

## IBEX 35: Valores de sus componentes

Realizado por: Leonor Pallares Ortega y Alberto Suárez Rodríguez.



### Descripción breve

Web Scraping: Obtención de Valores de componentes IBEX35

## Índice

<b>CONTEXTO</b> .....	3
<b>DATASET</b> .....	3
<b>PROPIETARIO</b> .....	7
<b>INSPIRACIÓN</b> .....	8
<b>LICENCIA</b> .....	8
<b>CÓDIGO</b> .....	9
<b>PUBLICACION del DATASET</b> .....	16
<b>VIDEO</b> .....	18

## CONTEXTO

Los datos han sido extraídos de la web [www.investing.com](http://www.investing.com). Hemos elegido esta web porque es una plataforma de mercados financieros que proporciona datos en tiempo real, cotizaciones, gráficos, herramientas financieras, noticias de última hora y análisis de 250 mercados del mundo a través de sus 44 ediciones internacionales. Con más de 21 millones de usuarios mensuales y más de 180 millones de sesiones, [investing.com](http://investing.com) es una de las tres mejores webs financieras del mundo según SimilarWeb (se especializa en análisis web, tráfico web y rendimiento) y Alexa. Los datos recogidos se centran en el mercado de índices, principalmente en el IBEX35 (índice de referencia en España) y sus componentes en el periodo\* comprendido desde hoy (fecha actual de ejecución del web scraping) hasta -1 mes.

## DATASET.

**Definición del título:** El título escogido para el dataset extraído mediante web scraping en esta primera práctica de la asignatura es: “Ibex35: Valores de sus componentes”.

**Descripción:** El conjunto de datos que se extraerá en esta práctica se compone de una serie de cotizaciones diarias del conjunto de empresas con más liquidez en las cuatro bolsas españolas (Madrid, Barcelona, Bilbao y Valencia) y de los datos más significativos de cada una de ellas.

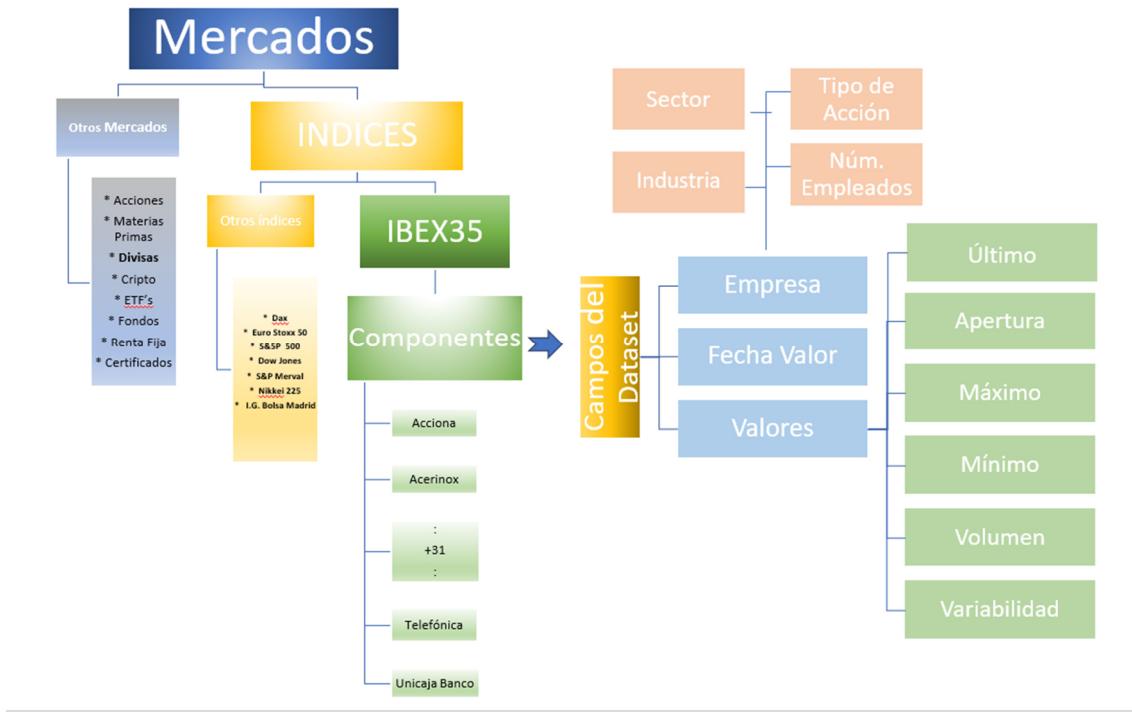
### Representación Gráfica:

Para representar gráficamente el dataset se ha construido un esquema en el que se identifica cada uno de los campos.

En esta representación se observarán dos tipologías claramente definidas:

- Variables numéricas: Los propios valores que tiene un índice a lo largo del día
- Variables categóricas: Es información relativa a la empresa con poca variabilidad en el tiempo.

A continuación, se adjunta el esquema comentado:



### Contenido:

El dataset extraído contiene 12 variables. A continuación, se detallará el contenido de cada uno de ellos.

