ASIGNATURA:
M2.851 - Tipología y ciclo de vida de los datos

# Práctica 1 - Web Scraping:

## "Panorámica de empleo - Data Scientist"

Ginés Molina Abril
Iñigo Alvarez Bianchi

09/11/2020

Ginés Molina Abril
Iñigo Alvarez Bianchi

# 1 Contexto

Nos interesa tener información amplia, manejable y relevante sobre las **ofertas de empleo** disponibles en el campo de Data Science por lo que hemos decidido hacer web scraping a **LinkedIn** ya que, además de red social profesional, es uno de los portales de empleo más importantes que hay. Esto nos puede ofrecer una visión mucho más clara del panorama actual y las distintas variables que afectan a cada uno de los puestos que son ofertados en esta web.

La idea se basa en poder obtener una lista de empleos según un rol demandado separados en distintas bases de datos por lugares (todo el mundo, países, ciudades, regiones). Estas bases de datos contendrán información relevante de las posiciones que posteriormente analizaremos y categorizaremos para encontrar métricas y patrones interesantes según su localización. También nos gustaría evaluar las diferencias entre las ofertas en un lugar concreto y las que se ofertan en remoto.

En resumen, la idea es que nos pueda ofrecer una **panorámica actual** de nuestros perfiles profesionales y nos sirva como herramienta para encontrar un trabajo posteriormente teniendo bien claros todos los detalles y obtener un claro indicador de la oferta que mejor se puede ajustar a lo que estamos buscando y que más se pueda ajustarse a nuestro perfil y expectativas.

# 2 Título

**“Panorámica de empleo - Data Scientist”**

# 3 Descripción del dataset

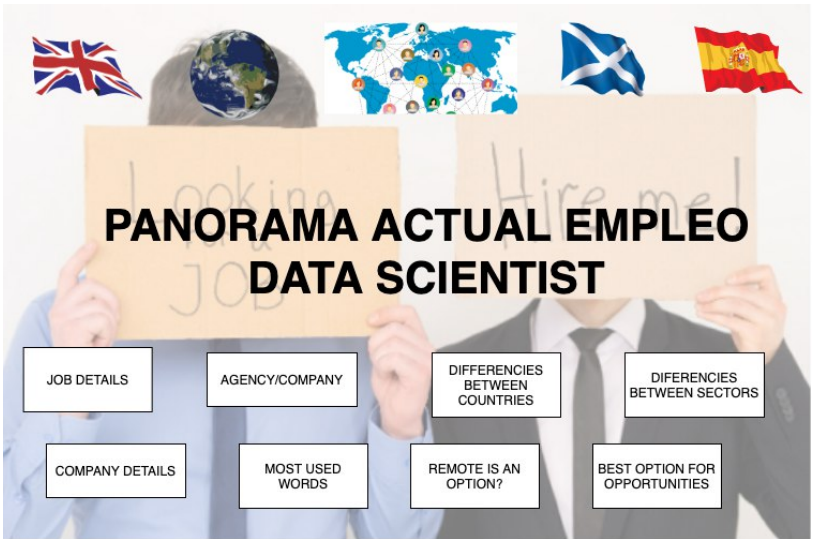
Este dataset es una recopilación de las ofertas de trabajo, en el campo de Data Science, publicadas en LinkedIn con sus datos más relevantes para poder realizar posteriormente un análisis. Debido a las limitaciones del sistema de búsqueda de Linkedin se pueden obtener un máximo de 1000 ofertas (40 páginas de resultados) por cada búsqueda realizada y ordenadas por relevancia.

Concretamente hemos obtenido los datos eligiendo distintas regiones para los puestos de trabajo: Todo el mundo, Reino Unido, España, Escocia y remoto.

