: 0.9784 - val_loss: 0.0584
Epoch 50/50
312/312 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9740 - loss:
0.0752 - val_accuracy: 0.9780 - val_loss: 0.0576
156/156 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9793 - loss:
0.0557
Validation Loss: 0.05758117884397507
Validation Accuracy: 0.9779647588729858
49/49 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Probabilidad de dominancia para el equipo McLaren en toda la época
(2021 en adelante): 0.3923
Probabilidad de dominancia para el equipo Williams en toda la época
(2021 en adelante): 0.0884
Probabilidad de dominancia para el equipo Ferrari en toda la época
(2021 en adelante): 0.5892
Probabilidad de dominancia para el equipo Red Bull en toda la época
(2021 en adelante): 0.7432
Probabilidad de dominancia para el equipo Sauber en toda la época
(2021 en adelante): 0.1292
Probabilidad de dominancia para el equipo Alfa Romeo en toda la época
(2021 en adelante): 0.0995
Probabilidad de dominancia para el equipo Aston Martin en toda la
época (2021 en adelante): 0.2434
Probabilidad de dominancia para el equipo Mercedes en toda la época
(2021 en adelante): 0.6335
Probabilidad de dominancia para el equipo Haas F1 Team en toda la
época (2021 en adelante): 0.1042
Probabilidad de dominancia para el equipo AlphaTauri en toda la época
(2021 en adelante): 0.1543
Probabilidad de dominancia para el equipo Alpine F1 Team en toda la
época (2021 en adelante): 0.2012
Probabilidad de dominancia para el equipo RB F1 Team en toda la época
(2021 en adelante): 0.2326
Probabilidad de dominancia para el equipo 1 en toda la época (2021 en
adelante): 0.3923
Probabilidad de dominancia para el equipo 3 en toda la época (2021 en
adelante): 0.0884
```

```
Probabilidad de dominancia para el equipo 6 en toda la época (2021 en adelante): 0.5892
Probabilidad de dominancia para el equipo 9 en toda la época (2021 en adelante): 0.7432
Probabilidad de dominancia para el equipo 15 en toda la época (2021 en adelante): 0.1292
Probabilidad de dominancia para el equipo 51 en toda la época (2021 en adelante): 0.0995
Probabilidad de dominancia para el equipo 117 en toda la época (2021 en adelante): 0.2434
Probabilidad de dominancia para el equipo 131 en toda la época (2021 en adelante): 0.6335
Probabilidad de dominancia para el equipo 210 en toda la época (2021 en adelante): 0.1042
Probabilidad de dominancia para el equipo 213 en toda la época (2021 en adelante): 0.1543
Probabilidad de dominancia para el equipo 214 en toda la época (2021 en adelante): 0.2012
Probabilidad de dominancia para el equipo 215 en toda la época (2021 en adelante): 0.2326

C:\Users\gonza\AppData\Local\Temp\ipykernel_27900\1361340852.py:54: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test['predicted_prob'] = predictions
```

ANEXO 7

En este fichero nos descargaremos a través de Web Scrapping las imágenes de los trazados de los circuitos y generaremos nuevas imágenes a través Data Augmentation

```
import pandas as pd

circuits=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\BASE DE DATOS\
circuits.csv",na_values="\\N")
```

Definimos un dataframe con los circuitos clasificados en nuestras categorías de interés

```
import pandas as pd



# Datos de la tabla corregidos con 76 elementos en cada lista
datos = {
    'Circuito': [
        'Adelaide Street Circuit', 'Ain Diab', 'Aintree', 'Albert Park
Grand Prix Circuit',
        'Autódromo do Estoril', 'Autodromo Enzo e Dino Ferrari',
'Autódromo Hermanos Rodríguez',
        'Autódromo Internacional do Algarve', 'Autódromo Internacional
Nelson Piquet',
        'Autodromo Internazionale del Mugello', 'Autódromo José Carlos
Pace', 'Autódromo Juan y Oscar Gálvez',
        'Autodromo Nazionale di Monza', 'AVUS', 'Bahrain International
Circuit', 'Baku City Circuit',
        'Brands Hatch', 'Buddh International Circuit', 'Charade
Circuit', 'Circuit Bremgarten',
        'Circuit de Barcelona-Catalunya', 'Circuit de Monaco',
'Circuit de Nevers Magny-Cours',
        'Circuit de Pedralbes', 'Circuit de Spa-Francorchamps',
'Circuit Gilles Villeneuve',
        'Circuit Mont-Tremblant', 'Circuit of the Americas', 'Circuit
Park Zandvoort', 'Circuit Paul Ricard',
        'Circuito da Boavista', 'Circuito de Jerez', 'Detroit Street
Circuit', 'Dijon-Prenois', 'Donington Park',
        'Fair Park', 'Fuji Speedway', 'Hockenheimring', 'Hungaroring',
'Indianapolis Motor Speedway',
        'Istanbul Park', 'Jarama', 'Jeddah Corniche Circuit', 'Korean
International Circuit', 'Kyalami',
        'Las Vegas Strip Street Circuit', 'Le Mans', 'Long Beach',
'Losail International Circuit',
        'Marina Bay Street Circuit', 'Miami International Autodrome',
'Monsanto Park Circuit', 'Montjuïc',
        'Mosport International Raceway', 'Nivelles-Baulers',
'Nürburgring', 'Okayama International Circuit',
```

```
        'Pescara Circuit', 'Phoenix street circuit', 'Prince George
Circuit', 'Red Bull Ring', 'Reims-Gueux',
        'Riverside International Raceway', 'Rouen-Les-Essarts',
'Scandinavian Raceway', 'Sebring International Raceway',
        'Sepang International Circuit', 'Shanghai International
Circuit', 'Silverstone Circuit', 'Sochi Autodrom',
        'Suzuka Circuit', 'Valencia Street Circuit', 'Watkins Glen',
'Yas Marina Circuit', 'Zeltweg', 'Zolder'
    ],
    'Velocidad': [
        'Lento', 'Rápido', 'Rápido', 'Rápido', 'Rápido', 'Rápido',
'Rápido', 'Rápido', 'Lento', 'Rápido',
        'Rápido', 'Rápido', 'Rápido', 'Rápido', 'Rápido', 'Rápido',
'Rápido', 'Rápido', 'Lento', 'Rápido',
        'Rápido', 'Lento', 'Lento', 'Rápido', 'Rápido', 'Rápido',
'Rápido', 'Rápido', 'Lento', 'Rápido',
        'Lento', 'Rápido', 'Lento', 'Rápido', 'Rápido', 'Lento',
'Rápido', 'Rápido', 'Lento', 'Rápido',
        'Rápido', 'Lento', 'Rápido', 'Rápido', 'Rápido', 'Rápido',
'Rápido', 'Lento', 'Rápido', 'Lento',
        'Rápido', 'Rápido', 'Lento', 'Lento', 'Rápido', 'Lento',
'Rápido', 'Rápido', 'Rápido', 'Rápido',
        'Rápido', 'Rápido', 'Rápido', 'Rápido', 'Rápido', 'Rápido',
'Rápido', 'Rápido', 'Rápido', 'Rápido',
        'Rápido', 'Lento', 'Rápido', 'Rápido','Rápido','Rápido'
    ],
    'Seguridad': [
        'Seguro', 'Inseguro', 'Inseguro', 'Seguro', 'Seguro',
'Inseguro', 'Seguro', 'Seguro', 'Inseguro',
        'Inseguro', 'Seguro', 'Seguro', 'Seguro', 'Inseguro',
'Seguro', 'Inseguro', 'Inseguro', 'Seguro',
        'Inseguro', 'Inseguro', 'Seguro', 'Inseguro', 'Seguro',
'Inseguro', 'Inseguro', 'Seguro', 'Inseguro',
        'Seguro', 'Seguro', 'Seguro', 'Inseguro', 'Seguro',
'Inseguro', 'Inseguro', 'Seguro', 'Inseguro',
        'Seguro', 'Seguro', 'Seguro', 'Seguro', 'Seguro', 'Seguro',
'Inseguro', 'Seguro', 'Seguro', 'Inseguro',
        'Inseguro', 'Seguro', 'Inseguro', 'Seguro', 'Seguro',
'Seguro', 'Inseguro', 'Inseguro', 'Inseguro',
        'Seguro', 'Inseguro', 'Seguro', 'Inseguro', 'Inseguro',
'Seguro', 'Seguro', 'Inseguro', 'Inseguro',
        'Inseguro', 'Inseguro', 'Seguro', 'Seguro', 'Seguro',
'Seguro', 'Inseguro', 'Seguro', 'Seguro', 'Seguro',
        'Inseguro','Seguro'
    ],
    'Adelantamientos': [
        'Pocos', 'Pocos', 'Muchos', 'Muchos', 'Pocos', 'Pocos',
'Muchos', 'Muchos', 'Pocos', 'Pocos',
        'Muchos', 'Pocos', 'Muchos', 'Pocos', 'Muchos', 'Muchos',
```

```
'Pocos', 'Muchos', 'Pocos', 'Pocos',
        'Pocos', 'Pocos', 'Pocos', 'Pocos', 'Muchos', 'Muchos',
'Pocos', 'Muchos', 'Pocos', 'Muchos',
        'Pocos', 'Pocos', 'Pocos', 'Muchos', 'Muchos', 'Pocos',
'Muchos', 'Muchos', 'Pocos', 'Muchos',
        'Muchos', 'Pocos', 'Muchos', 'Pocos', 'Pocos', 'Muchos',
'Pocos', 'Pocos', 'Muchos', 'Pocos',
        'Muchos', 'Muchos', 'Pocos', 'Pocos', 'Pocos', 'Pocos',
'Muchos', 'Pocos', 'Pocos', 'Pocos',
        'Pocos', 'Muchos', 'Pocos', 'Pocos', 'Pocos', 'Pocos',
'Pocos', 'Muchos', 'Muchos', 'Muchos',
        'Muchos', 'Pocos', 'Muchos', 'Pocos', 'Pocos', 'Pocos'
    ]
}

