

# Tipología y ciclo de vida de los datos

**Alumnos: Eduardo Ranedo Martínez / Luis Piñuela Galán**

**Fecha: 15/03/2023**

**PRAC: PRAC 1**

## Índice:

1. Contexto
2. Título
3. Descripción del dataset
4. Representación gráfica
5. Contenido
6. Propietario
7. Inspiración
8. Licencia
9. Código
10. Dataset
11. Video

| Contribuciones            | Firma  |
|---------------------------|--|
| Investigación previa      | Eduardo Ranedo Martínez / Luis Piñuela Galán |
| Redacción de respuestas   | Eduardo Ranedo Martínez / Luis Piñuela Galán |
| Desarrollo de código      | Eduardo Ranedo Martínez / Luis Piñuela Galán |
| Participación en el video | Eduardo Ranedo Martínez / Luis Piñuela Galán |

## 1. Contexto

El sitio web [www.imdb.com](http://www.imdb.com) es una de las principales bases de datos en internet de películas y programas de televisión del mundo. IMDb permite la búsqueda de más de 10 millones de títulos en función de multitud de filtros; títulos, género, año de lanzamiento, país, etc.

Además, IMDb ofrece reseñas y calificaciones de películas y programas de televisión, proporcionadas tanto por usuarios registrados como por críticos profesionales. También ofrece noticias y entrevistas sobre la industria del entretenimiento.

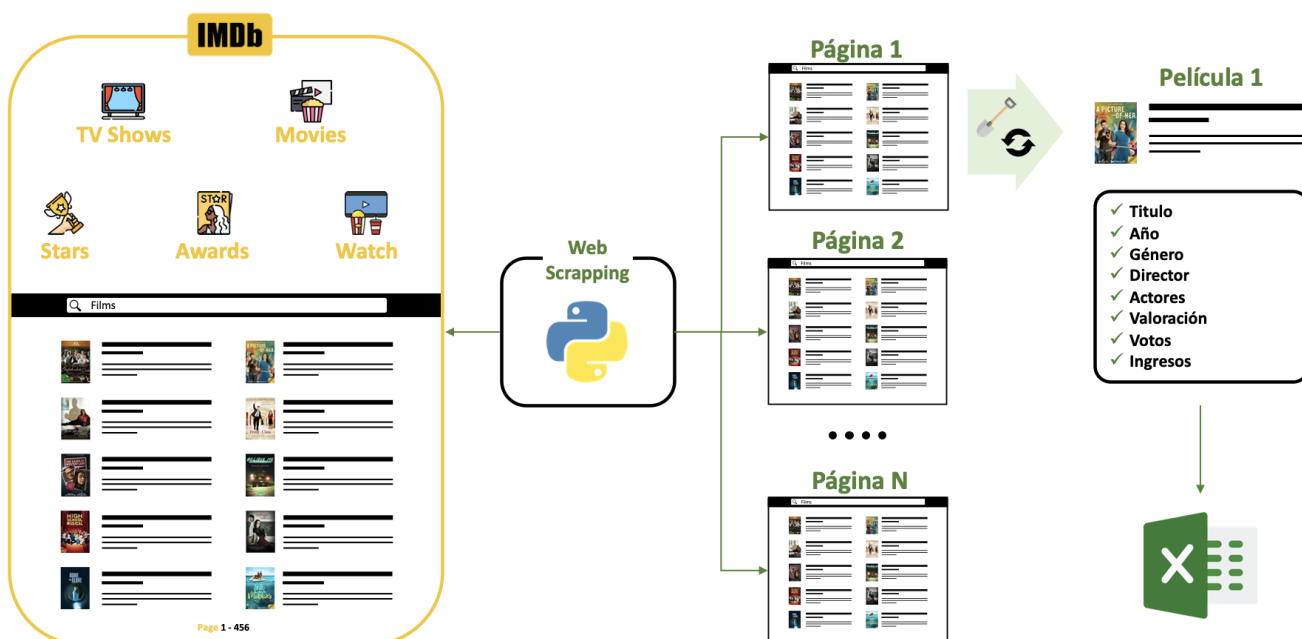
## 2. Título

La historia del cine en datos

## 3. Descripción del dataset

El dataset contiene los títulos de 420.000 largometrajes con sus principales características. La actualización de los datos es constante ya que se van incluyendo nuevas reseñas y subiendo nuevos títulos cada día.

## 4. Representación gráfica



Para explicar el dataset y el proyecto hemos usado la infografía superior para ilustrar la explicación:

- **IMDb:** parte más a la izquierda y en amarillo representa la web que se ha seleccionado para realizar el Web Scrapping, podemos fijarnos además que se ha seleccionado las películas como categoría a “minar” de la web.