Los ficheros csv obtenidos son los siguientes:

- Remote.csv: datos de las ofertas de empleo para trabajar en remoto
- Scotland.csv: datos de las ofertas de empleo para trabajar en Escocia
- Spain.csv: datos de las ofertas de empleo para trabajar en España
- UK.csv: datos de las ofertas de empleo para trabajar en el Reino Unido
- WorldWide.csv: datos de las ofertas de empleo a nivel mundial

# 4 Representación gráfica




# 5 Contenido

Definimos los siguientes campos para las ofertas de trabajo encontradas:

**Job_ID:** identificador de la oferta de empleo
**Date:** fecha de publicación
**Company_Name:** nombre de la empresa
**Role:** título o puesto de la oferta
**Location**: ubicación del puesto
**Description**: descripción del puesto (del que se puede extraer más información)
**Level**: nivel de experiencia requerida para el puesto
**Type:** tipo de contrato (jornada completa o parcial)
**Functions:** funciones del puesto de la oferta
**Industries:** sectores que involucra
**Solicitudes:** número de solicitudes enviadas a la oferta
**Empleados**: intervalo/número de empleados de la empresa
**Quick Application (True/False):** método de solicitud rápida por LinkedIn
**Emails**: e-mails de contacto
**Visualizaciones:** número de visualizaciones que tiene la oferta
**Recommended Flavor:** tipo de oferta

# 6 Agradecimientos

Ha sido posible elaborar este dataset gracias a las ofertas publicadas en LinkedIn por parte de las empresas y agencias de selección. Para obtenerlo hemos usado Python con Selenium y Beautiful Soup con diversas técnicas de web scraping para que sea lo menos sospechoso e invasivo posible. Especial mención a todos los artículos que estamos leyendo en internet al respecto para poder cumplir con lo que nos hemos propuesto.

# 7 Inspiración

Tener acceso a un dataset con las ofertas de trabajo que nos pueden resultar interesantes nos parece algo muy relevante para ver cuál es la situación de la contratación en el sector analizando los datos y también poder realizar unos filtrados adecuados para seleccionar las ofertas más interesantes y poder mandar nuestras candidaturas. Esto siempre puede ser útil y práctico, pero en la situación actual de crisis e incertidumbre el mercado de trabajo cobra una especial relevancia.

La principal inspiración sobre todo es el poder aplicar lo que estamos aprendiendo y que realmente nos acabe sirviendo de manera técnica y con resultados en nuestra vida profesional al solicitar un trabajo que cumpla las expectativas que buscamos. También es un buen portfolio, invertimos tiempo y tenemos resultado en la asignatura y esperamos obtener resultados en nuestro futuro profesional con su aplicación.

# 8 Licencia

Este dataset está protegido por copyright con todos los derechos de autor reservados, sin concesión de licencia de uso, por lo que no se permite su difusión, manipulación o uso de ningún tipo sin la autorización expresa de los autores.

No hemos elegido una licencia open-source ya que pensamos que puede no ser adecuada la difusión de los datos obtenidos al no tener expresamente la autorización por escrito de la compañía propietaria de los datos y el servicio web.

# 9 Código

En el siguiente enlace a Github se puede acceder al repositorio privado del proyecto (es necesario otorgar permiso):

GitHub

https://github.com/yinx89/WebScrapping

## 9.1 Propuesta de resolución de dificultades

### 9.1.1 Uso de proxies, red Tor y VPN

Ante el reto de tener que proteger la ejecución del script se decidió en un primer lugar la conexión con la red Tor y así poder saltar entre distintos proxies para evitar el bloqueo de IPs una vez se forma parte del circuito Tor. Por la forma en la que LinkedIn establece las medidas

anti-scraping parece que no se trata de una solución simple. Es necesario la autenticación en el sitio para obtener un mejor desempeño de la búsqueda y esta va a ser la dificultad que se va a tener que sortear ya que la localización va a jugar un papel importante. Por otro lado, se ha verificado que la web registra si se trata de una actividad sospechosa con chequeos como el lugar en el que queda establecido el usuario, su dirección IP e historiales de conexión, entre otros.

Hemos intentado también el uso de VPN establecidos en los lugares o países de procedencia para evitar la catalogación de posible salto entre países en segundos, pero esto nos ha conducido a pensar que se trata de algunas cosas más que deberíamos tener en cuenta. LinkedIn posee un sistema de detección de actividad sospechosa basada en estos aspectos, por lo que, para el desarrollo del script se ha decidido utilizar otros métodos para intentar no ser catalogado como actividad sospechosa y hacer un barrido de las búsquedas lo más "humano" posible manteniendo un mismo "lugar" o valores coherentes de localización.