# Asegurarse de que todas las listas tienen 76 elementos
print(f"Circuito: {len(datos['Circuito'])} elementos")
print(f"Velocidad: {len(datos['Velocidad'])} elementos")
print(f"Seguridad: {len(datos['Seguridad'])} elementos")
print(f"Adelantamientos: {len(datos['Adelantamientos'])} elementos")

# Crear un DataFrame
df = pd.DataFrame(datos)


# Crear un DataFrame
df = pd.DataFrame(datos)

# Guardar el DataFrame en un archivo CSV
df.to_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\DATOS EXTRAIDOS\
circuitos_clasificados.csv", index=False)


Circuito: 76 elementos
Velocidad: 76 elementos
Seguridad: 76 elementos
Adelantamientos: 76 elementos

display(df)
```

	Circuito	Velocidad	Seguridad	Adelantamientos
0	Adelaide Street Circuit	Lento	Seguro	Pocos
1	Ain Diab	Rápido	Inseguro	Pocos
2	Aintree	Rápido	Inseguro	Muchos
3	Albert Park Grand Prix Circuit	Rápido	Seguro	Muchos
4	Autódromo do Estoril	Rápido	Seguro	Pocos
..	...	...	...	...
71	Valencia Street Circuit	Lento	Seguro	Pocos
72	Watkins Glen	Rápido	Seguro	Muchos
73	Yas Marina Circuit	Rápido	Seguro	Pocos

```
74                          Zeltweg    Rápido  Inseguro           Pocos
75                           Zolder    Rápido    Seguro           Pocos

[76 rows x 4 columns]
```

Definimos la URL de la que extraeremos las imágenes de los ciruitos

```
import requests
from bs4 import BeautifulSoup
from rapidfuzz import fuzz, process  # Para coincidencias parciales
import pandas as pd

# Nombres de circuitos en tu base de datos
circuitos_db = circuits['name']
# URL de la página web
url = "https://51gt3.com/en/track"
```

Extraemos los trazados de todos los circuitos

```
import os

# Realizar la solicitud
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

# Crear una carpeta para guardar las imágenes si no existe
carpeta_imagenes = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS"


# Extraer nombres y descargar imágenes
circuitos = soup.find_all('div', class_='position-relative')
for circuito in circuitos:
    # Extraer la URL de la imagen
    img_tag = circuito.find('img', class_='card-img')
    if img_tag and 'data-src' in img_tag.attrs:
        img_url = img_tag['data-src']
        # Asegurar que la URL sea completa
        img_url = img_url if img_url.startswith("http") else f"https:
{img_url}"

        # Extraer el nombre del circuito
        sibling_div = circuito.find_next_sibling('div', class_='card-
body')
        if sibling_div:
            nombre_tag = sibling_div.find('h4', class_='card-title')
            if nombre_tag:
                nombre_circuito = nombre_tag.text.strip()
```

```
                # Formatear el nombre del archivo para evitar
caracteres no válidos
                nombre_archivo = f"{nombre_circuito}.png".replace(" ",
"_").replace("/", "_")
                ruta_completa = os.path.join(carpeta_imagenes,
nombre_archivo)

                # Descargar y guardar la imagen
                try:
                    print(f"Descargando {nombre_circuito} desde
{img_url}...")
                    img_data = requests.get(img_url).content
                    with open(ruta_completa, 'wb') as f:
                        f.write(img_data)
                    print(f"Guardado en {ruta_completa}")
                except Exception as e:
                    print(f"Error al descargar {nombre_circuito}:
{e}")
            else:
                print("Nombre del circuito no encontrado.")
    else:
        print("URL de la imagen no encontrada.")
```

```
URL de la imagen no encontrada.
URL de la imagen no encontrada.
Descargando Acura Grand Prix of Long Beach desde
https://img2.51gt3.com/rac/track/202305/4234608427f245d09bc52617f73235
3f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Acura_Grand_Prix_of_Long_Beach.png
Descargando Adelaide Street Circuit desde
https://img2.51gt3.com/rac/track/202309/d88889a9bb5e4b4fa2d08fab37cec7
2d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Adelaide_Street_Circuit.png
Descargando Albert Park Circuit desde
https://img2.51gt3.com/rac/track/202303/1de549cd55884f8dabb6c54b222360
46.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Albert_Park_Circuit.png
Descargando Algarve International Circuit desde
https://img2.51gt3.com/rac/track/202303/b5a6740660e041febc33c74497a1f4
98.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Algarve_International_Circuit.png
Descargando Anneau du Rhin desde
```

```
https://img2.51gt3.com/rac/track/202309/e344ca2efe8840a59fa8277b1caa5d59.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Anneau_du_Rhin.png
Descargando Autódromo Víctor Borrat Fabini desde https://img2.51gt3.com/rac/track/202412/4cc53304296b4d5f982714502c7f27ed.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autódromo_Víctor_Borrat_Fabini.png
Descargando Autódromo de Most desde https://img2.51gt3.com/rac/track/202305/9a4fbf836d3a462a9aedc23d89a4789a.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autódromo_de_Most.png
Descargando Autodromo di Franciacorta desde https://img2.51gt3.com/rac/track/202309/d363081e31224122b333803b0ad0af31.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autodromo_di_Franciacorta.png
Descargando Autodromo di Pergusa desde https://img2.51gt3.com/rac/track/202309/96fd09870fe24eb8bb06af97715108af.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autodromo_di_Pergusa.png
Descargando Autódromo Hermanos Rodríguez desde https://img2.51gt3.com/rac/track/202303/c9098eab7c3d4ce7b4bdaf1dbd9845af.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autódromo_Hermanos_Rodríguez.png
Descargando Autódromo Miguel E. Abed desde https://img2.51gt3.com/rac/track/202309/b67b611b77de4efcad4e3da730a3228d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autódromo_Miguel_E._Abed.png
Descargando Autodromo Nazionale Monza desde https://img2.51gt3.com/rac/track/202304/73988af861d14f0bb3b39149aefaff65.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autodromo_Nazionale_Monza.png
Descargando Vallelunga Circuit desde https://img2.51gt3.com/rac/track/202305/6da81629ffa041869e8ed69cb9d7f943.png?x-oss-process=style/_nowm...
```

```
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Vallelunga_Circuit.png
Descargando Automotodróm Slovakia Ring desde
https://img2.51gt3.com/rac/track/202305/e28b0055713442f9975e70e996734d
76.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Automotodróm_Slovakia_Ring.png
Descargando Autopolis Circuit desde
https://img2.51gt3.com/rac/track/202305/cd0f4a74f0de4cf990e2aa13cfd214
09.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Autopolis_Circuit.png
Descargando Bahrain International Circuit desde
https://img2.51gt3.com/rac/track/202303/d3984f0b93e24899af9cae74727db3
cf.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Bahrain_International_Circuit.png
Descargando Bahrain Endurance Circuit desde
https://img2.51gt3.com/rac/track/202303/6a9b5e0e0bc543c58e036cb27f4a1f
bd.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Bahrain_Endurance_Circuit.png
Descargando Bahrain Oasis Circuit desde
https://img2.51gt3.com/rac/track/202303/29d1de97c98d43098d150787eaf07b
92.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Bahrain_Oasis_Circuit.png
Descargando Bahrain Outer Circuit desde
https://img2.51gt3.com/rac/track/202303/85b6198e27e34629a65d5f800867ee
9f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Bahrain_Outer_Circuit.png
Descargando Bahrain Paddock Circuit desde
https://img2.51gt3.com/rac/track/202303/5bc3e46fb9504f798a7f2a2a8e97d1
74.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Bahrain_Paddock_Circuit.png
Descargando Baku City Circuit desde
https://img2.51gt3.com/rac/track/202303/e92a3676600d4c718a994a06ae7d8c
4d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
```

```
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Baku_City_Circuit.png
Descargando Bangsaen Street Circuit desde
https://img2.51gt3.com/rac/track/202304/af1f39bc6cfb476ea5a9c25af5ef94
17.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Bangsaen_Street_Circuit.png
Descargando Barber Motorsports Park desde
https://img2.51gt3.com/rac/track/202305/d3026a11444c4298af2bc736397f4c
12.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Barber_Motorsports_Park.png
Descargando Beijing Rui Si Race Track desde
https://img2.51gt3.com/rac/track/202411/b675534e25e544a5939c72602abcc4
de.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Beijing_Rui_Si_Race_Track.png
Descargando Bira International Circuit desde
https://img2.51gt3.com/rac/track/202407/365d234e40b54b0b9eeaeb42d7ee82
19.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Bira_International_Circuit.png
Descargando Brainerd International Raceway desde
https://img2.51gt3.com/rac/track/202309/fbb4879140cf4cc5bc14e7523ac8c1
f1.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Brainerd_International_Raceway.png
Descargando Brands Hatch Circuit desde
https://img2.51gt3.com/rac/track/202305/bbe31d62874c48049593a5cd290919
7a.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Brands_Hatch_Circuit.png
Descargando Brands Hatch - Indy Circuit desde
https://img2.51gt3.com/rac/track/202305/6c3d39238e2247ea9913cd443c15fc
d6.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Brands_Hatch_-
_Indy_Circuit.png
Descargando Brno Circuit desde
https://img2.51gt3.com/rac/track/202305/2b68c7bb66d549dd928933e9f7daf1
5f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Brno_Circuit.png
```

