SISTEMA DE GESTIÓN DE DATOS Y DE INFORMACIÓN

TEMA 1

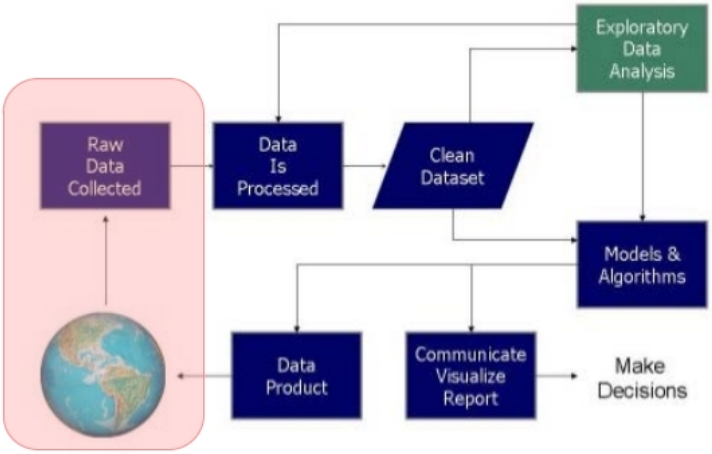
Captura y procesamiento de datos

1º Captura de datos 3
2º Glosario de ordenes 4
2.1 Librería Request 4
2.2 Librería Json 4
2.3 Librería Tabula 4
2.4 Libreria Pandas 4
2.4.1 Clase Dataframe 5
2.5 Libreria Selenium 8
2.5.1 Clase WebDriver 8
2.5.2 Clase WebElement 9

1º Captura de datos

– El análisis de datos compone todo un ciclo de etapas que tienen como objetivo obtener información relevante a cerca de los datos analizados. La primera fase de este ciclo es la recopilación de datos que vamos a analizar, lo cual puede realizarse de múltiples maneras:

- A través del uso de ficheros que sean accesibles mediante una URL. Para esto utilizamos descarga directa mediante la biblioteca Request de Python.
- Mediante el uso de APIs, las cuales pueden exigir identificación y el tratamiento es diferente para cada una de ellas.
- Utilizando técnicas de Web Scraping, las cuales se emplean en aquellas páginas donde el acceso a los datos esta pensado mediante el uso de interacción humana.



– Mientras que el Web Scraping se emplea en paginas web que han sido previamente seleccionadas y estudiadas, el Web Crawling se utiliza para operar con una gran cantidad de páginas web distintas, utilizando los links para viajar de una a otra con el objetivo de buscar una información determinada.

– El Web Scraping es una forma muy útil de obtener los datos de aquellas páginas que muestran información de forma automatizada, aunque puede ser necesario examinar el código HTML para ello. Podemos diferenciar varios tipos de páginas de Web Scraping:

- **Páginas estáticas:** Al pone la URL en el navegador obtenemos la información como parte de la página web resultante. En este caso, los ficheros pueden ser parte de la página (BeautifulShop) o estar en un fichero que hay que analizar (Request).
- **Páginas dinámicas:** La página web incluye un formulario que se debe rellenar para llegar a la páginas web con los datos deseados. Puede ser que el formulario genere una nueva URL con los datos (Request) o simplemente una nueva página sin cambiar la URL (Selenium).

– Selenium es una librería que brinda la posibilidad de navegar e interactuar con los distintos elementos que componen una página Web, lo cual es muy útil para obtener información y hacer test de servicios brindados por dicha página.

– Para poder utilizar Selenium necesitaremos un dichero intermedio que funcione de enlace entre la propia biblioteca y el buscador utilizado. Cada navegador utiliza un fichero diferente, en el caso de Google Chrome usaremos Chromedriver.exe.

2º Glosario de ordenes

2.1 Librería Request

Nos permite descargar un fichero web a través de la URL donde se encuentran.

get() => Descarga el fichero web que se encuentra a través de la dirección indicada.