- **Web Scrapping:** El script que hemos desarrollado entre los dos miembros del equipo permite acceder a esta web e iterar por las distintas páginas de películas (por defecto IMDb solo muestra máximo 250) para ir obteniendo los **div** que representa cada registro
- **Extracción y guardado:** Con cada Div se extraen 8 parámetros que se guardarán posteriormente en un fichero .csv para ser analizados más profundamente.

## 5. Contenido

Las variables elegidas para nuestro proyecto de scraping son las siguientes:

- Título: Muestra el nombre de la película en su nombre original
- Duración: Muestra en minutos la duración de la película
- Género: Indica el género cinematográfico de cada uno de los títulos, se puede encontrar que las películas tengan uno o varios géneros
- Año de publicación: Indica el año en el que se estrenó en cines o en la plataforma de streaming la película
- Valoración: Indica la media de puntuación que han dado los usuarios registrados en IMDb
- Director: Muestra el director de la película al igual que los géneros podemos encontrarnos con películas con diversos directores.
- Actores: Principales actores involucrados en la película
- Taquilla: Recaudación estimada de la película que ha hecho en cines y está en IMDb.
- 

## 6. Propietario

IMDb es una empresa que pertenece a Amazon, por tanto, la propiedad de los datos pertenece a dicha corporación. Existen dos servicios principales ofertados por la empresa: IMDb.com e IMDbpro.

El primero de ellos está más orientado a los fans, muestra el catálogo completo de títulos pero las características de exploración de datos son más limitadas. Actúa como foro y canal de noticias sobre el mundo del cine y del entretenimiento en general.

Por otro lado está IMDbpro recoge las características del primero pero las opciones de exploración son mucho más avanzadas. Este servicio es el más usado para analistas y personas de influencia de la industria.

IMDb tiene diferentes planes de pago para acceder a sus datos. Existen numerosos trabajos de analítica del cine utilizando esta base de datos, como por ejemplo *“El análisis de redes aplicado al Cine Español” (2010)* que utiliza la base datos IMDb pro para analizar la industria del cine español.

## 7. Inspiración

Hay multitud de estudios que se han realizado utilizando esta fuente de datos tal y como comentan Rafael Repiso y Rafael Marfil-Carmona en su artículo *“IMDb y su utilidad para la investigación cinematográfica: ejemplos de uso de datos desde la metodología de análisis de redes sociales” (2011)*.

En el artículo mencionan que hay 3 puntos clave en el uso de IMDB para estudios cinematográficos; por un lado la gran variedad de tipos de datos que permite realizar multitud de análisis estadísticos, por otro lado el elemento relacional de la base de datos permitiendo enlazar diferentes tipos de informaciones y un último elemento que es la estructura de la base de datos, completamente organizada y normalizada, facilitando los procesos de minería de datos.

Uno de los trabajos que realizaron los investigadores anteriores, utilizando la base de datos de IMDb, fue *“El análisis de redes aplicado al Cine Español”* (2010). En el trabajo aplican un análisis de redes sobre el cine español con el fin de analizar y visualizar la estructura social de la industria.

Los principales objetivos de la investigación son:

- Analizar las relaciones existentes entre las diferentes películas.
- Determinar los principales protagonistas de la red (actores y personal técnico).
- Representar gráficamente la estructura de la red.
- Caracterizar los actores principales dentro de la red.
- Contextualizar y visualizar las películas con otras producciones en las que han influido o que influyeron.

Para ello crearon diferentes redes, por ejemplo en una de ellas, los nodos que relacionan a las películas son los actores o actrices que han participado en dos o más películas. De manera que es posible ver el grado de intermediación, es decir, ver qué película comparte el mayor número de actores con el mayor número de películas.

A través de estas redes se pueden llegar a conclusiones como que la presencia del personal técnico en la industria supera la de los actores y actrices o que los actores secundarios tienen más presencia en la red de relaciones y, por lo tanto, tienen más posibilidades de estar en diferentes grupos de influencia en el futuro.

Otros trabajos como *“Análisis del fenómeno fan y antifan desde los datos de la IMDb”* (2018) también han explotado el potencial de esta base de datos. En este estudio utilizaron los datos de las películas, valoraciones y datos demográficos de los usuarios que votaron, para crear una métrica que caracterice las votaciones como fenómeno fan y antifan y asignar a cada película un valor global en esas escalas.

El estudio logró enseñar las discrepancias en la caracterización del fenómeno fan en función del país. Demostró que el fenómeno antifan está asociado a determinados grupos en respuesta a comportamientos del fenómeno fan. También demostró que existen diferencias en el fenómeno fan según edad, género y origen geográfico.

## 8. Licencia

Utiliza una licencia de Atribución-NoComercial-SinDerivadas (CC BY-NC-ND)

Se trata de la licencia más restrictiva de las seis licencias estandarizadas de Creative Commons que se utilizan para permitir el uso y la distribución de obras creativas y datos en línea.