**Empresa:** Corresponde a los nombres de empresa que componen el IBEX 35

**Fecha:** Fecha en la que se recogen los datos de la empresa.

**Último:** El precio de cierre de las acciones de la empresa en la fecha especificada.

**Apertura:** El precio de apertura de las acciones de la empresa en la fecha especificada.

**Máximo:** El precio más alto al que llegaron las acciones de la empresa en la fecha especificada.

**Mínimo:** El precio más bajo al que llegaron las acciones de la empresa en la fecha especificada.

**Volumen:** La cantidad total de acciones de la empresa que se negociaron en la fecha especificada.

**Variabilidad:** La variación porcentual entre el precio de cierre y el precio de apertura de las acciones.

**Industria:** Actividad a la que se dedica la empresa. (Variable categórica)

**Sector:** División a la que se dedica la empresa. (Variable categórica)

**Empleados:** Número de empleados de cada empresa

**Tipo de Acción:** Clases de acciones (ordinarias, privilegiadas, sin voto..)

Por otro lado, los datos de los valores están comprendidos en el período de 1 mes .

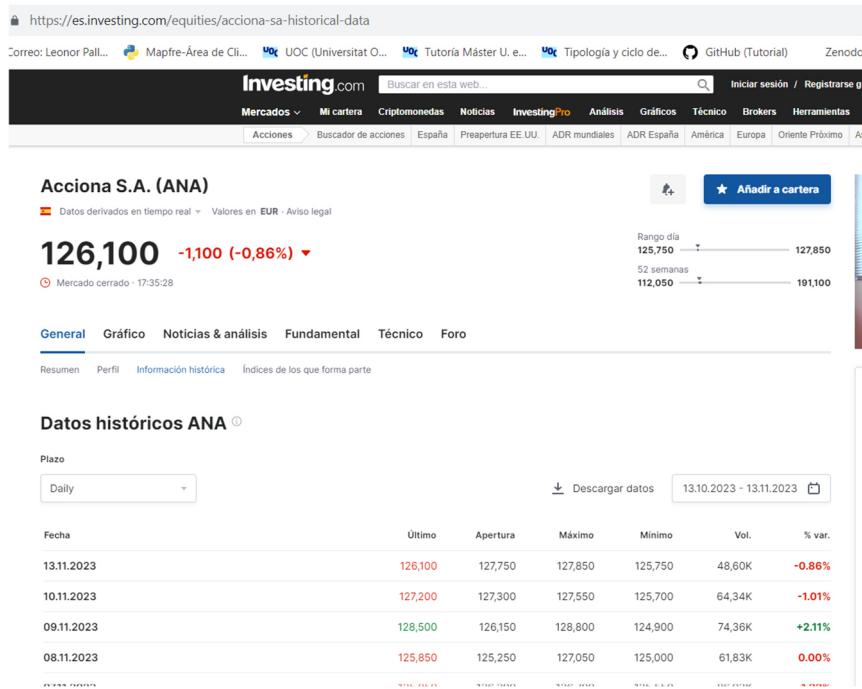
Finalmente, se resumirá el procedimiento seguido para la recogida de la información (en el apartado **CODIGO** se entrará en detalle en cada una de las partes del código utilizado):

- En primer lugar, destacaremos que la página utilizada está dispuesta de forma que accediendo a IBEX35/componentes se pueden obtener todos los links de las empresas y por cada una de ellas tendremos otros links que nos llevará al histórico de datos o a los links de profile de donde extraeremos la información que necesitamos.
- El web scraping realizado en esta práctica para obtener los datos del ibex35 se basa en dos iteraciones. En primer lugar se itera sobre los datos históricos de las acciones de cada uno de los componentes del IBEX35 y más tarde una iteración para entrar en la parte de los profile para obtener los datos categóricos.
- A continuación se adjunta una captura de pantalla de la disposición de la página desde la cual se ha extraído la información para poder entender mejor la metodología seguida:

Primero accedemos a la pagina de componentes de IBEX35, donde se muestra el listado de las 35 empresas