- **url:** Dirección del fichero que queremos descargar.
- **params:** Se trata de una colección de claves par-valor donde se indican los posibles parámetros que queremos pasar a la página web a la que vamos a hacer la descarga.

  Return: Request.response object

2.2 Librería Json

Nos permite crear e interaccionar con estructuras de dato de tipo JSON.

loads() => Transcribe la información pasada como parámetro a formato JSON.

- **Value:** Cadena de caracteres que queremos transcribir.

  Return: Python dictionary

2.3 Librería Tabula

La librería tabula nos permite leer tablas que se encuentran dentro de ficheros en formato pfd

read_pdf() => Lee todas las tablas que se encuentran dentro del fichero pdf indicado y las devuelve organizadas en una lista.

- **Nombre:** Dirección del fichero del cual queremos obtener las tablas.

2.4 Libreria Pandas

Esta librería nos permite descargar ficheros web con formatos especiales, es decir, que sus valores pueden estar separador por caracteres de separación. También nos permite trabajar con la estructura de datos denominada dataframe.

read_csv() => Descarga el fichero web que se encuentra a través de la dirección indicada, el cual se encuentra en formato csv (utilizando caracteres de separación). El fichero es retornado como un dataframe.

- **url:** Dirección del fichero que queremos descargar.
- **sep:** El separador, por defecto “,”.
- **header:** Para indicar si la primera línea contiene la cabecera (por defecto True).
- **thousands:** Decimal: separadores de miles y de decimales.
- **encoding:** Codificación de caracteres. Deber ser una codificaciones estándar

read_excel() => Descarga el fichero web que se encuentra a través de la dirección indicada, el cual se encuentra en formato excel. El fichero es retornado como un dataframe.

- **url:** Dirección del fichero que queremos descargar.

DataFrame() => Crea y retorna una estructura Dataframe a partir de los datos indicados

- **data:** Datos a partir del cual queremos crear la clase DataFrame. Pueden indicarse como:
  - Una lista de listas, donde cada una de estas últimas representa una fila dentro del DataFrame.
  - Un diccionario, donde las claves son un string con el nombre de cada columna y los valores son las listas con los valores de cada fila.
- **Colums:** Especifica el nombre de cada columna, los cuales deben expresarse como una lista de strings.

  Return: DataFrame object

2.4.1 Clase Dataframe

Se trata de una clase incluida dentro de la librería Pandas, la cual organiza sus datos en una tabla segmentada en filas y columnas, donde las cabeceras de las columnas indican los valores de las mismas y las filas suelen estar organizadas por índices. Cada uno de los elementos de la tabla puede albergar un único valor.

En el caso de querer acceder a una de la columnas, únicamente necesitaremos indicar el valor de referencia de las mismas. En el caso de querer acceder a más de una columna, deberemos indicar el nombre de cada una de ellas formando una lista.

Para eliminar una columna podemos utilizar la palabra reservada “del”, la cual suprime aquello que sea expresado a continuación.

Para filtrar filas lo normal es escribir una expresión booleana que solo cumplan las filas que queremos y acceder mediante este filtro. Esto es debido a que Python permite usar una secuencia de Trues y False para acceder a elementos, devolviendo solo en los que hay Trues. Esta expresión se la pasamos por parámetro al dataframe, de la misma manera que hacemos al acceder a las columnas.