La licencia permite a los usuarios descargar y compartir la obra siempre y cuando le den crédito al autor original (reconocimiento), no la usen con fines comerciales (no comercial), y no la modifiquen de ninguna manera (sin obra derivada).

## 9. Código

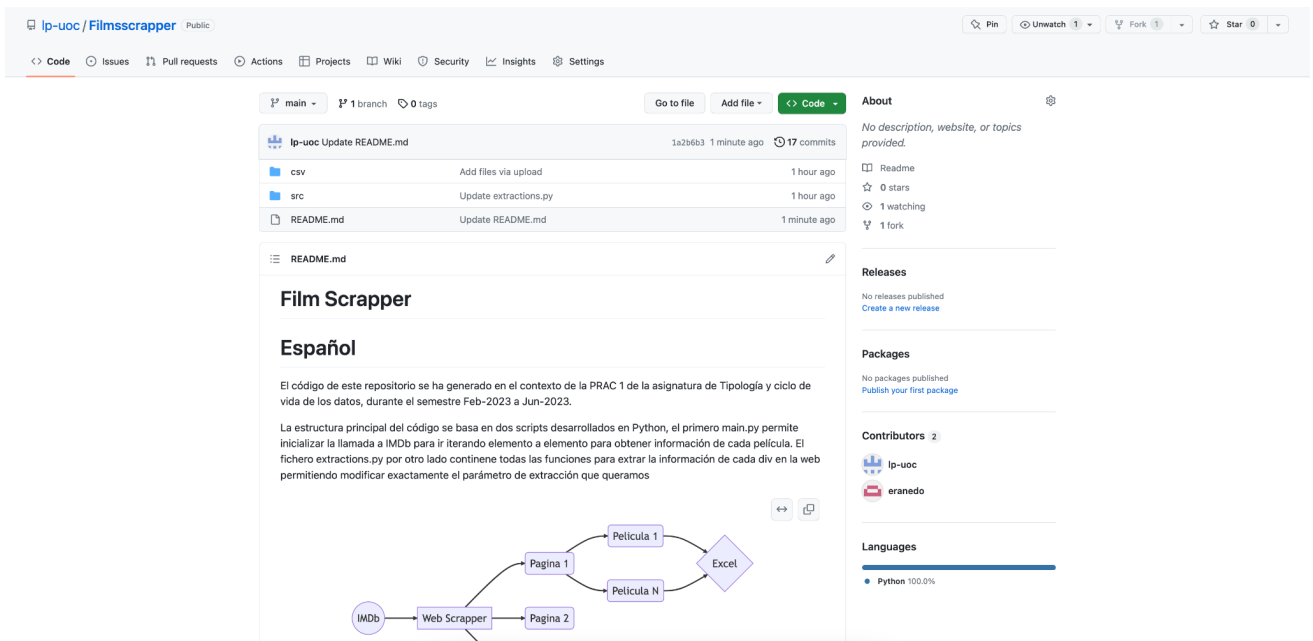
```

~-- jupyter-notebook - Python
Introduce el número de películas a obtener: 420000
Remaning web download 1680
Remaning web download 1679
Remaning web download 1678
Remaning web download 1677
Remaning web download 1676
Remaning web download 1675
Remaning web download 1674
Remaning web download 1673
Remaning web download 1672
Remaning web download 1671
Remaning web download 1670
Remaning web download 1669
Remaning web download 1668
Remaning web download 1667
Remaning web download 1666
Remaning web download 1665
Remaning web download 1664
Remaning web download 1663
Remaning web download 1662
Remaning web download 1661
Remaning web download 1660
Remaning web download 1659
Remaning web download 1658
Remaning web download 1657
Remaning web download 1656
Remaning web download 1655
Remaning web download 1654
Remaning web download 1653
Remaning web download 1652
Remaning web download 1651
Remaning web download 1650
Remaning web download 1649
Remaning web download 1648
Remaning web download 1647
Remaning web download 1646
Remaning web download 1645
Remaning web download 1644
Remaning web download 1643
Remaning web download 1642
Remaning web download 1641
Remaning web download 1640
Remaning web download 1639
Remaning web download 1638
Remaning web download 1637
Remaning web download 1636
Remaning web download 1635

```

Script en ejecución obteniendo información de IMDb

Repositorio en GitHub donde podemos encontrar el código desarrollado:  
<https://github.com/lp-uoc/Filmsscrapper>



El código se encuentra dividido en dos partes; la primera el archivo extractions.py que contiene una función por cada variable que va a contener nuestro dataset. Las funciones extraen los valores que queremos obtener del HTML.