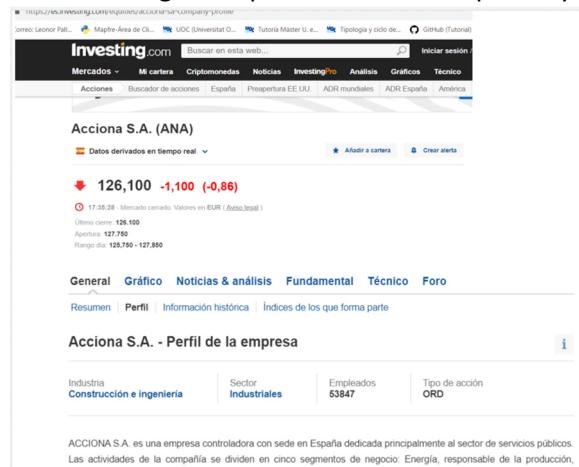
Nombre	Último	Máximo	Mínimo	Var.	% Var.	Vol.	Hora
Acciona	126,100	127,850	125,750	-1,100	-0,86%	48,60K	17:35:28
Acerinox	9,620	9,668	9,586	+0,008	+0,08%	422,94K	17:35:28
ACS	33,780	33,930	33,330	+0,520	+1,56%	263,85K	17:35:28
Aena	147,95	148,40	147,05	+0,65	+0,44%	108,92K	17:35:28
Amadeus	60,560	61,000	59,920	+0,300	+0,50%	433,81K	17:35:28
ArcelorMittal	20,510	20,650	20,280	+0,225	+1,11%	249,63K	17:35:28
Banco de Sabadell	1,2845	1,2890	1,2360	+0,0545	+4,43%	40,95M	17:35:28
Bankinter	6,264	6,272	6,122	+0,182	+2,99%	2,57M	17:35:28
BBVA	8,004	8,024	7,898	+0,104	+1,32%	12,74M	17:35:28
CaixaBank	3,930	3,934	3,865	+0,071	+1,84%	10,46M	17:35:28
Cellnex Telecom	30,98	31,16	30,30	+0,55	+1,81%	1,21M	17:35:28
Colonial	5,500	5,560	5,460	+0,040	+0,73%	934,81K	17:35:28
Corporacion Acciona ...	26,90	27,06	26,26	+0,36	+1,36%	216,37K	17:35:28
Enagás	15,835	15,875	15,740	0,000	0,00%	778,63K	17:35:28

Primero luego podemos pulsar a histórico y presentamos la pagina



Por cada una de las empresas accedemos al histórico, tal y como se muestra en la captura de pantalla se puede observar que la información diaria durante 1 mes. Por lo tanto, para acceder a ellas se ha realizado el comando “`find_all('table')`” de Python con el fin de obtener una lista con la información de cada una de las tablas. Despues de esto se ha recorrido cada uno de los elementos de la lista comentada (es decir, cada uno de los días) y se ha extraido la información que hay dentro.

Para las categóricas pulsamos el botón profile y presentamos la pagina



- Después de obtener la información de históricos en una lista y la de profile en otra, se crean los archivos csv que conforman los dataset con la sentencia “`csv.writer`” de Python.

## PROPIETARIO:

El propietario de la web es *FUSION MEDIA LIMITED*, investing.com es una plataforma de mercados financieros que proporciona datos en tiempo real, cotizaciones, gráficos, herramientas financieras, noticias de última hora y análisis de 250 mercados del mundo a través de sus 44 ediciones internacionales. Hemos analizado <https://www.investing.com/robots.txt> y no presenta ninguna restricción al scraping de datos que hemos realizado.

A continuación, se presenta un análisis de datos sobre IBEX35 encontrado en Internet para mostrar que tipo de análisis se pueden desarrollar con esta temática de datos.

- Se ha encontrado un ejemplo, en el expansión <https://www.expansion.com> Donde a partir de los valores obtenidos, igual que en nuestra práctica, se realizan diferentes estudios que generan un análisis técnico, un análisis de riesgos etc..

Por ejemplo:

En la pestaña valores refleja los datos marcados en la sesión del día.

Valores										
Análisis Fund.										
Ratios										
Análisis Técn.										
Riesgo										
Introduzca valor										
Valor	Último	Var. %	Var.	Ac.% año	Máx.	Mín.	Vol.	Capit.	Hora	
ACCIONA	125,850	0,00	0,000	-24,56	127,05	125,00	61.829	6.904	08/11	
ACCIONA ENER	26,840	-0,96	-0,260	-24,09	27,16	26,72	197.490	8.837	08/11	
ACERINOX	9,666	0,21	0,020	11,02	9,76	9,61	681.105	2.410	08/11	
ACS	33,260	0,24	0,080	32,72	33,30	32,93	759.181	9.252	08/11	
AENA	148,200	0,47	0,700	30,39	149,40	144,40	157.015	22.230	08/11	
AMADEUS IT GROUP	58,800	-1,04	-0,620	22,48	60,00	58,34	868.481	26.489	08/11	
ARCELORMITTAL	21,300	-0,61	-0,130	-12,78	21,49	21,25	153.943	18.165	08/11	

En la pestaña Análisis técnico:

En función de los valores obtenidos y con las comparaciones realizadas en el análisis, recomiendan que hacer con las acciones a corto plazo, que tendencia siguen las acciones de cada compañía perteneciente al IBEX 35

Valores	Análisis Fund.	Ratios	Análisis Técn.	Riesgo		
Introduzca valor <input type="text"/>						
Valor	Recomendación Técnica a corto plazo	Primera Resistencia	Primer Soporte	Tendencia a C/P	Tendencia a M/P	Tendencia a L/P
ACCIONA	Mantener	129,35	123,73			
ACCIONA ENER	Acumular	27,77	26,38			
ACERINOX	Acumular	9,99	9,42			
ACS	Mantener	33,53	32,51			
AENA	Acumular	152,42	143,53			
AMADEUS IT GROUP	Vender	59,95	55,19			
ARCELORMITTAL	Vender	23,69	20,65			
BANCO SABADELL	Acumular	1,24	1,1			

Dicha información se puede consultar en la web siguiente:

[https://www.expansion.com/mercados/cotizaciones/indices/ibex35\\_IIB.html](https://www.expansion.com/mercados/cotizaciones/indices/ibex35_IIB.html)

El ejemplo propuesto se asemeja debido a que estudia las mismas variables a partir de también de los valores publicados en los mercados y puede servir como referencia.