Algunos de los atributos principales (los parámetros se pasan entre corchetes):

- **columns:** Devuelve una lista que contiene las cabeceras de las columnas del DataFrame.
- **Index:** Muestra información sobre las filas que conforman el dataframe, pudiendo también modificar las referencias a las mismas. Subatributos:
  - **start:** Devuelve el primer índice.
  - **stop:** Devuelve el último índice.
  - **step:** Devuelve el salto numérico que hay entre dos índices consecutivos.
- **Values:** Retorna una lista con valor de cada una de las filas almacenadas en el DataFrame. También se le puede indicar el índice de una fila en concreto como parámetro para que devuelva la misma.
- **Iloc:** Retorna las filas del dataframe cuyos índices se han indicado como parámetro. Usando un segundo índice también podemos acceder a una columna en concreto dentro de la fila.
- **loc:** Retorna las filas y columnas indicadas como primer y segundo parámetro, respectivamente. En este caso, las filas y columnas deben expresarse mediante índices.

to_csv() => Guarda en memoria un datagrama en formato csv. Se trata de un método de la clase Dataframe que nos proporciona Pandas.

- **Nombre:** Dirección del fichero que queremos guardar.
- **Index:** Flag que nos indica si queremos exportar el índice del dataframe (primera columna).
- **Encoding:** Indica el juego de caracteres que utiliza el dataframe.

to_excel() => Guarda en memoria un datagrama en formato csv. Se trata de un método de la clase Dataframe que nos proporciona Pandas.

- **Nombre:** Dirección del fichero que queremos guardar.

drop() => Excluye del dataframe aquellos elementos indicados como parámetro y retorna el dataframe resultante.

Return: DataFrame object

DataFrame() => Crea un nuevo dataframe resultante de añadir al mismo nuevas filas cuyos datos son pasados como parámetro. El nuevo dataframe será retornado.

- **Data:** Información a incluir en el DataFrame, aportada mediante un diccionario.
- **ignore_index:** Flag que indica si la información debe añadirse de forma indexada.

Return: DataFrame object

sort_values() => Crea una copia del datagrama con las filas ordenadas y lo retorna.

- **By:** Indica la referencia a la fila por la que se quiere ordenar el datagrama.
- **Ascending:** Flag que configura si la ordenación debe ser ascendente.
- **Inplace:** Flag que modifica el comportamiento de la función haciendo que en vez de retornar un datagrama modifique aquel sobre el que se esta realizando la operación.

Return: DataFrame object

reindex()=> Crea un nuevo datagrama igual que aquel sobre el que ejecutamos la función reindexando sus filas y lo retorna.

- **Inplace:** Flag que modifica el comportamiento de la función haciendo que en vez de retornar un datagrama modifique aquel sobre el que se esta realizando la operación.

Return: DataFrame object

duplicated() => Devuelve una lista de booleanos que indican por cada una de las filas del datagrama, si esta esta repetida.

Return: Lista Boolean

drop_duplicated() => Elimina las filas duplicadas del datagrama. Se puede especificar como argumento la referencia a aquellas columnas concretas sobre cuyos valores se quiere comprobar la duplicidad (por defecto comprueba que exista duplicidad en todas las columnas).

Return: Lista Boolean

Isnull() => Devuelve un dataframe donde indica en cada uno de los campos si el dataframe original sobre el que se ejecuta el método tiene dicho campo a null.

Return: DataFrame object

dropna() => Elimina aquellas filas que tienen algún campo nulo del datagrama sobre el cual se ejecuta la función. El resultado lo devuelven ensamblado en otro datagrama.

- **How:** Especifica aquellas columnas cuyo valor tiene que ser nulo para eliminar una fila. Si asignamos dicho parámetro a "all" deberá ser nulo en todas las columnas del dataframe.
- **Axis:** Flag que hace a la función eliminar aquellas columnas del dataframe que se han quedado vacías tras la eliminación de nulos.
- **Thresh:** Parámetro con el cual la función eliminará todas aquellas filas cuya cantidad de valores distintos de nulo no sea igual o superior que al valor indicado.
- **Subset:** Especifica la referencia a aquellas columnas que se van a comprobar, de modo que se eliminarán todas aquellas filas cuyo valor sea nulo para dichas columnas.

Return: DataFrame object

fillna() => Busca los valores nulos que componen el datagrama y los sustituye por el valor indicado como parámetro.