### Función get\_url

```
def get_url(page):
    # Build the url to get

    # Get url next page
    data = str(page.find_all('a', class_="listner-page-next next-page")[0])

    # Get and Build the new url
    url = "https://www.imdb.com/" + str(re.search(r'"\/(.|\n)*"', data)[0].replace('"', ''))

    return url
```

La función `get_url` busca en la página el enlace HTML que dirige a la próxima página de la lista. Se buscan todos los elementos HTML que tienen la etiqueta 'a' y la clase 'listner-page-next next-page'.

Cuando encontramos el enlace a la siguiente página, la función extrae la URL y la concatena con "https://www.imdb.com/". Busca una cadena de caracteres que coincida con la expresión regular que representa la URL en el enlace HTML encontrado anteriormente, para así extraer la URL que sigue al carácter '/' en el enlace HTML. Luego, se elimina el carácter '"' utilizando la función 'replace' para asegurarse de que la URL resultante sea válida.

#### Función `get_title`

```
def get_title(soup, number):

    # Get the title from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        return soup[number].find('h3', class_='listner-item-header').a.string

    except:
        return 'N/A'
```

La función extrae el título de la película buscando los 'h3' con la clase 'listner-item-header' y luego buscando la etiqueta 'a' dentro de ese elemento, que contiene el texto del título de la película.

Si el título se encuentra correctamente, la función devuelve el texto del título como una cadena de caracteres. Si no lo encuentra, la función devuelve un NA.

#### Función `get_duration`

```
def get_duration(soup, number):

    # Get the duration from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        return soup[number].find('span', class_='runtime').string

    except:
        return "N/A"
```

La función busca el elemento HTML que contiene el valor que indica la duración de una película

Si la función find puede encontrar la etiqueta, entonces nos devuelve la duración como una cadena. Si la función find no la encuentra, devuelve un NA

#### Función get\_genre

```
def get_genre(soup, number):

    # Get the genre from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        return soup[number].find('span', class_='genre').string.replace("\n", "")

    except:
        return "N/A"
```

La función busca el elemento HTML que contiene el valor que indica el género de la película.

Si la función find puede encontrar la etiqueta, entonces devuelve el género y elimina los saltos de línea. Si la función find no la encuentra, devuelve un NA

#### Función get\_year

```
def get_year(soup, number):

    # Get the year from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        return soup[number].find('span', class_='lister-item-year text-muted unbold').get_text()

    except:
        return "N/A"
```

La función busca el elemento HTML que contiene el valor que indica el año de la película.

Si la función find puede encontrar la etiqueta, entonces devuelve el año. Si la función find no la encuentra, devuelve un NA

#### Función get\_rating

```
def get_rating(soup, number):

    # Get the rating from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        return soup[number].find('div', class_='inline-block ratings-imdb-rating').get_text().replace("\n", "")

    except:
        return "N/A"
```

La función busca el elemento HTML que contiene el valor que indica la puntuación de la película.

Si la función find puede encontrar la etiqueta, entonces devuelve la puntuación y elimina los saltos de línea. Si la función find no la encuentra, devuelve un NA

#### Función get\_gross



```
def get_gross(soup, number):

    # Get the gross from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        # Get all data from the div
        values = soup[number].find("p", {"class": "sort-num_votes-visible"})

        #Extract with regex
        gross = re.search(r'Gross.*\n.* name=' ,str(values))[0]

        # Save the value
        if gross != None:
            return re.search(r'\\"([0-9]|,){2,}\\\"' ,str(gross))[0]
        else:
            return 'N/A'

    except:
        return"N/A"
```

La función busca el elemento HTML que contiene el valor que indica el importe de las ventas de la película.

Si la función puede encontrar la etiqueta, entonces devuelve las ventas. Como el elemento HTML es compartido por otros elementos de la página, utilizamos la expresión regular para coger el formato específico de las ventas.

### Función get\_votes

```
def get_votes(soup, number):

    # Get the votes from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        # Get all data from the div
        values = soup[number].find("p", {"class": "sort-num_votes-visible"})

        #Extract with regex
        votes = re.search(r'Votes.*\n.* name=' ,str(values))[0]

        # Save the value
        if votes != None:
            return votes[0]
        else:
            return 'N/A'

    except:
        return"N/A"
```

La función busca el elemento HTML que contiene el valor que indica el número de votos de la película en IMDb.

Si la función puede encontrar la etiqueta, entonces devuelve el número de votos. Como el elemento HTML es compartido por otros elementos de la página, utilizamos la expresión regular para coger el formato específico de número de votos

### Función get\_directors

```
def get_directors(soup, number):

    # Get the director o directors from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        # Get all data from the div
        values = soup[number].find_all('p', class_="")

        #Extract with regex
        directors = re.search(r'Director((.\n*)\>\<',str(values))[0]

        # Save the value
        if directors != None:
            return re.findall(r'\>.*\<',directors)
        else:
            return 'N/A'

    except:
        return "N/A"
```

La función busca el elemento HTML que contiene el valor que indica los directores de la película.

