TITULACIÓN
Grado en Ingeniería Matemática e Inteligencia Artificial
ADQUISICIÓN DE DATOS
Curso 2023-2024

Práctica Final: Formula 1

Objetivos generales

- Aplicar los conocimientos adquiridos en la asignatura, utilizando APIs, web scrapping y Pandas.
- Conocer el funcionamiento de la librería Scrapy.
- Analizar los datos obtenidos y extraer conclusiones.

Descripción de la práctica

En esta práctica, se va a tomar el papel de un analista de datos de un equipo de Formula 1. Mediante el uso de APIs y web scrapping, el objetivo de la práctica es conseguir un dataset que permita extraer conclusiones sobre la influencia de los pit-stops (paradas en boxes) en los resultados y otros parámetros del campeonato mundial de la Formula 1.

La práctica se realizará en grupos de 4 personas. **El entregable final de la práctica será un fichero formula_one.zip** que contenga los archivos .py utilizados y un pequeño documento (máximo 2 caras) con los resultados de la pregunta 4. **El código deberá contener un archivo main.py que llame a las distintas funciones / clases utilizadas y genere los archivos que se piden.**

Contenido de la práctica

1. **Implementación de un crawler de Scrapy.** Se pide implementar un crawler (o Spider) de la librería Scrapy. El objetivo del crawler será hacer web scraping de la página web de Wikipedia de las distintas temporadas de Formula 1 (desde 2012 hasta la actualidad). El crawler deberá llevar a cabo la siguiente tarea:
    a. Acceder a la tabla de resultados de cada una de las temporadas de la Formula 1. Por ejemplo, la temporada 2023 se encuentra en este enlace.

Results and standings

Grands Prix

Round	Grand Prix[g]	Pole position	Fastest lap	Winning driver	Winning constructor	Report
1	Bahrain Grand Prix	Max Verstappen	Zhou Guanyu	Max Verstappen	Red Bull Racing-Honda RBPT	Report
2	Saudi Arabian Grand Prix	Sergio Pérez	Max Verstappen	Sergio Pérez	Red Bull Racing-Honda RBPT	Report
3	Australian Grand Prix	Max Verstappen	Sergio Pérez	Max Verstappen	Red Bull Racing-Honda RBPT	Report
4	Azerbaijan Grand Prix	Charles Leclerc	George Russell	Sergio Pérez	Red Bull Racing-Honda RBPT	Report
5	Miami Grand Prix	Sergio Pérez	Max Verstappen	Max Verstappen	Red Bull Racing-Honda RBPT	Report
6	Monaco Grand Prix	Max Verstappen	Lewis Hamilton	Max Verstappen	Red Bull Racing-Honda RBPT	Report

    b. Parsear la tabla para obtener el enlace a cada uno de los “Report” de cada gran premio.

c. Acceder a cada uno de los “Report” siguiendo ese enlace, y obtener la siguiente tabla de resultados de cada carrera:

Race classification [edit]

Pos.	No.	Driver	Constructor	Laps	Time/Retired	Grid	Points
1	1	Max Verstappen	Red Bull Racing-Honda RBPT	57	1:33:56.736	1	25
2	11	Sergio Pérez	Red Bull Racing-Honda RBPT	57	+11.987	2	18

d. Esta tabla deberá ser convertida a un DataFrame y guardada en formato csv. Deberá crearse una carpeta por cada año (desde 2012 a 2023), y la carpeta contener los csv de cada una de las carreras (un csv por carrera).

2. **Acceso a los datos de pit-stops utilizando la Ergast F1 API.** Para esta sección, se va a utilizar la Ergast F1 API. Los pasos a seguir son los siguientes:
   a. Utilizando el endpoint de pit-stops, obtener los pit-stops de cada carrera desde 2012 hasta 2023.
   b. Utilizando el endpoint de drivers, obtener información adicional para cada piloto, más concretamente el mapping entre su “driverID” y su número. Este paso es importante ya que necesitaremos el número de cada piloto para poder enlazar con los resultados del apartado 1. **Nota:** Será necesario tener en cuenta que Max Verstappen cambia de número en 2022.
   c. Deberá producirse, para cada carrera desde 2012 a 2023, un DataFrame. Cada fila corresponderá a un Piloto, y deberá contener las columnas “DriverId”, “DriverNumber”, “NPitstops” (Número de pitstops en esa carrera), “MedianPitStopDuration” (Duración mediana de los pitstops del piloto en esa carrera).
   d. Estos DataFrames se deberán guardar en formato csv en las mismas carpetas que el apartado 1.
3. **Cruzado de los datos de ambas fuentes.** En esta sección, se deberá implementar un código que obtenga un único DataFrame de las siguientes características:
   a. Cada fila del DataFrame se corresponderá al resultado de un piloto en una carrera.
   b. Deberá tener las columnas de los DataFrames del apartado 1, y también las del apartado 2, es decir, se deberá hacer un merge de las dos fuentes de datos usando el identificador del piloto. Además, deberá tener las columnas “Season” and “RaceNumber” para identificar a qué temporada y carrera se corresponde ese resultado.
   c. Este DataFrame generado deberá ser exportado en formato csv.
4. **Análisis de resultados.** Utilizando el DataFrame generado en el apartado anterior, se analizarán los resultados y se extraerá alguna conclusión sobre los datos generados. **Esta conclusión deberá ir acompañada de gráficos que apoyen la hipótesis. En esta pregunta se valorará mucho la creatividad así como la buena presentación de las hipótesis y teorías generadas.**

Consejos generales

1. Es muy recomendable seguir el tutorial de Scrapy para programar el crawler. Todos los recursos necesarios para hacerlo correctamente se explican en este tutorial. Para poder ejecutar un crawler de Scrapy dentro de un código de Python, es útil seguir este otro ejemplo.
2. Scrapy no se puede ejecutar dentro de un jupyter notebook de forma sencilla, por lo que no es recomendable usarlos.
3. La Ergast F1 Api permite recibir respuestas en XML y en JSON. Es muy recomendable obtener las respuestas en formato JSON. Para ello, se deberá añadir ".json" al final de la URL de cada petición, tal y como se explica en la documentación.

Evaluación:

- 30% cada uno de los apartados 1, 2 y 3.
- 10% el apartado 4.

Nota: El formato y las buenas prácticas se tendrán en cuenta dentro de la nota de cada pregunta. Se recomienda utilizar un formateador (Black o Prettier por ejemplo) para el código. Así mismo, separar las distintas tareas en funciones / clases / archivos es muy recomendable.