Entendemos que podríamos contratar servicios más sofisticados de proxies propios (no gratuitos) y poder saltar entre ellos, hay varias empresas que ofertan estos proxies con características de localización que nos ayudarían a solucionar el problema de las localizaciones y de un modo muy personalizado. Pero, en definitiva, optamos por las soluciones mas baratas.

### 9.1.2 Resolución de CAPTCHA

Ante las dificultades que ofrece la página para ser escapeada nos encontramos con esta medida de seguridad que es lanzada cuando se realizan demasiadas peticiones. Con la ejecución del código se mandan demasiadas peticiones que son idénticas al servidor (se va obteniendo los valores de manera secuencial). Se ha optado por controlar cuando aparece esta comprobación de seguridad y realizarla manualmente, pudiendo continuar con la ejecución del código una vez resuelta.

Se ha investigado al respecto y también existen servicios sofisticados de resolución de captchas. Hemos optado por finalmente obviar que se trata de una comprobación que deberemos hacer pasadas algunas horas, cuando se realizan demasiadas ejecuciones de código idénticas y secuenciales.

### 9.1.3 Scroll-down con Javascript

Otra de las dificultades que presenta la web es la de construcción de contenido dinámico. Esto supone que no todos los elementos van a ser cargados de inicio y se tendrá que controlar que el elemento que se puede estar buscando puede no estar cargado en ese instante. La primera medida que va enfocada en mitigar este problema es la carga de los resultados de búsqueda a través de código Javascript que, de manera secuencial, revisa el número de elementos del resultado de la búsqueda y los va cargando "por bloques" dejando un tiempo prudencial e imitando un scroll-down del usuario.

Hemos pensado que sería interesante incorporar la funcionalidad extra de los clicks en los títulos de empleo de manera aleatoria, de esta manera se podrían registrar patrones distintos y una actividad más "humana".

### 9.1.4 Pagination

De la misma manera que el apartado anterior, se debe ir cargando el contenido que registran las 40 páginas de búsqueda de manera secuencial. Como anteriormente se ha mencionado, en algunas búsquedas de resultados (como por ejemplo en todo el mundo) el número máximo de registros queda limitado a los primeros 1000 registros ordenados por relevancia según el propio LinkedIn. Este aspecto se podría modificar para filtrar los resultados más recientes, pero hemos detectado que muchos de ellos están incompletos por las empresas y no ofrecen demasiado valor, por lo que hemos considerado este filtrado más relevante para el estudio.

### 9.1.5 Uso de tqdm

Se utiliza tqdm para poder tener una actualización detallada sobre el desarrollo del script y las fases/tiempos/bloques restantes. Esto es de gran importancia para poder evaluar cuando se detecta un posible elemento que aún no fue cargado y se puede visualizar como el script vuelve a ejecutar por ejemplo el scroll-down, pues hay veces que la conexión a internet no es estable y esto debe ser controlado también para que no establezca valores incorrectos o detenga la ejecución del script.

### 9.1.6 Reintentar si falla la carga de una función – Esperas implícitas

Esta función nos ayuda a controlar que el resto de las funciones del script sean ejecutadas correctamente, controlar las excepciones y aplicar un retardo apropiado en el caso de que se necesite más tiempo para su ejecución por lo mismo que se ha descrito en apartados anteriores. Además, hemos decidido utilizar las esperas implícitas (ya que las esperas explícitas siempre requerían confirmación de identidad desde el sitio web) y así podemos mantener un barrido de la página sin problemas. Cuando se ha podido cargar ese elemento y es encontrado, entonces se puede continuar, estableciendo su valor determinado o marcándolo como “None” si no existe.

### 9.1.7 Establecimiento de User-Agent

Se realiza la falsificación del User-Agent como otra de las medidas para que el script no sea bloqueado. También se ha utilizado la librería fake_useragent, pero nos ha traído más problemas al tratarse de algunos User-Agent obsoletos o antiguos para lo que es requerido para la utilización de LinkedIn.

### 9.1.8 Class Information – pandas - Codificación

Se decide utilizar una clase “Information”, con las que ir almacenando las listas resultantes y poder establecer sus funciones asociadas para, por ejemplo, su almacenamiento como .csv a través de “pandas”. Estas listas son resultado de cada una de las funciones que se van evaluando de manera secuencial para poder obtener cada una de las variables que se buscan de cada puesto de trabajo. Valores como el de “descripción” ha tenido que ser limpiado previamente a través de replace()/strip() para tener controlado que se almacenan correctamente todos los caracteres. En nuestro caso utilizamos “utf-8-signed”, apropiado para ver correctamente todos los caracteres en descripciones por todo el mundo y en distintos idiomas.