Si la función puede encontrar la etiqueta, entonces devuelve a los directores. Como el elemento HTML es compartido por otros elementos de la página, utilizamos la expresión regular para sacar a los directores.

### Función get\_actors

```
def get_actors(soup, number):

    # Get the actors from the text and with the number of element
    # text: soup with the list of divs with movie information
    # number: nº of element to check

    try:
        # Get all data from the div
        values = soup[number].find_all('p', class_="")

        #Extract with regex
        actors = re.search(r'Star((.|\\n)*\\</p\\>\\)', str(values))[0]

        # Save the value
        if actors != None:
            return re.findall(r'\\>.*\\<', actors)
        else:
            return 'N/A'

    except:
        return "N/A"
```

La función busca el elemento HTML que contiene el valor que indica los actores de la película.

Si la función puede encontrar la etiqueta, entonces devuelve a los actores. Como el elemento HTML es compartido por otros elementos de la página, utilizamos la expresión regular para sacar a los actores.

Una vez definidas todas las funciones, creamos el script main.py que se encarga de ejecutar todos los pasos hasta conseguir extraer todos los valores de la web y almacenarlos en un csv.

El script main.py contiene el siguiente código:

```
# Import the Extractions the logic
import extractions

# Import the package
import pandas as pd
import time
from bs4 import BeautifulSoup
import requests

# Get number of films
n_films = int(input("Introduce el número de películas a obtener: "))
film_range = range(1,n_films,250)

# Create the lists
titles = []
duration = []
genre = []
year = []
rating = []
gross = []
votes = []
directors = []
actors = []

dur = len(film_range)

# Just initialice the url
url_to_get = "https://www.imdb.com/search/title/?title_type=feature&release_date=,2022-12-31&sort=release_date,desc&count=250&start=0"

# Get the slice of films
for n in film_range:

    #Map the html to beautiful soup
    page = BeautifulSoup(requests.get(url_to_get).content,features="html.parser")

    # Freeze the code
    time.sleep(4)

    # Get data from the file
    soup = page.find_all('div', class_='lister-item mode-advanced')

    for e in range(250):
        #Get title
        titles.append(extractions.get_title(soup, e))
        #Get duration
        genre.append(extractions.get_genre(soup, e))
        #Get genre
        duration.append(extractions.get_duration(soup, e))
        #Get year
        year.append(extractions.get_year(soup,e))
        #Get rating
        rating.append(extractions.get_rating(soup,e))
        #Get Gross
        gross.append(extractions.get_gross(soup,e))
        #Get votes
        votes.append(extractions.get_votes(soup,e))
        #Get Director/Directors
        directors.append(extractions.get_directors(soup,e))
        #Get Actors
        actors.append(extractions.get_actors(soup,e))
```

```
# Get new url
url_to_get = extractions.get_url(page)

# Show the status
print("Remaning web download", dur)
dur = dur - 1

# Join all data in dataset
data = pd.DataFrame(list(zip(titles,
                             duration,
                             genre,
                             year,
                             rating,
                             gross,
                             directors,
                             actors)),
                    columns=['titulo',
                             'duracion',
                             'genero',
                             'year',
                             'rating',
                             'gross',
                             'directors',
                             'actors'])

data.to_csv("datos.csv", index=False, sep=";")
```

Creamos una lista para cada una de las variables que tendrá nuestro dataset (título, duración, género, año, calificación, ingresos brutos, votos, directores y actores).

La variable "dur" contiene el número de iteraciones que se realizarán en el bucle.

La variable "url\_to\_get" contiene la URL de la página web de IMDb que se utilizará para extraer la información.

Realizamos un bucle for que recorre una serie de rangos de películas (cada página de IMDb muestra 250 películas) y para cada rango, se envía una solicitud HTTP a la URL correspondiente para obtener la página web de IMDb. Se ha añadido una pausa de 4 segundos entre solicitud y solicitud para evitar saturar el servidor y que nos puedan denegar el acceso.