## INSPIRACIÓN:

El dataset obtenido puede usarse para distintos análisis. En primer lugar, para hacer regresiones e intentar predecir futuros valores lo cual es útil para inversores. También para estudiar los comportamientos del mercado y optimizar carteras de inversión. Otra opción es la evaluación de riesgos estudiando la volatilidad de cada empresa.

Esas serían algunas de las posibles opciones. Además, se podría enriquecer el dataset con otras variables como, por ejemplo, el sector económico de las empresas, o tamaño de las empresas, para realizar otros estudios de interés como analizar la evolución de un sector económico en concreto.

Se considera que se puede comparar con el ejemplo mencionado en el ejercicio anterior (propietario) por las razones comentadas, la similitud de las variables y la idea de analizar los valores con el fin de ayudar a posibles inversores, etc.

## LICENCIA:

La licencia escogida es **Released Under CC BY-NC-SA 4.0 License (Atribución-NoComercial-CompartirlGual 4.0 Internacional)**.

El motivo por el cual usaremos esta licencia es para permitir que otros usuarios puedas usar el dataset, modificarlo, copiarlo y compartirlo, siempre que se otorgue atribución a los autores, y además tener control sobre su uso comercial. Finalmente, destacar que en ningún momento hemos tenido la intención de publicar el dataset de forma restringida ya que no contiene ningún tipo de información confidencial y, por lo tanto, puede servir de ayuda a cualquier futuro análisis que se pueda realizar

## CÓDIGO:

En este ejercicio se adjuntará el código de Python con el que se ha generado los dos ficheros .csv (datos\_ibex35\_dataset.csv y datos\_categoricos\_dataset.csv ) que conformarán nuestro dataset, se adjunta mediante imágenes separadas para poder explicar más fácilmente el código ya que este en una sola captura de pantalla no cabe:

En el código que se adjunta podemos distinguir tres partes claramente seccionadas: la importación de las librerías necesarias, la definición de las funciones que extraen la información de la página web escogida y finalmente un proceso principal donde se invocan a las funciones que crearán los archivos csv que contendrán los datasets obtenidos.

A continuación, se detallarán los procedimientos seguidos en cada una de estas partes:

- Importación de las librerías necesarias

```
1  ### IMPORTACIÓN DE LAS LIBRERÍAS NECESARIAS PARA EJECUTAR NUESTRO CÓDIGO
2  import os                               #Para Funcionalidades de sistema operativo (directorios etc)
3  import requests
4  import pandas as pd
5  import csv
6  from bs4 import BeautifulSoup
7  from selenium import webdriver
8  from selenium.webdriver.common.by import By
9  from datetime import datetime
10 #from datetime import timedelta
11 from dateutil.relativedelta import *    #Para tratamiento de fechas
12 import warnings
13 warnings.filterwarnings('ignore')      #Desactivamos los warnings de tipo obsoleto
```

En esta parte del código se han importado las librerías “datetime”, “timedelta” y “relativedelta” para trabajar con fechas, las librerías “requests”, “BeautifulSoup” y selenium para realizar el web scraping, la librería “pandas” especializada para trabajar con estructuras de datos, la librería “csv” para crear los archivos .csv que contendrán los datasets obtenidos, y la librería “os” para trabajar con las rutas de carpetas del pc.

- Definición de funciones:

En esta segunda parte del código se definen las funciones a utilizar para realizar el web scraping

## ⇒ Definición: Función “SeleniumURL”

```
20  ## Definicion de la funcion mediante la cual se ha obtener las URLs de empresas IBEX35 a traves de SELENIUM
21 def SeleniumURL(url_inicial):
22     driver = webdriver.Chrome()          #Inicializamos el driver
23     driver.get(url_inicial)            #Accedemos a la url especificada en param. de entrada
24
25     # Aceptar el aviso de cookies si es necesario
26     try:
27         xpath_cookie = '//*[@id="onetrust-accept-btn-handler"]'
28         cookies_popup = driver.find_element(By.XPATH, xpath_cookie)
29         cookies_popup.click()
30     except:
31         pass
32
33     # Encontrar y guardar en una lista las URLs que contienen "equities" en href y "CFD" en el título
34     enlaces = driver.find_elements(By.TAG_NAME, "a")
35     enlaces_filtrados = []
36
37     for enlace in enlaces:
38         href = enlace.get_attribute("href")
39         title = enlace.get_attribute("title")
40         if href is not None and "equities" in href and "CFD" in title:
41             enlaces_filtrados.append(href)
42
43     # Crear variables para almacenar la lista de URLs filtradas y elimina duplicados
44     lista_de_urls = list(set(enlaces_filtrados))
45     lista_de_urls2 = []
46     lista_de_urls_categoricas_2 = []
47
48     # Iterar sobre las URLs filtradas y buscar elementos "a" con "historical" en la URL for url in enlaces_filtrados:
49     for url in set(enlaces_filtrados):
50         driver.get(url) # Abrir la URL
51
52         # Confeccionamos lista URL_hist: Obtener elementos "a" con "historical" en el contexto de la página actual
53         elementos_a = driver.find_elements(By.XPATH, '//a[contains(@href, "historical")]')
54         for elemento in elementos_a:
55             lista_de_urls2.append(elemento.get_attribute("href"))
56
57         # Confeccionamos lista URL_Categoricas: Obtener elementos "a" con "profile" en el contexto de la página actual
58         elementos_b = driver.find_elements(By.XPATH, '//a[contains(@href, "profile")]')
59         for elementob in elementos_b:
60             lista_de_urls_categoricas_2.append(elementob.get_attribute("href"))
61
62     # Eliminamos posibles duplicados y ordenamos las listas con las URLs necesarias para obtener los datos
63     URL_hist = list(set(lista_de_urls2))
64     URL_hist.sort()
65     URL_categoricas = list(set(lista_de_urls_categoricas_2))
66     URL_categoricas.sort()
67
68     # Cerrar el navegador
69     driver.quit()
70
71     return URL_hist, URL_categoricas
72
```