```
Descargando Buddh International Circuit desde
https://img2.51gt3.com/rac/track/202304/0c55cf37dcb9452092bc316dde5ac0
0d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Buddh_International_Circuit.png
Descargando Bugatti Circuit desde
https://img2.51gt3.com/rac/track/202305/2f9dca5144c740479f85e665559c2f
3b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Bugatti_Circuit.png
Descargando Chang International Circuit desde
https://img2.51gt3.com/rac/track/202304/115ed1b0993a490d9a2125210ecb1f
2a.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Chang_International_Circuit.png
Descargando Chengdu Tianfu International Circuit desde
https://img2.51gt3.com/rac/track/202312/8a0254ba123649a788c91c3e1fbd16
4f.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Chengdu_Tianfu_International_Circuit.png
Descargando Circuit de Barcelona-Catalunya desde
https://img2.51gt3.com/rac/track/202303/35ad041fd64f44628adaec94b07696
07.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_de_Barcelona-Catalunya.png
Descargando Circuit de Barcelona-Catalunya National Circuit desde
https://img2.51gt3.com/rac/track/202305/c4c98819dc4b4b2a8dad804363bc19
c6.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_de_Barcelona-Catalunya_National_Circuit.png
Descargando Le Mans 24 Hours Circuit de la Sarthe desde
https://img2.51gt3.com/rac/track/202305/1c82606f59f340b9af39f9174029ea
7d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Le_Mans_24_Hours_Circuit_de_la_Sarthe.png
Descargando Circuit de Lédenon desde
https://img2.51gt3.com/rac/track/202305/aea51f8370eb4bd6856c29a20e626e
a9.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_de_Lédenon.png
Descargando Circuit de Monaco desde
https://img2.51gt3.com/rac/track/202304/adbd43e013004af186a50503a1c2b2
```

```
60.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_de_Monaco.png
Descargando Circuit de Nevers Magny-Cours desde
https://img2.51gt3.com/rac/track/202304/f7d4ee33c3b5486285d1af2bae6a06
81.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_de_Nevers_Magny-Cours.png
Descargando Circuit International Automobile Moulay El Hassan desde
https://img2.51gt3.com/rac/track/202305/fa01069a289f42578ee990f049bafe
09.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_International_Automobile_Moulay_El_Hassan.png
Descargando Circuit of the Americas desde
https://img2.51gt3.com/rac/track/202303/d093da62dab34f54b494979cce5a7a
1c.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_of_the_Americas.png
Descargando Circuit Pau-Arnos desde
https://img2.51gt3.com/rac/track/202309/3e0de400628646da81b5323dc1f840
2d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Circuit_Pau-
Arnos.png
Descargando Circuit Paul Armagnac desde
https://img2.51gt3.com/rac/track/202305/859cabafe52644009c8cc1f61d561e
5f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_Paul_Armagnac.png
Descargando Paul Ricard Circuit desde
https://img2.51gt3.com/rac/track/202304/b16da65815684d12aea6b42f423658
82.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Paul_Ricard_Circuit.png
Descargando Circuit Ricardo Tormo desde
https://img2.51gt3.com/rac/track/202304/e96ba2e3abbc4183b11627ecde2bf3
51.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_Ricardo_Tormo.png
Descargando Circuit Zandvoort desde
https://img2.51gt3.com/rac/track/202304/f7d718f5f16f49038f69f21a3f3d97
2f.png?x-oss-process=style/_nowm...
```

```
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuit_Zandvoort.png
Descargando Circuit Zolder desde
https://img2.51gt3.com/rac/track/202305/ad7f0a9354834df8a4898d1eb7f549
d0.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Circuit_Zolder.png
Descargando Circuito de Jerez desde
https://img2.51gt3.com/rac/track/202309/af983d581b0c4450b755f697694f4d
1c.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuito_de_Jerez.png
Descargando Circuito de Navarra desde
https://img2.51gt3.com/rac/track/202305/23311b9193c54cf3a8a5ebeb317273
a9.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuito_de_Navarra.png
Descargando Circuito del Jarama desde
https://img2.51gt3.com/rac/track/202305/d4def35d563945af92e217bdd9da45
3c.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuito_del_Jarama.png
Descargando Circuito do Estoril desde
https://img2.51gt3.com/rac/track/202304/ef32715177c2493390f9b468a5863e
77.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Circuito_do_Estoril.png
Descargando Croft Circuit desde
https://img2.51gt3.com/rac/track/202309/dc4100e81b2b4b44b24fbf32bcc9cf
1d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Croft_Circuit.png
Descargando DEKRA Lausitzring desde
https://img2.51gt3.com/rac/track/202305/f6f1a382d81e42c8898e907f1dd4d4
d0.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
DEKRA_Lausitzring.png
Descargando DERKA Lausitzring-T1 Oval desde
https://img2.51gt3.com/rac/track/202309/132ef9b08e99408ea952f5ac43e54d
61.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\DERKA_Lausitzring-
T1_Oval.png
```

```
Descargando Detroit Grand Prix (IndyCar) desde
https://img2.51gt3.com/rac/track/202305/cfc2c05b54224b61be5dda1b29140a
54.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Detroit_Grand_Prix_(IndyCar).png
Descargando Dijon-Prenois Circuit desde
https://img2.51gt3.com/rac/track/202305/ac2bd3efaba0438ea3edd19de25ca2
50.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Dijon-
Prenois_Circuit.png
Descargando Donington Park desde
https://img2.51gt3.com/rac/track/202305/04ed487923dc4373bdab93c252584a
7b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Donington_Park.png
Descargando Dubai Autodrome-Club Circuit desde
https://img2.51gt3.com/rac/track/202309/e8c8ecd6f7784499ac06d2ab792d91
36.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Dubai_Autodrome-
Club_Circuit.png
Descargando Dubai Autodrome-Grand Prix Course desde
https://img2.51gt3.com/rac/track/202303/a9f182d0923a464aa2661fcb557c73
88.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Dubai_Autodrome-
Grand_Prix_Course.png
Descargando Dubai Autodrome-International Course desde
https://img2.51gt3.com/rac/track/202303/57d2dccf938c49ecbfd33f639c4d9b
e1.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Dubai_Autodrome-
International_Course.png
Descargando Dubai Autodrome-National Circuit desde
https://img2.51gt3.com/rac/track/202309/ad384badda00402dac51c63e6e2329
e1.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Dubai_Autodrome-
National_Circuit.png
Descargando Everland Speedway desde
https://img2.51gt3.com/rac/track/202309/621f4257485f403eafb2c5767370fb
03.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Everland_Speedway.png
Descargando Fuji International Speedway Circuit desde
https://img2.51gt3.com/rac/track/202304/c1e5028ba5cc4f1abd76de8f32a601
```

```
24.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Fuji_International_Speedway_Circuit.png
Descargando Beijing Goldenport Park Circuit desde
https://img2.51gt3.com/rac/track/202304/de124ea75410407489b56b27a9d7be
0b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Beijing_Goldenport_Park_Circuit.png
Descargando Grand Prix of St. Petersburg desde
https://img2.51gt3.com/rac/track/202305/17d7678d3bc242d2b6796c4d49f04c
4b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Grand_Prix_of_St._Petersburg.png
Descargando Guangdong International Circuit desde
https://img2.51gt3.com/rac/track/202304/dd2b7a7a28e44212a100fd03dde5e9
93.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Guangdong_International_Circuit.png
Descargando Guizhou Junchi International Circuit desde
https://img2.51gt3.com/rac/track/202304/de85de8773a640acbde1e2c848b1b2
8f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Guizhou_Junchi_International_Circuit.png
Descargando Hockenheimring desde
https://img2.51gt3.com/rac/track/202304/9e9b949390ca443bb291a78a85da90
40.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Hockenheimring.png
Descargando Homestead–Miami Speedway desde
https://img2.51gt3.com/rac/track/202309/ab8103528ec1427bb8bfc9c57ba065
28.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Homestead–
Miami_Speedway.png
Descargando Hungaroring desde
https://img2.51gt3.com/rac/track/202309/f24f80e559c54c12ba9a7bd87e2881
0b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Hungaroring.png
Descargando Hyderabad Street Circuit desde
https://img2.51gt3.com/rac/track/202309/cd6404b7b5724ba6a46a9210ee3487
1b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
```

```
Hyderabad_Street_Circuit.png
Descargando Igora Drive Autodrome desde
https://img2.51gt3.com/rac/track/202304/dc51184da22c4630a9a25913313d8b
3c.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Igora_Drive_Autodrome.png
Descargando Imola Circuit desde
https://img2.51gt3.com/rac/track/202304/15ab044da2b542b587a5ddba4a9ce7
6e.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Imola_Circuit.png
Descargando Inje Speedium desde
https://img2.51gt3.com/rac/track/202305/bf8a71afcd3a4171ad79d303564f0f
f0.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Inje_Speedium.png
Descargando Interlagos Circuit desde
https://img2.51gt3.com/rac/track/202304/c0bcb433231b430d8586c98f252ee4
fd.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Interlagos_Circuit.png
Descargando Istanbul Park desde
https://img2.51gt3.com/rac/track/202304/c0335937e96c410483bc73ecae1168
03.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Istanbul_Park.png
Descargando Jeddah Corniche Circuit desde
https://img2.51gt3.com/rac/track/202304/a791717c486b4cd48642b066d35a11
0c.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Jeddah_Corniche_Circuit.png
Descargando Jiangsu Wanchi International Circuit desde
https://img2.51gt3.com/rac/track/202107/736c5ae8901d424b9778575b830922
f0.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Jiangsu_Wanchi_International_Circuit.png
Descargando Knockhill Racing Circuit desde
https://img2.51gt3.com/rac/track/202309/4c64f3b7a23c497ba1ac09d1cfdfd2
2a.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Knockhill_Racing_Circuit.png
Descargando Korea International Circuit desde
https://img2.51gt3.com/rac/track/202304/6d7dd277558048598f4e41bbb43f9b
f5.png?x-oss-process=style/_nowm...
```