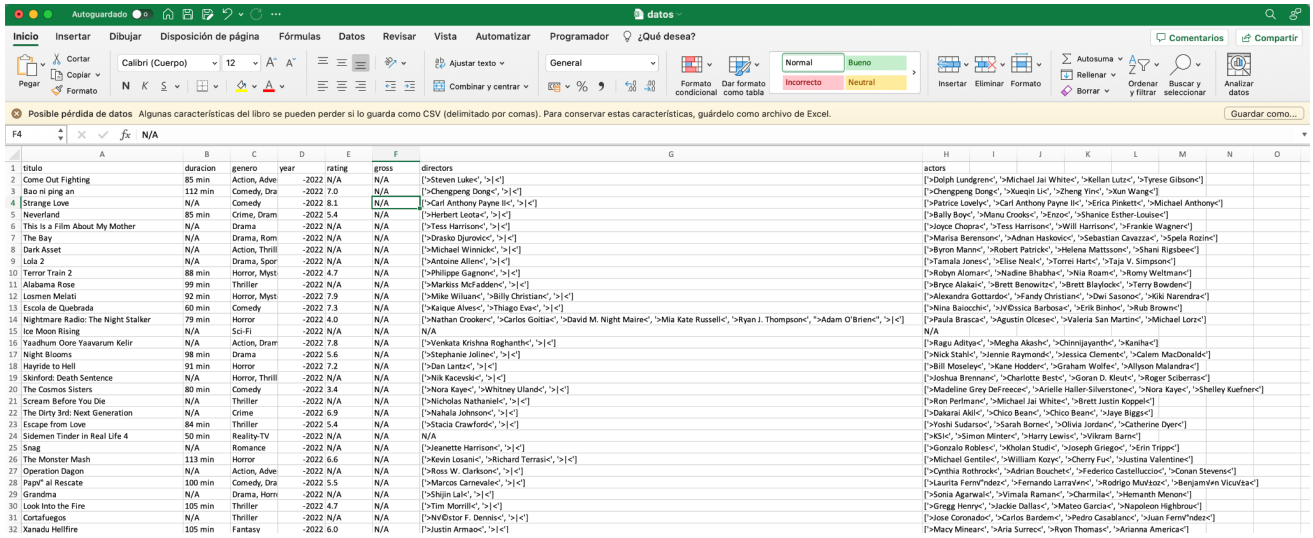
Para cada película en la página web, se llama a las funciones personalizadas del módulo "extractions" para extraer la información relevante (título, duración, género, año, calificación, ingresos brutos, votos, directores y actores) y se agrega a la lista correspondiente.

Después de haber procesado todas las películas en la página web actual, se llama a la función "get\_url" del módulo "extractions" para encontrar la URL de la página web siguiente.

Se repite el proceso para cada rango de películas.

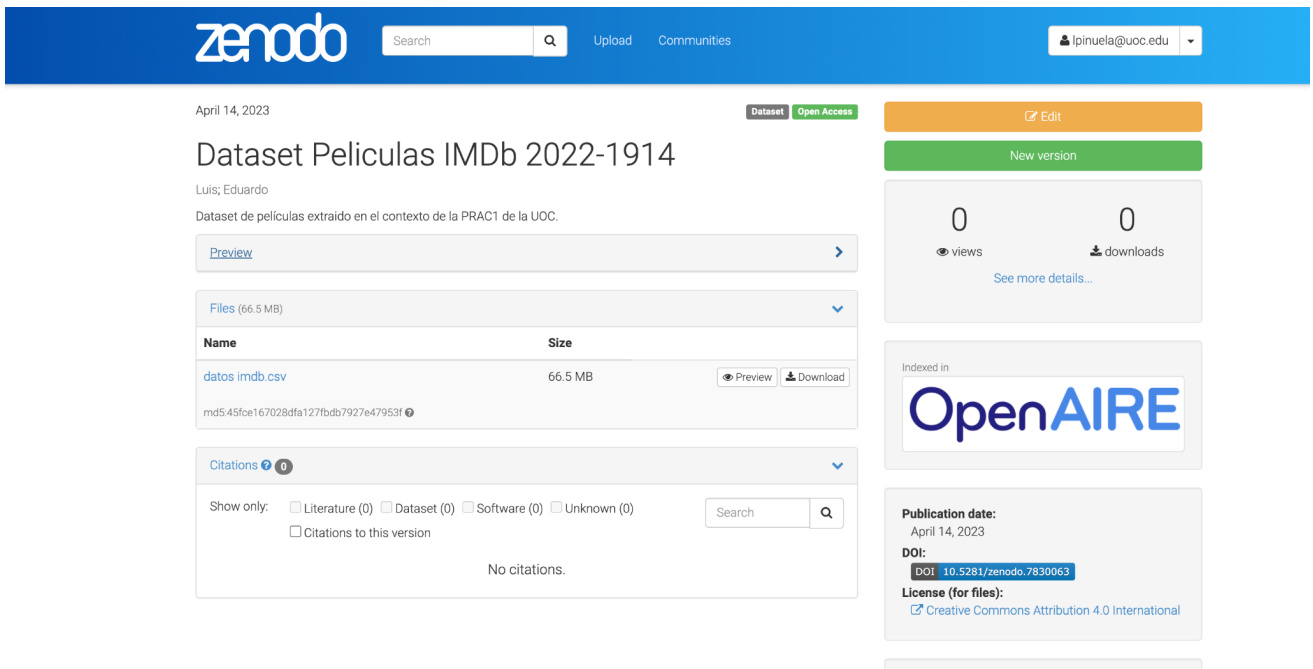
Finalmente, se crea un DataFrame de Pandas a partir de las listas de cada variable y se guarda en un archivo CSV.