### 9.1.9 Búsqueda de elementos – search_x(query,data)

Se detalla un DOCSTRING por cada una de estas funciones y reciben como argumentos una consulta o query y el objeto data. La idea es que estas funciones desarrolladas de manera modular puedan ser modificadas de manera mucho más clara en el futuro y que controlen los distintos aspectos en específico de cada variable que buscamos en cada elemento en concreto.

Cada consulta es pasada a cada función search_X para encontrar la(s) variable(s) que necesitamos respecto a cada uno de los elementos. Esto se ha decidido realizar de esta manera para que sea mucho más claro en su funcionamiento y tratamiento. Esta query queda, por lo tanto, apuntando a cada una de las variables que son necesarias en cada uno de los componentes refiriéndose a su clase o por ejemplo a su disposición jerárquica dentro del html.

Se utilizan gran variedad de métodos para obtener los elementos. Lo que hemos entendido que siempre será bueno apuntar a los elementos de estas maneras excepto por "id", ya que se trata de contenido dinámico, y estos valores "id" van cambiando en cada ejecución. Además, se establece el registro como "None" si no se encuentra ese variable pero sí se ha cargado el elemento.

### 9.1.10 Ramas GIT

Se utiliza un método jerárquico de ramas para poder facilitar el trabajo entre los dos miembros del proyecto. Una rama para facilitar el trabajo independiente de cada uno de ellos, una rama "develop" para poder realizar los "merge" y tratar posibles fallos, y una rama "máster" que será la que obtendrá el código ya testado.

### 9.1.11 Nuevas variables interesante para evaluar – Problemas con variable de salario

Por un lado, hemos incluido dos variables que consideramos interesantes al primer estudio previo y que las hemos descubierto tras la primera valoración del proyecto. En primer lugar, la variable "emails" que encuentra los correos asociados a la oferta laboral en la descripción del puesto de trabajo, porque identificamos que muchos puestos de trabajo prefieren el contacto por correo.

Por otro lado, evaluando la variable "href" de cada uno de los puestos de trabajo identificamos un campo "Recommended Flavor" que podría darnos pistas interesantes sobre el tipo de oferta que se trata y así poder realizar un análisis interesante resultado de esta categorización.

Finalmente hemos decidido prescindir por ahora de evaluar el campo "Salario" por varios motivos. El primero es que no existe un salario definido en cada oferta de trabajo. El segundo es que existe muchos formatos al tratarse de varias monedas, decimales, rangos de precios… Esto dificultaría un correcto análisis del salario. Muchos de estos puestos de trabajo tampoco ofrecían contenido de salario "estimado" por lo que hemos tenido que prescindir por ahora de este campo. También hay algunos puestos de trabajo que, ofreciendo este rango de salarios según experiencia, resulta difícil de categorizar.

# 10 Dataset

Se puede acceder al dataset a través del enlace siguiente de Zenodo (solicitando el acceso al mismo):


http://doi.org/10.5281/zenodo.4263419

# 11 Recusos:


- Subirats, L., Calvo, M. (2018). Web Scraping. Editorial UOC.
- Lawson, R. (2015). Web Scraping with Python. Packt Publishing Ltd.
- https://selenium-python.readthedocs.io/
- https://www.selenium.dev/
- http://digital.csic.es/handle/10261/178496
- http://www.consorciomadrono.es/investigam/licencias/
- https://realpython.com/beautiful-soup-web-scraper-python/
- https://realpython.com/modern-web-automation-with-python-and-selenium/
- https://towardsdatascience.com/real-world-example-on-web-scraping-with-selenium-and-beautiful-soup-3e615dbc1fa1
- https://boredhacking.com/tor-webscraping-proxy/


# 12 Contribuciones:

| **Contribuciones** | **Firma** |
|---|---|
| Investigación previa | GMA, IAB |
| Redacción de las respuestas | GMA, IAB |
| Desarrollo código | GMA, IAB |