```
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Korea_International_Circuit.png
Descargando Kuwait Motor Town desde
https://img2.51gt3.com/rac/track/202304/fea5a5d57415471896fba20fc6d44a
5b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Kuwait_Motor_Town.png
Descargando Kyalami Grand Prix Circuit desde
https://img2.51gt3.com/rac/track/202305/1a6fd3813dbb421bbb0aee79cac6d4
d8.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Kyalami_Grand_Prix_Circuit.png
Descargando Las Vegas Strip Street Circuit desde
https://img2.51gt3.com/rac/track/202408/d5a00fba79924029b03c33fde14074
c8.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Las_Vegas_Strip_Street_Circuit.png
Descargando Lihpao International Circuit desde
https://img2.51gt3.com/rac/track/202305/37ee7bf7727d409ba0a941543ce684
d6.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Lihpao_International_Circuit.png
Descargando Lime Rock Park desde
https://img2.51gt3.com/rac/track/202305/169101084f074b43b8f5798489c7c8
53.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Lime_Rock_Park.png
Descargando Lusail International Circuit desde
https://img2.51gt3.com/rac/track/202304/09453dfd832a4c4288021df1595cae
63.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Lusail_International_Circuit.png
Descargando Macau Guia Circuit desde
https://img2.51gt3.com/rac/track/549e0a0f3c8149abb316e2084d43b2bb.jpg?
x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Macau_Guia_Circuit.png
Descargando Pertamina Mandalika International Street Circuit desde
https://img2.51gt3.com/rac/track/202408/a10acebbe3c74f91afae81c07d0f16
a3.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Pertamina_Mandalika_International_Street_Circuit.png
```

```
Descargando Singapore Marina Bay Street Circuit desde
https://img2.51gt3.com/rac/track/202304/c7997eea68a34e63bf5457a6528554
f0.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Singapore_Marina_Bay_Street_Circuit.png
Descargando Miami International Autodrome desde
https://img2.51gt3.com/rac/track/202309/c7342d1e729f4e93a47fc2df2781a4
b0.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Miami_International_Autodrome.png
Descargando Michelin Raceway Road Atlanta desde
https://img2.51gt3.com/rac/track/202305/ec34763911e74a6e9f161b490b30a5
4f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Michelin_Raceway_Road_Atlanta.png
Descargando Mid-Ohio Sports Car Course desde
https://img2.51gt3.com/rac/track/202305/27d0f0a3d48543e8bf5705410583d8
3f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Mid-
Ohio_Sports_Car_Course.png
Descargando Misano World Circuit desde
https://img2.51gt3.com/rac/track/202309/fe1b0789c5444c63907024a8da445a
1e.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Misano_World_Circuit.png
Descargando Mobility Resort Motegi desde
https://img2.51gt3.com/rac/track/202305/5a8c23a4f77948dab5845398d7ad83
15.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Mobility_Resort_Motegi.png
Descargando Moscow Raceway desde
https://img2.51gt3.com/rac/track/202304/e3160761cfb74e8ab23ee4ac1bba8a
01.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Moscow_Raceway.png
Descargando MotorLand Aragón desde
https://img2.51gt3.com/rac/track/202304/2ae7b0d0195a46be8d45d96fefc413
ca.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\MotorLand_Aragón.png
Descargando Motorland Aragón-National Circuit desde
https://img2.51gt3.com/rac/track/202309/a413777931e048d5a82d4efb2f08cb
a2.png?x-oss-process=style/_nowm...
```

```
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Motorland_Aragón-
National_Circuit.png
Descargando Motorsport Arena Oschersleben desde
https://img2.51gt3.com/rac/track/202305/57989e4b32904fe193132ad243606f
45.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Motorsport_Arena_Oschersleben.png
Descargando Mount Panorama Circuit desde
https://img2.51gt3.com/rac/track/202403/a068e9fe89f1471594711b1d624190
a8.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Mount_Panorama_Circuit.png
Descargando Mugello Circuit desde
https://img2.51gt3.com/rac/track/202304/1d4fc49b83044f5ebbfde0b2e21983
6f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Mugello_Circuit.png
Descargando Nashville Street Circuit desde
https://img2.51gt3.com/rac/track/202305/ccb94c038dc74f649825edd3b25dd3
e4.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Nashville_Street_Circuit.png
Descargando New Jersey Motorsports Park desde
https://img2.51gt3.com/rac/track/202309/4b1a0b6482c4415c856ee9fd299357
2f.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
New_Jersey_Motorsports_Park.png
Descargando Newcastle Street Circuit desde
https://img2.51gt3.com/rac/track/202309/8592464b62144d1793310b7bc9d8d9
94.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Newcastle_Street_Circuit.png
Descargando Ningbo International Circuit desde
https://img2.51gt3.com/rac/track/bfc5e282554d4ab7a1144dfaa294cb2e.jpg?
x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Ningbo_International_Circuit.png
Descargando Norisring desde
https://img2.51gt3.com/rac/track/202305/252050961c6144969e9ed1edb6ebd4
f3.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Norisring.png
```

```
Descargando Nürburgring Grand Prix Circuit desde
https://img2.51gt3.com/rac/track/202304/2478955935b2421b9bc575c3f641123d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Nürburgring_Grand_Prix_Circuit.png
Descargando Okayama International Circuit desde
https://img2.51gt3.com/rac/track/202304/82ab56903cc148c8902bc60272f9d919.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Okayama_International_Circuit.png
Descargando Autodromo José Carlos Bassi desde
https://img2.51gt3.com/rac/track/202411/43e8157355f14443a6813b548a43c163.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Autodromo_José_Carlos_Bassi.png
Descargando Phillip Island Grand Prix Circuit desde
https://img2.51gt3.com/rac/track/202408/adf0bf8c9ef1493cafd8018779675ba8.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Phillip_Island_Grand_Prix_Circuit.png
Descargando Pingtan Street Circuit desde
https://img2.51gt3.com/rac/track/202304/2af19dbd9c0d489f9fee5f58df453428.jpeg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Pingtan_Street_Circuit.png
Descargando Pingtan Street Circuit 2.937 desde
https://img2.51gt3.com/rac/track/202309/c73ad8c932f3410589a98363afd5d730.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Pingtan_Street_Circuit_2.937.png
Descargando Portland International Raceway desde
https://img2.51gt3.com/rac/track/202305/d73f5406998c4a688d3872644cbe708b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Portland_International_Raceway.png
Descargando Qinhuangdao Shougang Motorsport Valley desde
https://img2.51gt3.com/rac/track/1861243e10b046b0af7e3e57391845ab.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Qinhuangdao_Shougang_Motorsport_Valley.png
Descargando Queensland Raceway desde
```

```
https://img2.51gt3.com/rac/track/202408/4ad2511d0e8f4e42bbbc3c52766b35c3.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Queensland_Raceway.png
Descargando Red Bull Ring desde https://img2.51gt3.com/rac/track/202304/10482227212b4ac3a557ce0197cb87a0.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Red_Bull_Ring.png
Descargando Road America desde https://img2.51gt3.com/rac/track/202305/344d4a5199fa4abeb203c03512dab91e.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Road_America.png
Descargando Sachsenring desde https://img2.51gt3.com/rac/track/202305/dafef558d6d241ef81c6b2670573d485.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Sachsenring.png
Descargando Sepang International Circuit desde https://img2.51gt3.com/rac/track/ccd74fdadcbb48e0a5a65831e0960c3c.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Sepang_International_Circuit.png
Descargando Shandong Weifang International Circuit desde https://img2.51gt3.com/rac/track/202302/765c4034959144d6bcb15d97513f1ae9.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Shandong_Weifang_International_Circuit.png
Descargando Shanghai International Circuit desde https://img2.51gt3.com/rac/track/8a961b9f3cc24e33965cfb9fea0194fa.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Shanghai_International_Circuit.png
Descargando Shanghai Tianma Circuit desde https://img2.51gt3.com/rac/track/9064c8b43b724b448c103cb03f0ff47f.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Shanghai_Tianma_Circuit.png
Descargando Silverstone Circuit desde https://img2.51gt3.com/rac/track/202304/fed0c74be75347a490b23f65a87c1d0e.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
```

```
Silverstone_Circuit.png
Descargando Snetterton Circuit desde
https://img2.51gt3.com/rac/track/202305/e3373dd331b44ebc83810b9439aec161.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Snetterton_Circuit.png
Descargando Sochi Autodrom desde
https://img2.51gt3.com/rac/track/202304/1d2fa625fc8042709dc6d4ac16a2d35b.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Sochi_Autodrom.png
Descargando Sonoma Raceway desde
https://img2.51gt3.com/rac/track/202309/18b763770c9d4e8db692a588bd82f02c.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Sonoma_Raceway.png
Descargando Circuit de Spa-Francorchamps desde
https://img2.51gt3.com/rac/track/202304/1aebcbf68ab14bce81924c06009fbe62.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Circuit_de_Spa-Francorchamps.png
Descargando Sportsland Sugo desde
https://img2.51gt3.com/rac/track/202305/78aaed5deb0f4ab0a2206878554f7214.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Sportsland_Sugo.png
Descargando Surfers Paradise Street Circuit desde
https://img2.51gt3.com/rac/track/202305/0092007418c74848a955a905af307a0e.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Surfers_Paradise_Street_Circuit.png
Descargando Suzuka Circuit desde
https://img2.51gt3.com/rac/track/aacbce6c41dd4e5496eea246fc5e7c6b.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Suzuka_Circuit.png
Descargando Sydney Motorsport Park desde
https://img2.51gt3.com/rac/track/202305/b82bfceeec854c2283df8d50bf481f4e.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Sydney_Motorsport_Park.png
Descargando Symmons Plains Raceway desde
https://img2.51gt3.com/rac/track/202408/965b4e6a49f54790afc4e6f6b8a4cd1f.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
```