La función “SeleniumURL” tiene como parámetro de entrada “url\_inicial”, y retorna 2 listas de URL’s, una para acceder a los datos históricos (URL\_hist) y otra para acceder a los datos de las variables categóricas (URL\_categoricas) de cada una de las 35 empresas que componen el IBEX35. Esta función nos ayudará a automatizar el navegador, Chrome en nuestro caso, como si de una “marioneta” se tratara y para ello es necesario disponer de la variable driver que nos permitirá interactuar con el navegador y los sitios web.

Los puntos de esta función son:

- Inicialización del driver, en nuestro caso con webdriver.chrome(), y procedemos a navegar por la web accediendo a la url\_inicial utilizando el método .get().
- Implementación de la aceptación de avisos de cookies para poder seguir navegando configurando el xpath\_cookie y haciendo clic en el botón correspondiente utilizando la función.click().

- Procederemos a localizar los elementos de tag “a” dentro del DOM, utilizando el acceso por nombre de etiqueta a través de la función `find_elements_by_tag_name()`, localizador `By.TAG_NAME`.
- Creamos una lista vacía llamada “enlaces\_filtrados”, la cual se irá informado con los enlaces que se van encontrando cuando cumplen las condiciones:
  - En atributo “href” contenga “equities” y en el atributo “title” contenga “CFD”, consiguiendo tener las URL de cada una de las empresas que componen el IBEX35.
- Una vez tenemos la lista **enlaces\_filtrados**, procedemos a obtener los links necesarios a través de una iteración sobre ella, para ello haremos una iteración (bloque for) donde
  - abrimos URL a partir de la función `get()`
  - Conformamos la primera lista de retorno **URL\_hist** – Para ello Obtenemos todos los elementos que contengan el valor “//a[contains(@href, "historical")” utilizando el acceso por XPath a través de la función `find_elements_by_xpath()`, localizador `By.XPATH`. y se le asignará a “elementos a”, Se realiza una iteración sobre este elemento y obtendremos los links que contienen la información de los valores históricos de cada una de las empresas utilizando la función `get_attribute("href")` , esta nueva información se almacenará en una variable auxiliar de tipo lista “lista\_de\_urls2” que una vez ordenada (sort) y eliminada duplicados (set) será el retorno **URL\_hist**
  - Conformamos la segunda lista de retorno **URL\_categoricas** – Para ello Obtenemos todos los elementos que contengan el valor “//a[contains(@href, "profile")” utilizando el acceso por XPath a través de la función `find_elements_by_xpath()`, localizador `By.XPATH`. y se le asignará a “elementos b”, Se realiza una iteración sobre este elemento y obtendremos los links que contienen la información del profile de las empresas utilizando la función `get_attribute("href")` , esta nueva información se almacenará en una variable auxiliar de tipo lista “lista\_de\_urls\_categoricas\_2” que una vez ordenada (sort) y eliminada duplicados (set) será el retorno **URL\_categoricas**
- Por último, procedemos a salir del navegador con la función `.quit()` para liberar los recursos que pueda estar utilizando.
- Retornamos al proceso principal las listas `URL_hist` y `URL_categoricas`