- **Value:** Valor por el que se va a realizar la sustitución. Este valor puede ser alfanumérico, en cuyo caso se sustituirán todos los nulos de la tabla por el mismo, o puede ser un diccionario, donde se indique el valor para cada uno de las columnas del datagrama.

Return: DataFrame object

replace() => Sustituye los valores indicados que se encuentran definidos dentro del dataframe y devuelve el resultado en forma de un nuevo dataframe.

- **to_replace:** Valor a reemplazar.
- **to_value:** Valor por el cual vamos a reemplazar.

Return: DataFrame object

"columna".hist() => Crea un histograma con los valores de la columna sobre la cual se esta ejecutando el método.

"columna".unique() => Devuelve una lista con cada uno de los diferentes valores que puede tomar la columna sobre la que ejecutamos la función. Retorna el número de valores distintos encontrados.

Return: Int

“columna”.value_counts() => Devuelve una lista con el número de valores distintos que podemos encontrar en la columna sobre la que ejecutamos la función y el número de veces que aparecen cada uno de ellos.

Return: Series

“columna”.value_counts() => Devuelve una lista booleana sobre que indica si para cada valor de la columna sobre la que se ejecuta la función esta repetido en el datagrama o no.

Return: Series

“columna”.map() => Aplica la función pasada como parámetro a cada uno de los valores que componen la columna sobre la que ejecutamos la función map.

- **function:** Nombre de la función que vamos a aplicar.

  Return: Series

2.5 Libreria Selenium

La librería Selenium nos permite interaccionar con páginas web dinámicas y poder obtener datos de las mismas. Podremos simular la interacción de un usuario con dicha página web.

2.5.1 Clase WebDriver

Webdriver es la clase principal de Selenium, de modo que cada instancia de la misma nos proporciona el control automatizado sobre una sesión del navegador.

Chrome() => Retorna una instancia de la clase WebDriver preparada para interacción con una sesión del buscador Chrome. Esta función abre automáticamente la pestaña de búsqueda de nuestra navegador con la cual se ha enlazado la variable retornada.

- **Service:** Driver de servicio para la ejecución de la sesión en el buscador.
- **Options:** Opciones del buscador Chrome.

  Return:WebDriver

get() => Realiza la conexión del controlador automático con la página web correspondiente a la dirección URL pasada como parámetro.

- **Url:** Dirección web con la que queremos realizar la conexión.

find_element() => Busca un elemento en el documento HTML correspondiente a la página web con la que hemos realizado la conexión previamente. Esta búsqueda se puede realiza por una gran diversidad de características de los elementos.

- **FuncionBusqueda:** Función que indica el tipo de búsqueda que se va a realizar. Estas funciones se referencian mediante el parámetro By y pueden ser las siguientes:
  - By.ID

- By.NAME
- By.XPATH
- By.LINK_TEXT
- By.PARTIAL_LINK_TEXT
- By.TAG_NAME
- By.CLASS_NAME
- By.CSS_SELECTOR

- **ValorBusqueda:** Indica el valor por el que va a realizarse la búsqueda.

  Return: Promise<WebElement>

find_elements() => Realiza la misma acción que el método find_element pero con la diferencia de que este devuelve un array con todos los elementos que coinciden con la búsqueda solicitada. También consta de los mismos parámetros.

Return: Promise<Array<WebElement>>

2.5.2 Clase_WebElement

Esta clase referencia a un elemento web y normalmente es utilizada para interaccionar con aquellos elementos que se encuentran dentro del HTML con el que hemos realizado la conexión anteriormente.

Click() => Equivalente a realizar un click sobre el elemento que tenemos referenciado.

Return:Promise<undefine>

send_keys() => Equivale a realzar una entrada mediante teclado sobre el elemento que tenemos referenciado.

- **Input:** Serie de teclas que representan la entrada desde teclado, la cual puede ser numérica o una cadena de caracteres. Para indicar la pulsación de una tecla en concreto podemos utilizar la clase Keys.

  Return:Promise<undefine>