```
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Symmons_Plains_Raceway.png
Descargando Taupo International Motorsport Park desde
https://img2.51gt3.com/rac/track/202305/3036768ee1674801a296d827b7463a
fe.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Taupo_International_Motorsport_Park.png
Descargando The Bend Motorsport Park desde
https://img2.51gt3.com/rac/track/202305/95600b8292bb45d9970b252bfa2ad3
fd.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
The_Bend_Motorsport_Park.png
Descargando The Bend Motorsport Park - International Circuit desde
https://img2.51gt3.com/rac/track/202309/c56fb009f44e46c48c7bd7f721fba8
4a.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
The_Bend_Motorsport_Park_-_International_Circuit.png
Descargando Ordos International Circuit desde
https://img2.51gt3.com/rac/track/1d0dbfe1ab57472ea27b62fe7208da35.jpg?
x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Ordos_International_Circuit.png
Descargando Thruxton Circuit desde
https://img2.51gt3.com/rac/track/202309/5e11e825b7c744f1a12804e871ddce
2d.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Thruxton_Circuit.png
Descargando Tianjin V1 International Circuit desde
https://img2.51gt3.com/rac/track/202302/8c26fc353e5949908e84c0438944e2
33.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Tianjin_V1_International_Circuit.png
Descargando Tianjin International Circuit E Circuit desde
https://img2.51gt3.com/rac/track/202305/a1d402e524a049a98bda0ac99545ce
e5.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Tianjin_International_Circuit_E_Circuit.png
Descargando Townsville Street Circuit desde
https://img2.51gt3.com/rac/track/202309/1ec9859cb53d46dab5e129eb6e5ea4
da.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Townsville_Street_Circuit.png
```

```
Descargando TT Circuit Assen desde
https://img2.51gt3.com/rac/track/202305/ef38d406a9384862acd1aac0d4349d
34.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\TT_Circuit_Assen.png
Descargando Virginia International Raceway desde
https://img2.51gt3.com/rac/track/202305/7993de9045364519b9526d2d162b5f
08.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Virginia_International_Raceway.png
Descargando Wanneroo Raceway desde
https://img2.51gt3.com/rac/track/202309/c2a25672128548ec99efc40cd8bb2f
84.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\Wanneroo_Raceway.png
Descargando Watkins Glen International desde
https://img2.51gt3.com/rac/track/202305/fbc2519ce917489ea6c385147e8b19
6a.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Watkins_Glen_International.png
Descargando WeatherTech Raceway Laguna Seca desde
https://img2.51gt3.com/rac/track/202305/cbf13c969f28425299c2c450576fe0
52.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
WeatherTech_Raceway_Laguna_Seca.png
Descargando Wuhan International Circuit desde
https://img2.51gt3.com/rac/track/202410/99f075093b6042c1a0a1f162b37e3d
16.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Wuhan_International_Circuit.png
Descargando Wuhan Street Circuit desde
https://img2.51gt3.com/rac/track/23c250a781d54be29653718bc5d686d2.jpg?
x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Wuhan_Street_Circuit.png
Descargando Xiamen International Circuit desde
https://img2.51gt3.com/rac/track/202302/4ec8480b071f446e830423f575b74c
59.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Xiamen_International_Circuit.png
Descargando Jiangsu Yancheng Street Circuit desde
https://img2.51gt3.com/rac/track/202405/56a4eb7540f44af3b2913a4486adf5
45.jpg?x-oss-process=style/_nowm...
```

```
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Jiangsu_Yancheng_Street_Circuit.png
Descargando Yas Marina Circuit desde
https://img2.51gt3.com/rac/track/202303/0dc5fd65ccd14cf8867584641f5649
ed.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Yas_Marina_Circuit.png
Descargando Yas Marina Circuit Short (Corkscrew) Circuit desde
https://img2.51gt3.com/rac/track/202305/1b8a8ec405e142179b425b21314987
81.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Yas_Marina_Circuit_Short_(Corkscrew)_Circuit.png
Descargando Zhejiang International Circuit desde
https://img2.51gt3.com/rac/track/b1825139fa334b2f9a2539ba1dd7d060.jpg?
x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Zhejiang_International_Circuit.png
Descargando Zhejiang Wuyi Sanmei Circuit desde
https://img2.51gt3.com/rac/track/202412/7e5882a6d97d4318a1c7cd95c97fd5
7b.jpg?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Zhejiang_Wuyi_Sanmei_Circuit.png
Descargando Zhengzhou International Autodrome desde
https://img2.51gt3.com/rac/track/202309/a6f039cc78c549eab5c9f8c6084dfe
cd.png?x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Zhengzhou_International_Autodrome.png
Descargando Zhuhai International Circuit desde
https://img2.51gt3.com/rac/track/6de339fb6f4b403cb7c4104605155921.jpg?
x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Zhuhai_International_Circuit.png
Descargando Zhuzhou International Circuit desde
https://img2.51gt3.com/rac/track/05d4b148889b47cfa14dab234549d697.jpg?
x-oss-process=style/_nowm...
Guardado en C:\Users\gonza\OneDrive\Escritorio\Universidad\4to curso\
MINERIA DE DATOS\TRABAJO FINAL\IMAGENES CIRCUITOS\
Zhuzhou_International_Circuit.png
```

Definimos la función para generar nuevas imágenes

```
import os
import numpy as np
from tensorflow.keras.preprocessing.image import load_img,
img_to_array, save_img
import cv2

# Ruta al directorio donde están tus imágenes
carpeta_original = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\FOTOS\CIRCUITOS BBDD"
carpeta_aumentadas = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\FOTOS\FOTOS RED"

# Función para aplicar transformaciones aleatorias
def generar_imagen_aleatoria(img_array):
    # Dimensiones de la imagen
    alto, ancho, _ = img_array.shape

    # Generar un factor aleatorio para estiramiento/compresión
horizontal y vertical
    factor_x = np.random.uniform(0.65, 1.35)  # Horizontal entre 80% y
120%
    factor_y = np.random.uniform(0.65, 1.35)  # Vertical entre 80% y
120%

    # Generar un ángulo aleatorio para la rotación
    angulo = np.random.uniform(-180,180 )  # Rotación entre -15° y 15°

    # Aplicar estiramiento/compresión
    matriz_transf = np.array([[factor_x, 0, 0],
                              [0, factor_y, 0]])
    img_transformada = cv2.warpAffine(img_array, matriz_transf,
(ancho, alto), borderMode=cv2.BORDER_REPLICATE)

    # Aplicar rotación
    centro = (ancho // 2, alto // 2)
    matriz_rot = cv2.getRotationMatrix2D(centro, angulo, 1)
    img_rotada = cv2.warpAffine(img_transformada, matriz_rot, (ancho,
alto), borderMode=cv2.BORDER_REPLICATE)

    return img_rotada
```

Generamos 15 nuevas imágenes por cada circuito original

```
# Procesar cada imagen
for imagen_nombre in os.listdir(carpeta_original):
    ruta_imagen = os.path.join(carpeta_original, imagen_nombre)

    try:
        # Cargar la imagen y convertirla en array
```

```
        img = load_img(ruta_imagen)
        img_array = img_to_array(img).astype(np.uint8)  # Asegurar
uint8 para OpenCV

        # Generar 15 imágenes aleatorias
        for i in range(15):
            nueva_imagen = generar_imagen_aleatoria(img_array)

            # Guardar la imagen generada
            ruta_guardado = os.path.join(
                carpeta_aumentadas,
                f"{os.path.splitext(imagen_nombre)[0]}_aleatoria_{i +
1}.png"
            )
            save_img(ruta_guardado, nueva_imagen)

        print(f"15 imágenes generadas para {imagen_nombre}")
    except Exception as e:
        print(f"Error procesando {imagen_nombre}: {e}")

print("Transformaciones completadas.")
```

```
15 imágenes generadas para Adelaide_Street_Circuit.png
15 imágenes generadas para Aintree.png
15 imágenes generadas para Ain_Diab.png
15 imágenes generadas para Albert_Park_Circuit.png
15 imágenes generadas para Autodromo_Juan_y_Oscar_Galvez.png
15 imágenes generadas para Autodromo_Nazionale_Monza.png
15 imágenes generadas para Autódromo_do_Estoril.png
15 imágenes generadas para Autódromo_Enzo_e_Dino_Ferrari.png
15 imágenes generadas para Autódromo_Hermanos_Rodríguez.png
15 imágenes generadas para Autódromo_Internacional_do_Algarvev.png
15 imágenes generadas para Autódromo_Internacional_Nelson_Piquet.png
15 imágenes generadas para Autódromo_Internazionale_del_Mugello.png
15 imágenes generadas para Autódromo_José_Carlos_Pace.png
15 imágenes generadas para AVUS.png
15 imágenes generadas para Bahrain_International_Circuit.png
15 imágenes generadas para Baku_City_Circuit.png
15 imágenes generadas para Brands_Hatch_Circuit.png
15 imágenes generadas para Buddh_International_Circuit.png
15 imágenes generadas para Charade_Circuit.png
15 imágenes generadas para Circuit de pedralbes.png
15 imágenes generadas para Circuito_da_Boavista.png
15 imágenes generadas para Circuito_de_Jerez.png
15 imágenes generadas para Circuito_Gilles_Villeneuve.png
15 imágenes generadas para Circuit_Bremgarten.png
15 imágenes generadas para Circuit_de_Barcelona-Catalunya.png
15 imágenes generadas para Circuit_de_Monaco.png
15 imágenes generadas para Circuit_de_Nevers_Magny-Cours.png
15 imágenes generadas para Circuit_de_Spa-Francorchamps.png
```