## ⇒ Definición: Función “HistIbex35”

```
74     ### Definicion de la funcion mediante la cual se va a realizar el WEB SCRAPING de Historicos IBEX35
75     def HistIbex35(url_complet, headers):
76
77         ddH = []                                # Se crea una lista vacía donde se recopilará la información que se extrae de la web
78
79         print("url_complet: ", url_complet)
80         page = requests.get(url_complet, headers )
81         soup = BeautifulSoup(page.content, features="html.parser")
82
83         #En los datos variables que almacenamos el código HTML buscamos la etiqueta H1, para localizar el nombre empresa IBEX35
84         Empresa = soup.find('h1').getText()
85
86         # En los datos variables que almacenamos el código HTML buscamos la palabra clave «table» con la función.find_all()
87         datos = soup.find_all('table')
88         #print(datos[1].prettify())
89
90         datos = datos[:-3] # Eliminamos los tres últimos valores de la lista "datos" porque son tres tablas de datos
91             # globales que no nos interesan en el estudio
92
93         # Con el comando .read_html, la maquina interpretara el html y luego recuperaremos el bloque1 en un DataFrame, que
94         # es donde se encuentran los datos que buscamos.
95         datos1 = pd.DataFrame(pd.read_html(str(datos))[1])
96
97         for i in range(len(datos1)): # recorremos los valores de cada una de las filas del DataFrame
98             try:
99                 Fecha = datos1.Fecha[i]
100            except Exception as err:
101                datos1 = pd.DataFrame(pd.read_html(str(datos))[0])
102
103            Fecha      = datos1.Fecha[i]
104            ultim_avg = format((datos1.Ultimo[i]/1000,'0,.3f').replace({_old: '.', _new: ','}))
105            apert_avg = format((datos1.Apertura[i] / 1000, '0,.3f').replace({_old: '.', _new: ','}))
106            maxim_avg = format((datos1.Máximo[i] / 1000, '0,.3f').replace({_old: '.', _new: ','}))
107            minim_avg = format((datos1.Mínimo[i] / 1000, '0,.3f').replace({_old: '.', _new: ','}))
108            vol_avg   = datos1['Vol.'][i]
109            pvar_avg  = datos1['% var.'][i]
110
111            #Almacenamos los datos obtenidos en un array para añadirlos en el DataFrame dd
112            dd_diaX  = [Empresa, Fecha, ultim_avg, apert_avg, maxim_avg, minim_avg, vol_avg, pvar_avg]
113
114            ddH.append(dd_diaX)
115
116        return ddH
117
```

En esta parte del código se ha definido la función “HistIbex35” que nos ayudará a obtener la información mediante web scraping de la página web de “Datos históricos” de investing.com. Esta función tiene el parámetro de entrada “url\_complet” que corresponde a la URL de la página web comentada por cada una de las empresas que componen el IBEX35 (lista URL\_hist), actualmente el periodo de tiempo que se está obteniendo es fijo de 1 mes teniendo en cuenta el día de ejecución del web scraping. A continuación, se detallan los cálculos y pasos que se realizan dentro de la función:

- En primer lugar, se crea una lista vacía llamada “ddH”, la cual se irá llenando con la información extraída mediante el web scraping.
- Después de obtener la URL completa que se pasa por parámetro, se realiza el web scraping. Para ello se ejecuta la sentencia “requests.get(url\_complet)” para obtener la web que se está solicitando. Una vez hecho esto, se ejecuta la función “BeautifulSoup” para obtener el código fuente de dicha web, una vez aquí se ejecuta la sentencia “find(‘h1’)” con el objetivo de obtener el nombre de la empresa IBEX35 que estamos tratando en cada llamada y finalmente se ejecuta la sentencia “find\_all(‘table’)” para obtener las partes de este código fuente que estén dentro de ‘table’, es decir, la

información que queremos extraer (la información en la página web se encontraba dentro de tablas de datos y por eso buscamos por la palabra ‘table’).

Una vez se ha obtenido la información resultante de la sentencia “find\_all(‘table’)” se borran los tres últimos elementos de la lista ya que son tres tablas con estadísticas globales que para esta práctica no aportan valor alguno.

- A continuación, con el comando .read\_html haremos que la maquina interprete el html y así recuperaremos la información en un DataFrame (datos1) con un formato mas amigable.
- Una vez construido el DataFrame “datos1” (en la cual se encuentra la información de los valores que necesitamos) se realiza un bucle “for” para recorrer cada uno de los elementos de “datos1” (es decir, cada uno de los días del mes en cuestión) y en cada una de estas iteraciones extraemos la información que deseamos mediante las asignaciones correspondientes tal y como se muestran en la captura de pantalla.  
Comentar que para los campos numéricos se ha utilizado el método format, para obtener los decimales.
- Finalmente, creamos una lista llamada “dd\_diaX” con la información recopilada para cada una de las variables que conforman el dataset y de cada día referente a la iteración que estamos realizando, juntándose en la lista “ddH” que hemos al inicio de la función a través de la función append().