## 10. Dataset



|    | A                                  | B        | C              | D     | E      | F     | G                                  | H                                | I | J | K | L | M | N | O |
|----|------------------------------------|----------|----------------|-------|--------|-------|------------------------------------|----------------------------------|---|---|---|---|---|---|---|
| 1  | titulo                             | duration | genero         | year  | rating | gross | directors                          | actors                           |   |   |   |   |   |   |   |
| 2  | Come Out Fighting                  | 85 min   | Action, Adve   | -2022 | N/A    | N/A   | [>Steven Luke<,> > <]              | [>Dolph Lundgren<,> > <]         |   |   |   |   |   |   |   |
| 3  | Bao in ping an                     | 112 min  | Comedy, Dra    | -2022 | 7.0    | N/A   | [>Chengpeng Dong<,> > <]           | [>Chengpeng Dong<,> > <]         |   |   |   |   |   |   |   |
| 4  | Strange Love                       | N/A      | Comedy         | -2022 | 8.1    | N/A   | [>Carl Anthony Payne II<,> > <]    | [>Patrice Levely<,> > <]         |   |   |   |   |   |   |   |
| 5  | Neverland                          | 85 min   | Crime, Dram    | -2022 | 5.4    | N/A   | [>Herbert Loefer<,> > <]           | [>Bally Boy<,> > <]              |   |   |   |   |   |   |   |
| 6  | This is a Film About My Mother     | N/A      | Drama          | -2022 | N/A    | N/A   | [>Tess Harrison<,> > <]            | [>Joyce Chopra<,> > <]           |   |   |   |   |   |   |   |
| 7  | The Bay                            | N/A      | Drama, Rom     | -2022 | N/A    | N/A   | [>Drasko Djurdjevic<,> > <]        | [>Marina Benenson<,> > <]        |   |   |   |   |   |   |   |
| 8  | Dark Asset                         | N/A      | Action, Thrill | -2022 | N/A    | N/A   | [>Michael Wicinski<,> > <]         | [>Byron Mann<,> > <]             |   |   |   |   |   |   |   |
| 9  | Lola 2                             | N/A      | Drama, Spor    | -2022 | N/A    | N/A   | [>Antoine Allene<,> > <]           | [>Tamala Jones<,> > <]           |   |   |   |   |   |   |   |
| 10 | Terror Train 2                     | 88 min   | Horror, Myst   | -2022 | 4.7    | N/A   | [>Philippe Gagnon<,> > <]          | [>Robyn Alomari<,> > <]          |   |   |   |   |   |   |   |
| 11 | Alabama Rose                       | 99 min   | Thriller       | -2022 | N/A    | N/A   | [>Markus McFadden<,> > <]          | [>Mryso Alakai<,> > <]           |   |   |   |   |   |   |   |
| 12 | Losmen Melati                      | 92 min   | Horror, Myst   | -2022 | 7.9    | N/A   | [>Mike Willaun<,> > <]             | [>Alexandra Gattardo<,> > <]     |   |   |   |   |   |   |   |
| 13 | Escola de Quebrada                 | 60 min   | Comedy         | -2022 | 7.3    | N/A   | [>Kaique Alves<,> > <]             | [>Nina Balocchi<,> > <]          |   |   |   |   |   |   |   |
| 14 | Nightmare Radio: The Night Stalker | 79 min   | Horror         | -2022 | 4.0    | N/A   | [>Nathan Crooker<,> > <]           | [>Paula Brasca<,> > <]           |   |   |   |   |   |   |   |
| 15 | Ice Moon Rising                    | N/A      | Sci-Fi         | -2022 | N/A    | N/A   | [>Venkata Krishna Roghanth<,> > <] | [>Ragu Adityan<,> > <]           |   |   |   |   |   |   |   |
| 16 | Yaadum Dore Yaavarum Kellir        | N/A      | Action, Dram   | -2022 | 7.8    | N/A   | [>Stephanie Jolinet<,> > <]        | [>Nick Stahl<,> > <]             |   |   |   |   |   |   |   |
| 17 | Night Blooms                       | 98 min   | Drama          | -2022 | 5.6    | N/A   | [>Dan Latzer<,> > <]               | [>Bill Moseley<,> > <]           |   |   |   |   |   |   |   |
| 18 | Wayide to Hell                     | 91 min   | Horror         | -2022 | 7.2    | N/A   | [>Nora Kaye<,> > <]                | [>Joshua Brennan<,> > <]         |   |   |   |   |   |   |   |
| 19 | Skinford: Death Sentence           | N/A      | Horror, Thrill | -2022 | N/A    | N/A   | [>Nik Kaciveli<,> > <]             | [>Madeline Grey DeFreese<,> > <] |   |   |   |   |   |   |   |
| 20 | The Cosmos Sisters                 | 80 min   | Comedy         | -2022 | 3.4    | N/A   | [>Nicholas Nathaniel<,> > <]       | [>Ron Perlmann<,> > <]           |   |   |   |   |   |   |   |
| 21 | Scream Before You Die              | N/A      | Thriller       | -2022 | N/A    | N/A   | [>Natalia Johnson<,> > <]          | [>Dakari Adin<,> > <]            |   |   |   |   |   |   |   |
| 22 | The Dirty 3rd: Next Generation     | N/A      | Crime          | -2022 | 6.9    | N/A   | [>Stacia Crawford<,> > <]          | [>Yoshi Sudarso<,> > <]          |   |   |   |   |   |   |   |
| 23 | Escape from Love                   | 84 min   | Thriller       | -2022 | 5.4    | N/A   | N/A                                | [>KSi<,> > <]                    |   |   |   |   |   |   |   |
| 24 | Sidemen Tinder in Real Life 4      | 50 min   | Reality-TV     | -2022 | N/A    | N/A   | [>Jeanette Harrison<,> > <]        | [>Gonzalo Robles<,> > <]         |   |   |   |   |   |   |   |
| 25 | Snag                               | N/A      | Romance        | -2022 | N/A    | N/A   | [>Kevin Lozano<,> > <]             | [>Michael Gentile<,> > <]        |   |   |   |   |   |   |   |
| 26 | The Monster Mash                   | 113 min  | Horror         | -2022 | 6.6    | N/A   | [>Ross W. Clarkon<,> > <]          | [>Cynthia Rothrock<,> > <]       |   |   |   |   |   |   |   |
| 27 | Operation Dagon                    | N/A      | Action, Adve   | -2022 | N/A    | N/A   | [>Marcos Carnevalle<,> > <]        | [>Laurea Fernndez<,> > <]        |   |   |   |   |   |   |   |
| 28 | Papn' al Rescate                   | 100 min  | Comedy, Dra    | -2022 | 5.5    | N/A   | [>Shijin Loh<,> > <]               | [>Xonia Agarwal<,> > <]          |   |   |   |   |   |   |   |
| 29 | Grandma                            | N/A      | Drama, Rom     | -2022 | N/A    | N/A   | [>Tim Merrill<,> > <]              | [>Gregg Henry<,> > <]            |   |   |   |   |   |   |   |
| 30 | Look Into the Fire                 | 105 min  | Thriller       | -2022 | 4.7    | N/A   | [>NVDor F. Dennis<,> > <]          | [>Jose Coronado<,> > <]          |   |   |   |   |   |   |   |
| 31 | Cortagijos                         | N/A      | Thriller       | -2022 | N/A    | N/A   | [>Justin Amaec<,> > <]             | [>Macy Minear<,> > <]            |   |   |   |   |   |   |   |
| 32 | Xanadu Hellfire                    | 105 min  | Fantasy        | -2022 | 6.0    | N/A   |                                    | [>Aria Suneer<,> > <]            |   |   |   |   |   |   |   |