```
15 imágenes generadas para Circuit_Mont-Tremblant.png
15 imágenes generadas para Circuit_of_the_Americas.png
15 imágenes generadas para Circuit_Park_Zandvoort.png
15 imágenes generadas para Circuit_Paul_Ricard.png
15 imágenes generadas para Detroit_Street_Circuit.png
15 imágenes generadas para Dijon-Prenois_Circuit.png
15 imágenes generadas para Donington_Park.png
15 imágenes generadas para Fair_Park_Circuit.png
15 imágenes generadas para Fuji_International_Speedway_Circuit.png
15 imágenes generadas para Hockenheimring.png
15 imágenes generadas para Hungaroring.png
15 imágenes generadas para Indianapolis_Motor_Speedway.png
15 imágenes generadas para Istanbul_Park.png
15 imágenes generadas para Jarama.png
15 imágenes generadas para Jeddah_Corniche_Circuit.png
15 imágenes generadas para Korea_International_Circuit.png
15 imágenes generadas para Kyalami.png
15 imágenes generadas para Las_Vegas_Strip_Street_Circuit.png
15 imágenes generadas para Le_Mans.png
15 imágenes generadas para Long_Beach.png
15 imágenes generadas para Losail_International_Circuit.png
15 imágenes generadas para Marina_Bay_Street_Circuit.png
15 imágenes generadas para Miami_International_Autodrome.png
15 imágenes generadas para Monsanto_Park_Circuit.png
15 imágenes generadas para Montjuic.png
15 imágenes generadas para Mosport_International_Raceway.png
15 imágenes generadas para Nivelles-Baulers.png
15 imágenes generadas para Nürburgring.png
15 imágenes generadas para Okayama_International_Circuit.png
15 imágenes generadas para Pescara_Circuit.png
15 imágenes generadas para Phoenix_Street_Circuit.png
15 imágenes generadas para Prince_George_Circuit.png
15 imágenes generadas para Red_Bull_Ring.png
15 imágenes generadas para Reims-Gueux.png
15 imágenes generadas para Riverside_International_Circuit-png.png
15 imágenes generadas para Rouen-Les-Essarts.png
15 imágenes generadas para Scandinavian_Raceway.png
15 imágenes generadas para Sebring_International_Raceway.png
15 imágenes generadas para Sepang_International_Circuit.png
15 imágenes generadas para Shanghai_International_Circuit.png
15 imágenes generadas para Silverstone_Circuit.png
15 imágenes generadas para Sochi_Autodrom.png
15 imágenes generadas para Suzuka_Circuit.png
15 imágenes generadas para Valencia_Street_Circuit.png
15 imágenes generadas para Watkins_Glen.png
15 imágenes generadas para Yas_Marina_Circuit.png
15 imágenes generadas para Zeltweg.png
15 imágenes generadas para Zolder.png
Transformaciones completadas.
```

Creamos un dataframe que contenga los nombres de las imágenes y su clasificación

```
import os

# Rutas
carpeta_imagenes = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\FOTOS\FOTOS RED" # Carpeta
con todas las imágenes

# Cargar las etiquetas
df_etiquetas = df.drop(columns=['Circuito'])

# Listar las imágenes y ordenarlas
imagenes = sorted(os.listdir(carpeta_imagenes))  # Asegúrate de que
estén ordenadas alfabéticamente

# Comprobar si el número total de imágenes es divisible entre el
tamaño del bloque
tamanio_bloque = 16
if len(imagenes) % tamanio_bloque != 0:
    print("Advertencia: el número total de imágenes no es múltiplo del
tamaño del bloque.")

# Crear un nuevo DataFrame para almacenar nombres de imágenes y sus
etiquetas
df_imagenes_etiquetadas = pd.DataFrame(columns=["nombre_imagen",
"velocidad", "seguridad", "adelantamientos"])

# Asignar etiquetas a cada imagen en bloques
indice_etiquetas = 0
for i in range(0, len(imagenes), tamanio_bloque):
    bloque_imagenes = imagenes[i:i + tamanio_bloque]

    # Obtener las etiquetas correspondientes a este bloque
    if indice_etiquetas < len(df_etiquetas):
        etiquetas = df_etiquetas.iloc[indice_etiquetas].values  #
Extraer etiquetas como array
        for imagen in bloque_imagenes:
            df_imagenes_etiquetadas =
pd.concat([df_imagenes_etiquetadas, pd.DataFrame({
                "nombre_imagen": [imagen],
                "velocidad": [etiquetas[0]],
                "seguridad": [etiquetas[1]],
                "adelantamientos": [etiquetas[2]],
            })])
        indice_etiquetas += 1
    else:
        print("Advertencia: No hay suficientes filas de etiquetas para
todas las imágenes.")
        break
```

```
display(df_imagenes_etiquetadas)
```

```
              nombre_imagen velocidad seguridad adelantamientos
0                  AVUS.png     Lento    Seguro           Pocos
0     AVUS_aleatoria_1.png     Lento    Seguro           Pocos
0    AVUS_aleatoria_10.png     Lento    Seguro           Pocos
0    AVUS_aleatoria_11.png     Lento    Seguro           Pocos
0    AVUS_aleatoria_12.png     Lento    Seguro           Pocos
..                     ...       ...       ...             ...
0   Zolder_aleatoria_5.png    Rápido    Seguro           Pocos
0   Zolder_aleatoria_6.png    Rápido    Seguro           Pocos
0   Zolder_aleatoria_7.png    Rápido    Seguro           Pocos
0   Zolder_aleatoria_8.png    Rápido    Seguro           Pocos
0   Zolder_aleatoria_9.png    Rápido    Seguro           Pocos

[1216 rows x 4 columns]
```

Exportamos este dataframe a un csv para entrenar nuestra Red Convolucional

```
df_imagenes_etiquetadas.to_csv(r"C:\Users\gonza\OneDrive\Escritorio\
Universidad\4to curso\MINERIA DE DATOS\TRABAJO FINAL\DATOS EXTRAIDOS\
circuitos_clasificados.csv",index=False)
```

ANEXO 8

En este fichero se entrena una red neuronal convolucional que pueda predecir características de un circuito en base a su trazado

Cargamos las imágenes etiquetadas y dividimos en entrenamiento y test

```
from sklearn.model_selection import train_test_split
import pandas as pd
import os
import numpy as np
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
from sklearn.metrics import f1_score



df=pd.read_csv(r"C:\Users\gonza\OneDrive\Escritorio\Universidad\4to
curso\MINERIA DE DATOS\TRABAJO FINAL\DATOS EXTRAIDOS\
imágenes_etiquetadas.csv")
# Dividir en train y test (80%-20%)
train_df, test_df = train_test_split(df, test_size=0.2,
random_state=100473601, stratify=df[["velocidad", "seguridad",
"adelantamientos"]])



# Tamaño de las imágenes (ajústalo según tu modelo)
image_size = (128, 128)

def cargar_imagenes(df, carpeta_imagenes):
    imagenes = []
    etiquetas = []
    for _, row in df.iterrows():
        # Cargar y redimensionar la imagen
        ruta_imagen = os.path.join(carpeta_imagenes,
row["nombre_imagen"])
        img = load_img(ruta_imagen, target_size=image_size)
        img_array = img_to_array(img) / 255.0  # Normalizar entre 0 y
1
        imagenes.append(img_array)

        # Guardar las etiquetas como vector
        etiquetas.append([row["velocidad"], row["seguridad"],
row["adelantamientos"]])
    return np.array(imagenes), np.array(etiquetas)

# Cargar las imágenes y etiquetas
carpeta_imagenes = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\FOTOS\FOTOS RED"
X_train, y_train = cargar_imagenes(train_df, carpeta_imagenes)
X_val, y_val = cargar_imagenes(test_df, carpeta_imagenes)
```

Vamos a definir una rejilla de valores de hiperparámetros para probar todas las combinaciones y poder escoger la mejor red

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout
from tensorflow.keras.optimizers import Adam
import numpy as np
from sklearn.metrics import f1_score
from tqdm import tqdm

# Definir el grid de hiperparámetros, incluyendo 'epochs'
param_grid = {
    "num_filters": [(32, 64, 128), (64, 128, 256)],
    "dropout_rate": [0.3, 0.5],
    "learning_rate": [0.001, 0.0001],
    "batch_size": [32, 64,128],
    "epochs": [ 20,30,40]  # Añadido 'epochs' como un hiperparámetro
}


# Función para crear el modelo
def build_model(num_filters, dropout_rate, learning_rate):
    model = Sequential([
        Conv2D(num_filters[0], (3, 3), activation='relu',
input_shape=(128, 128, 3)),
        MaxPooling2D((2, 2)),
        Conv2D(num_filters[1], (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(num_filters[2], (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(dropout_rate),
        Dense(64, activation='relu'),
        Dense(3, activation='sigmoid')  # Salida para 3
características binarias
    ])
    model.compile(
        optimizer=Adam(learning_rate=learning_rate),
        loss='binary_crossentropy',
        metrics=['accuracy']  # F1 Score se calculará manualmente
    )
    return model

# Almacenar resultados
best_f1 = 0
best_model = None
```

```
best_params = None
```