⇒ **Definición: Función “categIbex35”**

```
118     """ Definicion de la funcion mediante la cual se va a realizar el WEB SCRAPING para la obtencion de las variables categoricas
119     """ Por cada empresa que componen el IBEX35.
120     def categIbex35(url_complet, headers):
121         print("url_complet (Categoricas): ", url_complet)
122         page = requests.get(url_complet, headers)
123         soup = BeautifulSoup(page.text, features="lxml")
124         dato = soup.find(name='div', attrs={'class': 'companyProfileHeader'})
125
126         # Se crean listas vacias donde se recopilará la información que se extrae de la web para las var. categoricas
127         ddCat = []
128         dda = []      # Lista auxiliar por añadiendo cada uno de los elementos
129
130         # En los datos variables que almacenamos el código HTML buscamos la etiqueta H1, para localizar el nombre empresa IBEX35
131         Empresa = soup.find('h1').getText()
132         dda.append(Empresa)
133
134         # Obtenemos los datos de las etiquetas "a" ("Industria" y "Sector")
135         a_tags = dato.find_all('a')
136         for tag in a_tags:
137             dda.append(tag.get_text())
138
139         # Obtenemos los datos de las etiquetas "p" ("Empleados" y "Tipo de Accion")
140         p_tags = dato.find_all('p')
141         for tag in p_tags:
142             dda.append(tag.get_text())
143
144         #Añadimos las listas creadas con los campos independientes a una q contenga toda la linea.
145         ddCat.append(dda)
146
147     return ddCat
```

En esta parte del código se ha definido la función “categlbex35” que nos ayudará a obtener la información mediante web scraping de la página web de “Profiles” de investing.com. Esta función tiene el parámetro de entrada “url\_complet” que corresponde a la URL de la página web comentada por cada una de las empresas que componen el IBEX35 (lista URL\_categoricas). A continuación, se detallan los cálculos y pasos que se realizan dentro de la función:

- Por cada URL completa que se pasa por parámetro, se realiza el web scraping. Para ello se ejecuta la sentencia “requests.get(url\_complet)” para obtener la web que se está solicitando. Una vez hecho esto, se ejecuta la función “BeautifulSoup” para obtener el código fuente de dicha web, constuimos el obteto dato buscando en soup los datos de la clase companyProfileHeader, utilizando la sentencia soup.find donde buscaremos la etiqueta div con el atributo clase ‘companyProfileHeader’
- Una vez tenemos la información requerida en la variable “dato”, crearemos dos listas vacías, ddCat = contendrá los valores de cada variable obtenida en dda
- Procedemos a obtener los diferentes campos, soup.“find(‘h1’)” obtiene la etiqueta h1 donde se encuentra el nombre de la empresa IBEX35
- Para obtener las variables Empleados y tipos de accion, buscaremos las etiquetas de tipo p dentro de dato (dato.find\_all (“p”)) y haremos un recorrido por cada elemento para obtener el valor con la función get\_text(),
- Para obtener las variables industria y sector, buscaremos las etiquetas de tipo a dentro de dato (dato.find\_all (“a”)) y haremos un recorrido por cada elemento para obtener el valor con la función get\_text(),
- Una vez obtenidas los datos, se añadirán en forma de lista ddCat con la función append() de cada elemento retornado y conformará la lista de retorno de la función.

⇒ **Definición: Función “witreFicheros”**

Esta función nos servirá para imprimir los distintos datasets en formato .csv  
Se le pasa como parámetros de entrada el nombre del archivo y la información que se quiere imprimir (fdatasets), en la función

- Se crea la ruta donde se creará el archivo csv con el dataset obtenido mediante la concatenación de la ruta del directorio actual y el nombre que le queramos dar al archivo csv.
- Finalmente, se crea el archivo csv en la ruta especificada mediante la función “csv.writer”.

## ■ Proceso Principal

```
160
161 ##### -----
162 #   Proceso PRINCIPAL
163 #
164 #   A partir de este punto se ejecutará la función y se creará el fichero csv que contendrá el
165 #   DATASET extraído mediante WEB SCRAPING
166 ##### -----
167
168 ##----- Definimos user-Agent -----
169 # defining the User-Agent header to use in the GET request below
170 headers = {
171     'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/107.0.0.0 Safari/537.36'
172 }
173
174 ##----- Definiciones de variables -----
175 # Definimos las cabeceras para cada uno de los ficheros necesarios para la práctica (datos Históricos, datos Categóricos)
176 camposH = ["Empresa", "Fecha", "Último", "Apertura", "Máximo", "Mínimo", "Vol.", "%var."] #lista con los nombres de los campos a recopilar en histórico
177 camposC = ["Empresa", "Industria", "Sector", "Empleados", "Tipo de Acción"] #lista con los nombres de los campos a recopilar en categóricos
178
179 # Generamos los datasetse que contendrá todos los valores a imprimir para cada uno de los ficheros
180 fdatasets = [] # dataset, auxiliar para enviar a imprimir
181
182 dataset1 = [] # dataset1: Para el fichero "datos_ibex35_dataset.csv"
183 dataset1.append(camposH) # El primer elemento de dd son los nombres de los campos a recopilar
184
185 dataset2 = [] # dataset2: Para el fichero "datos_categoricos_dataset.csv"
186 dataset2.append(camposC) # El primer elemento de dd son los nombres de los campos a recopilar
187
188 ##----- Inicio de programa -----
189 url_inicial = 'https://es.investing.com/indices/spain-35-components' # URL donde se encuentran los componentes IBEX35
190 print('URL inicial: ', url_inicial)
191
192 link_ibex35, link_categoricas = SeleniumURL(url_inicial) # Obtenemos las direcciones de los históricos y categóricas de componentes IBEX35
193
194 i=0 #Bucle para obtener los datos para el fichero "datos_ibex35_dataset.csv".
195 while i < len(link_ibex35): # Accedemos a la página de cada link obtenido para traernos los datos históricos
196     url_complet = link_ibex35[i]
197     dataset = HistIbex35(url_complet, headers)
198     dataset1 = dataset1 + dataset
199     i += 1
200
201
202 i=0 #Bucle para obtener los datos para el fichero "datos_categoricos_dataset.csv".
203 while i < len(link_categoricas): # Accedemos a la página de cada link obtenido para traernos los datos históricos
204     url_complet = link_categoricas[i]
205     dataset = categIbex35(url_complet, headers)
206     dataset2 = dataset2 + dataset
207     i += 1
208
209
210 # Escritura de los datasets en nuestros diferentes archivos .csv
211 nombre_archivo = "datos_ibex35_dataset.csv"
212 fdatasets = dataset1
213 writeFicheros(nombre_archivo, fdatasets)
214
215
216 nombre_archivo = "datos_categoricos_dataset.csv"
217 fdatasets = dataset2
218 writeFicheros(nombre_archivo, fdatasets)
219
```