Ejemplo de dataset en fichero excel recién extraído.



April 14, 2023

**Dataset** **Open Access**

**Dataset Peliculas IMDb 2022-1914**

Luis; Eduardo

Dataset de películas extraído en el contexto de la PRAC1 de la UOC.

[Preview](#)

**Files** (66.5 MB)

| Name           | Size    |
|----------------|---------|
| datos imdb.csv | 66.5 MB |

[md5:45f0e167028dfa1271bdb7927e47953f](#)

**Citations** 0

Show only: ☐ Literature (0) ☐ Dataset (0) ☐ Software (0) ☐ Unknown (0)

☐ Citations to this version

No citations.

**Publication date:** April 14, 2023

**DOI:** [10.5281/zenodo.7830063](https://doi.org/10.5281/zenodo.7830063)

**License (for files):** [Creative Commons Attribution 4.0 International](#)

Información subida a Zenodo

URL en Zenodo: <https://zenodo.org/record/7830063#.ZDnLfluxBy70>

## 11. Video

URL en Drive:

[https://drive.google.com/drive/folders/1Rn6zJEdkZEEAGVbFZxPzNU6nqzbZD6iW?usp=share\\_link](https://drive.google.com/drive/folders/1Rn6zJEdkZEEAGVbFZxPzNU6nqzbZD6iW?usp=share_link)