En este bucle se prueban todas las redes y se escoge la mejor

```
# Bucle para probar combinaciones de hiperparámetros
for num_filters in param_grid["num_filters"]:
    for dropout_rate in param_grid["dropout_rate"]:
        for learning_rate in param_grid["learning_rate"]:
            for batch_size in param_grid["batch_size"]:
                for epochs in param_grid["epochs"]:  # Iterar sobre el
número de épocas
                    print(f"Entrenando con filtros {num_filters},
dropout {dropout_rate}, lr {learning_rate}, batch_size {batch_size},
epochs {epochs}")

                    # Construir el modelo
                    model = build_model(num_filters, dropout_rate,
learning_rate)

                    # Crear una barra de progreso para las épocas
                    with tqdm(total=epochs, desc=f'Epochs {epochs}',
ncols=100) as pbar:
                        # Callback para actualizar la barra de
progreso
                        class
TQDMProgressBar(tf.keras.callbacks.Callback):
                            def on_epoch_end(self, epoch, logs=None):
                                pbar.update(1)

                        # Entrenar el modelo con el callback
                        history = model.fit(
                            X_train, y_train,
                            validation_data=(X_val, y_val),
                            epochs=epochs,  # Número de épocas
variable
                            batch_size=batch_size,
                            verbose=0,  # Desactivado para usar la
barra de progreso
                            callbacks=[TQDMProgressBar()]
                        )

                    # Evaluar en el conjunto de validación
                    y_val_pred = (model.predict(X_val) >
0.5).astype(int)
                    f1 = f1_score(y_val, y_val_pred,
average='weighted')  # Weighted para clases desbalanceadas

                    print(f"F1 Score en validación: {f1}")
```

```
                    # Actualizar el mejor modelo si este es mejor
                    if f1 > best_f1:
                        best_f1 = f1
                        best_model = model
                        best_params = {
                            "num_filters": num_filters,
                            "dropout_rate": dropout_rate,
                            "learning_rate": learning_rate,
                            "batch_size": batch_size,
                            "epochs": epochs
                        }


# Guardar el mejor modelo
best_model.save("mejor_modelo_circuitos.h5")
print(f"Mejor modelo guardado con F1 Score: {best_f1}")
print(f"Mejores hiperparámetros: {best_params}")
```

```
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 32, epochs 20

Epochs 20: 100%|████████████████████████████████████████████████████|
20/20 [03:08<00:00,  9.44s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 72ms/step
F1 Score en validación: 0.7435032699572858

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)


Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 32, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████|
30/30 [04:07<00:00,  8.24s/it]

1/8 ━━━━━━━━━━━━━━━━━━━━ 1s 144ms/step



8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 81ms/step
F1 Score en validación: 0.7274328711248871
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 32, epochs 40
```

```
c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|██████████████████████████████████████████████████|
40/40 [05:26<00:00,  8.17s/it]

WARNING:tensorflow:5 out of the last 17 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x0000019C318F0040> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due
to (1) creating @tf.function repeatedly in a loop, (2) passing tensors
with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has reduce_retracing=True option that can avoid
unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more
details.
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 67ms/step
F1 Score en validación: 0.7176100628930817
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 64, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|██████████████████████████████████████████████████|
20/20 [10:41<00:00, 32.06s/it]

WARNING:tensorflow:5 out of the last 17 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_dist
ributed at 0x0000019BCDEDD1C0> triggered tf.function retracing.
Tracing is expensive and the excessive number of tracings could be due
to (1) creating @tf.function repeatedly in a loop, (2) passing tensors
with different shapes, (3) passing Python objects instead of tensors.
For (1), please define your @tf.function outside of the loop. For (2),
@tf.function has reduce_retracing=True option that can avoid
unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more
details.
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 76ms/step
```

```
F1 Score en validación: 0.7344734734405346
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 64, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 30: 100%|████████████████████████████████████████████████████|
30/30 [03:54<00:00,  7.83s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 63ms/step
F1 Score en validación: 0.7073038384231413
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 64, epochs 40

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|████████████████████████████████████████████████████|
40/40 [04:42<00:00,  7.05s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 67ms/step
F1 Score en validación: 0.7211390787166807
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 128, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|████████████████████████████████████████████████████|
20/20 [02:41<00:00,  8.05s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 67ms/step
F1 Score en validación: 0.7373605906302563
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 128, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
```

```
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 30: 100%|████████████████████████████████████████████████████|
30/30 [03:30<00:00,  7.03s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 73ms/step
F1 Score en validación: 0.7105457433170806
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.001,
batch_size 128, epochs 40

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|████████████████████████████████████████████████████|
40/40 [06:24<00:00,  9.62s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 70ms/step
F1 Score en validación: 0.7457495373303029
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001,
batch_size 32, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|████████████████████████████████████████████████████|
20/20 [02:34<00:00,  7.74s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 67ms/step
F1 Score en validación: 0.7235239173915716
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001,
batch_size 32, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
Epochs 30: 100%|██████████████████████████████████████████████████| 30/30 [03:42<00:00,  7.41s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 70ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7459748674453094
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001, batch_size 32, epochs 40

Epochs 40: 100%|██████████████████████████████████████████████████| 40/40 [05:14<00:00,  7.85s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 67ms/step
F1 Score en validación: 0.7047784784457717
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001, batch_size 64, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epochs 20: 100%|██████████████████████████████████████████████████| 20/20 [11:49<00:00, 35.48s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 69ms/step
F1 Score en validación: 0.7020604939685158
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001, batch_size 64, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epochs 30: 100%|██████████████████████████████████████████████████| 30/30 [03:32<00:00,  7.08s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 60ms/step
```

```
c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7182863774067517
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001,
batch_size 64, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████|
40/40 [05:01<00:00,  7.55s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 83ms/step
F1 Score en validación: 0.7225659996665049
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001,
batch_size 128, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|████████████████████████████████████████████████|
20/20 [02:21<00:00,  7.08s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 74ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7124532271389168
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001,
batch_size 128, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████|
30/30 [1:14:59<00:00, 149.99s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 90ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
```

```
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7135323454025385
Entrenando con filtros (32, 64, 128), dropout 0.3, lr 0.0001,
batch_size 128, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████|
40/40 [05:01<00:00,  7.55s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 96ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7019468475293327
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 32, epochs 20

Epochs 20: 100%|████████████████████████████████████████████████|
20/20 [02:46<00:00,  8.34s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 66ms/step
F1 Score en validación: 0.7071836238739077

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)


Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 32, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████|
30/30 [03:41<00:00,  7.39s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 102ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
```

```
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7249652995975349
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 32, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████|
40/40 [05:08<00:00,  7.71s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 69ms/step
F1 Score en validación: 0.7201656968796462
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 64, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|████████████████████████████████████████████████|
20/20 [02:36<00:00,  7.84s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 82ms/step
F1 Score en validación: 0.7311070491394432
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 64, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 30: 100%|████████████████████████████████████████████████|
30/30 [03:40<00:00,  7.35s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 68ms/step
F1 Score en validación: 0.7088858642825399
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 64, epochs 40

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
```

```
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|██████████████████████████████████████████████████|
40/40 [04:59<00:00,  7.49s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 75ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7340285793209794
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 128, epochs 20

Epochs 20: 100%|██████████████████████████████████████████████████|
20/20 [02:27<00:00,  7.39s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 65ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7452454570088283
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 128, epochs 30

Epochs 30: 100%|██████████████████████████████████████████████████|
30/30 [03:29<00:00,  6.99s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 69ms/step
F1 Score en validación: 0.7355049315780717
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.001,
batch_size 128, epochs 40

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
```

```
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|██████████████████████████████████████████████████|
40/40 [05:00<00:00,  7.52s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 62ms/step
F1 Score en validación: 0.7500320157090412
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 32, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|██████████████████████████████████████████████████|
20/20 [02:28<00:00,  7.42s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 69ms/step
F1 Score en validación: 0.7312948478222476
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 32, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 30: 100%|██████████████████████████████████████████████████|
30/30 [26:15<00:00, 52.53s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 66ms/step
F1 Score en validación: 0.7094894954832289
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 32, epochs 40

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|██████████████████████████████████████████████████|
40/40 [06:03<00:00,  9.09s/it]
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 113ms/step
F1 Score en validación: 0.7175898998376771
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 64, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|████████████████████████████████████████████████████|
20/20 [03:29<00:00, 10.45s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 108ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7013588398601358
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 64, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████|
30/30 [05:12<00:00, 10.42s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 126ms/step
F1 Score en validación: 0.7261751077133046
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 64, epochs 40

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|████████████████████████████████████████████████████|
40/40 [06:35<00:00,  9.90s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 151ms/step
F1 Score en validación: 0.7263791881530225
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 128, epochs 20
```

```
c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|████████████████████████████████████████████████████|
20/20 [03:29<00:00, 10.50s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 144ms/step
F1 Score en validación: 0.7040300296680527
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 128, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 30: 100%|████████████████████████████████████████████████████|
30/30 [04:51<00:00,  9.73s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 100ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7302284529725732
Entrenando con filtros (32, 64, 128), dropout 0.5, lr 0.0001,
batch_size 128, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████████|
40/40 [05:13<00:00,  7.85s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 78ms/step
F1 Score en validación: 0.7113185582467401
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 32, epochs 20

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
```

```
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 20: 100%|██████████████████████████████████████████████████|
20/20 [08:43<00:00, 26.19s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 193ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.699453978751528
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 32, epochs 30

Epochs 30: 100%|██████████████████████████████████████████████████|
30/30 [13:49<00:00, 27.66s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 221ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7084962491908426
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 32, epochs 40

Epochs 40: 100%|██████████████████████████████████████████████████|
40/40 [17:46<00:00, 26.65s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 187ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
F1 Score en validación: 0.7404824985127213
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 64, epochs 20

Epochs 20: 100%|████████████████████████████████████████████████|
20/20 [09:03<00:00, 27.15s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 200ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7075492749279702
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 64, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████|
30/30 [12:27<00:00, 24.93s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 172ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7376594728147291
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 64, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████|
40/40 [17:24<00:00, 26.11s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 185ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
F1 Score en validación: 0.7106270551656213
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 128, epochs 20

Epochs 20: 100%|████████████████████████████████████████████████|
20/20 [08:11<00:00, 24.55s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 240ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7252306379429782
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 128, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████|
30/30 [2:04:09<00:00, 248.32s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 173ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.7075649921581674
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.001,
batch_size 128, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████|
40/40 [36:11<00:00, 54.28s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 274ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

```
F1 Score en validación: 0.657542843099587
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001,
batch_size 32, epochs 20

Epochs 20: 100%|████████████████████████████████████████████████|
20/20 [12:53<00:00, 38.68s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 293ms/step
F1 Score en validación: 0.7198130746424938
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001,
batch_size 32, epochs 30

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 30: 100%|████████████████████████████████████████████████|
30/30 [21:04<00:00, 42.14s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 3s 290ms/step
F1 Score en validación: 0.7206885244687737
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001,
batch_size 32, epochs 40

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
Epochs 40: 100%|████████████████████████████████████████████████|
40/40 [17:58<00:00, 26.96s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 174ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When
using Sequential models, prefer using an `Input(shape)` object as the
first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

F1 Score en validación: 0.72609036618165
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001,
batch_size 64, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████| 20/20 [07:43<00:00, 23.19s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 172ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.734119537919914
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001, batch_size 64, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████| 30/30 [11:57<00:00, 23.93s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 210ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7338521588098337
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001, batch_size 64, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████| 40/40 [8:19:44<00:00, 749.61s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 169ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7211006171310862
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001, batch_size 128, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████████| 20/20 [08:12<00:00, 24.63s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 175ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7276369545291961
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001, batch_size 128, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████| 30/30 [1:13:16<00:00, 146.55s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 237ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.728302994845338
Entrenando con filtros (64, 128, 256), dropout 0.3, lr 0.0001, batch_size 128, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████████| 40/40 [16:26<00:00, 24.67s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 158ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.719131037078351
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 32, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████████████| 20/20 [08:13<00:00, 24.68s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 4s 512ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7425750683631004
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 32, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████████| 30/30 [12:37<00:00, 25.26s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 199ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7442121838198255
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 32, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████████████| 40/40 [16:38<00:00, 24.95s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 203ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.749802167419791
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 64, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████| 20/20 [08:36<00:00, 25.84s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 253ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7349025154578763
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 64, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████| 30/30 [12:26<00:00, 24.88s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 195ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7159903414128055
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 64, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████| 40/40 [16:45<00:00, 25.15s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 155ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.735254818259624
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 128, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████████████| 20/20 [08:15<00:00, 24.76s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 164ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7492149682053252
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 128, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████████| 30/30 [12:51<00:00, 25.72s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 195ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7369197751148732
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.001, batch_size 128, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████████████| 40/40 [17:32<00:00, 26.31s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 210ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7209826731970105
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 32, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████████████| 20/20 [08:30<00:00, 25.54s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 226ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7241420874704618
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 32, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████████| 30/30 [13:37<00:00, 27.24s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 197ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.6963237214841496
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 32, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████████████| 40/40 [17:40<00:00, 26.51s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 200ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.6858005268525176
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 64, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████████| 20/20 [08:32<00:00, 25.63s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 200ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7152811845538165
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 64, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████| 30/30 [13:12<00:00, 26.41s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 219ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7123279335528738
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 64, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████████| 40/40 [16:34<00:00, 24.86s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 198ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.7045746112206265
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 128, epochs 20
```

```
Epochs 20: 100%|████████████████████████████████████████████████████| 20/20 [09:14<00:00, 27.73s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 175ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.6794944678655436
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 128, epochs 30

Epochs 30: 100%|████████████████████████████████████████████████████| 30/30 [12:39<00:00, 25.31s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 195ms/step

c:\Users\gonza\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

F1 Score en validación: 0.6994664735919245
Entrenando con filtros (64, 128, 256), dropout 0.5, lr 0.0001, batch_size 128, epochs 40

Epochs 40: 100%|████████████████████████████████████████████████████| 40/40 [17:07<00:00, 25.70s/it]

8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 191ms/step

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.

F1 Score en validación: 0.6964191491009597
Mejor modelo guardado con F1 Score: 0.7500320157090412
Mejores hiperparámetros: {'num_filters': (32, 64, 128), 'dropout_rate': 0.5, 'learning_rate': 0.001, 'batch_size': 128, 'epochs': 40}
```

Una vez que tenemos la mejor red, definimos una función que cargue una imagen y prediga sus características

```
from tensorflow.keras.preprocessing.image import load_img,
img_to_array

def predecir_circuito(ruta_imagen, modelo, image_size=(128, 128)):
    # Cargar y preparar la imagen
    img = load_img(ruta_imagen, target_size=image_size)
    img_array = img_to_array(img) / 255.0  # Normalizar
    img_array = np.expand_dims(img_array, axis=0)  # Añadir dimensión
batch

    # Predicción
    predicciones = modelo.predict(img_array)
    return predicciones[0]

# Cargar el modelo
modelo = tf.keras.models.load_model("mejor_modelo_circuitos.h5")
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
```

Predecimos 5 circuitos españoles para ver sus características en caso de entrar en la Fórmula 1

```


# Tamaño de las imágenes usado durante el entrenamiento
image_size = (128, 128)


# Función para preparar una imagen
def preparar_imagen(ruta_imagen):
    img = load_img(ruta_imagen, target_size=image_size)  # Cargar la
imagen y redimensionarla
    img_array = img_to_array(img) / 255.0  # Normalizar los píxeles a
valores entre 0 y 1
    return np.expand_dims(img_array, axis=0)  # Añadir una dimensión
para el batch

# Ruta de la carpeta con las imágenes
carpeta_imagenes = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\FOTOS\FOTOS PRUEBA\New
folder"
# Obtener la lista de imágenes
imagenes = [f for f in os.listdir(carpeta_imagenes) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]

# Crear una lista para almacenar los resultados
resultados = []
```

```python
# Recorrer todas las imágenes de la carpeta
for imagen_nombre in imagenes:
    ruta_imagen = os.path.join(carpeta_imagenes, imagen_nombre)
    # Preparar la imagen
    imagen_procesada = preparar_imagen(ruta_imagen)
    # Realizar la predicción
    prediccion = modelo.predict(imagen_procesada)
    prediccion_binaria = (prediccion > 0.5).astype(int)  # Convertir a
valores binarios si es necesario
    # Guardar el resultado
    resultados.append({
        "nombre_imagen": imagen_nombre,
        "prediccion_continua": prediccion.tolist(),
        "prediccion_binaria": prediccion_binaria.tolist()
    })

# Crear mapeos inversos para decodificar las etiquetas
mapa_velocidad_inv = {0: "Lento", 1: "Rápido"}
mapa_seguridad_inv = {0: "Inseguro", 1: "Seguro"}
mapa_adelantamientos_inv = {0: "Pocos Adelantamientos", 1: "Muchos
Adelantamiendos"}

# Mostrar los resultados con mapeo
for resultado in resultados:
    print(f"Imagen: {resultado['nombre_imagen']}")

    # Predicciones continuas (sin cambios)
    print(f"Predicción continua: {resultado['prediccion_continua']}")

    # Predicciones binarias (convertir de 1/0 a etiquetas)
    pred_binaria = resultado['prediccion_binaria'][0]  # Extraer lista
de predicciones binarias
    print(f"Clasificación :  {mapa_velocidad_inv[pred_binaria[0]]}, "
          f" {mapa_seguridad_inv[pred_binaria[1]]}, "
          f" {mapa_adelantamientos_inv[pred_binaria[2]]}")
    print()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
Imagen: CHESTE.png
Predicción continua: [[1.0, 0.9999998807907104, 0.9999698996543884]]
Clasificación :  Rápido,  Seguro,  Muchos Adelantamiendos

Imagen: CIRCUITO ALMERIA.png
Predicción continua: [[0.9999918341636658, 0.992418110370636,
```

```
0.0011154836975038052]]
Clasificación :  Rápido,  Seguro,  Pocos Adelantamientos

Imagen: CIRCUITO DE NAVARRA.png
Predicción continua: [[1.0, 0.9999995231628418, 0.8977177739143372]]
Clasificación :  Rápido,  Seguro,  Muchos Adelantamiendos

Imagen: CIRCUITO GUADIX.jpg
Predicción continua: [[0.999999463558197, 1.584191522852052e-05,
0.011371226049959661]]
Clasificación :  Rápido,  Inseguro,  Pocos Adelantamientos

Imagen: CIRCUITO VALENCIA.png
Predicción continua: [[0.00011638475552899763, 0.9985663890838623,
4.4724185954692075e-07]]
Clasificación :  Lento,  Seguro,  Pocos Adelantamientos
```

A continuación vemos las características más típicas de los circuitos españoles

```
carpeta_imagenes = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\FOTOS\FOTOS PRUEBA"  # Cambia
esto a la ruta de tu carpeta
#carpeta_imagenes = r"C:\Users\gonza\OneDrive\Escritorio\Universidad\
4to curso\MINERIA DE DATOS\TRABAJO FINAL\FOTOS\FOTOS PRUEBA\New
folder"
# Obtener la lista de imágenes
imagenes = [f for f in os.listdir(carpeta_imagenes) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]

# Crear una lista para almacenar los resultados
resultados = []

# Recorrer todas las imágenes de la carpeta
for imagen_nombre in imagenes:
    ruta_imagen = os.path.join(carpeta_imagenes, imagen_nombre)
    # Preparar la imagen
    imagen_procesada = preparar_imagen(ruta_imagen)
    # Realizar la predicción
    prediccion = modelo.predict(imagen_procesada)
    prediccion_binaria = (prediccion > 0.5).astype(int)  # Convertir a
valores binarios si es necesario
    # Guardar el resultado
    resultados.append({
        "nombre_imagen": imagen_nombre,
        "prediccion_continua": prediccion.tolist(),
        "prediccion_binaria": prediccion_binaria.tolist()
    })
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 46ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
```

```python
from collections import Counter


# Lista de predicciones
resultados

# Extraer predicciones continuas y binarias
predicciones_continuas = [item['prediccion_continua'][0] for item in
resultados]
predicciones_binarias = [item['prediccion_binaria'][0] for item in
resultados]

# Convertir a arrays para facilitar operaciones
predicciones_continuas = np.array(predicciones_continuas)
predicciones_binarias = np.array(predicciones_binarias)

# Calcular la media de las predicciones continuas
media_continuas = np.mean(predicciones_continuas, axis=0)

# Calcular la media de las predicciones binarias
media_binarias = np.mean(predicciones_binarias, axis=0)

# Calcular la moda de las predicciones binarias
moda_binarias = []
for i in range(predicciones_binarias.shape[1]):
```

```
    columna = predicciones_binarias[:, i]
    moda = Counter(columna).most_common(1)[0][0]
    moda_binarias.append(moda)

# Resultados
print("Media de las predicciones continuas:", media_continuas)
print("Moda de las predicciones binarias:", moda_binarias)
```

```
Media de las predicciones continuas: [0.87536575 0.74794224
0.24814561]
Moda de las predicciones binarias: [np.int64(1), np.int64(1),
np.int64(0)]
```