En el proceso principal, Después de definir las diferentes funciones se realizan sus ejecuciones  
Con la siguiente secuencia

- Creamos un user-agent, que utilizaremos para mejorar la recuperación y facilitar la interacción con el contenido web
- Definimos las cabeceras de nuestros dos ficheros .csv con los nombres de los campos (camposH y camposC )
- Generaremos los datasets (dataset1 y dataset2) que luego enviaremos a imprimir en la función writeFicheros

- Se define la url\_inicial donde vamos a encontrar los componentes de IBEX35 dentro de investing.com.
- Ejecutamos la función SeleniumURL (definida al inicio) utilizando como parámetro la variable “url\_inicial”, esta función nos retornará 2 listas con todas las URL, donde podemos encontrar en una lista la historia de los valores por fecha, de cada empresa que componen IBEX35 y se la asignaremos a la variable **link\_ibex35** y por otro lado la lista donde contendrá las URL de cada empresa para acceder al profile de esta y se la asignaremos a la variable **link\_categoricas**
- Por cada lista que tenemos se realizará un bucle “while” de forma que se irá iterando mientras existan elementos en cada una de las listas
- Para la lista link\_ibex35, dentro de cada bucle, se realiza la llamada para ejecutar la función HistIbex35 utilizando como parámetro la variable “url\_complet”, y el user-Agent y asignaremos el DataFrame que retorna a dataset, este se irá añadiendo a la variable **dataset1** con el objetivo de que esta última contenga toda la información a trasladar a un fichero .csv
- Lo mismo ocurre para la lista link\_categoricas, dentro de su bucle, se realiza la llamada para ejecutar la función categIbex35 utilizando como parámetro la variable “url\_complet”, y el user-Agent y asignaremos el DataFrame que retorna a dataset, este se irá añadiendo a la variable **dataset2** con el objetivo de que esta última contenga toda la información a trasladar a un fichero .csv
- Finalmente, se llamará a la función writeFicheros para cada uno de los datasets generados e indicándole el nombre de cada uno de ellos creando los archivo csv en la ruta especificada mediante la función “csv.writer”.

## PUBLICACION del DATASET

Zenodo:

Para finalizar la práctica se ha publicado el dataset en formato CSV en Zenodo. El DOI obtenido en la publicación es [10.5281/zenodo.10118489](https://doi.org/10.5281/zenodo.10118489). A continuación, adjuntamos una captura de pantalla a modo de comprobación de que la publicación se ha realizado correctamente.

The screenshot shows the Zenodo interface. At the top, there's a blue header bar with the Zenodo logo on the left, a search bar containing 'Search records...', and navigation links for 'Communities' and 'My dashboard' on the right. Below the header, the page title 'Ibex35: Valores de sus componentes' is displayed, along with the authors 'Suarez, Alberto; Pallares, Leonor'. A note below the title states: 'El dataset se basa en el conjunto de datos basados en una serie de cotizaciones diarias del conjunto de empresas que conforman el índice de referencia IBEX35'. The main content area shows a table of files, with two rows visible: 'datos\_categoricos\_dataset.csv' and 'datos\_ibex35\_dataset.csv'. Each row includes a preview and download link.

Published November 14, 2023 | Version v1

[Dataset](#) [Open](#)

## Ibex35: Valores de sus componentes

Suarez, Alberto; Pallares, Leonor

El dataset se basa en el conjunto de datos basados en una serie de cotizaciones diarias del conjunto de empresas que conforman el índice de referencia IBEX35

### Files

<a href="#">datos_categoricos_dataset.csv</a>	<a href="#">»</a>
<a href="#">Files (62.9 kB)</a>	
<b>Name</b>	<b>Size</b>
<a href="#">datos_categoricos_dataset.csv</a> md5:87b55e65ebfa0a88cb5849ec1fc006cb ⓘ	2.8 kB
<a href="#">datos_ibex35_dataset.csv</a> md5:c51b808b68d7d12d49213d72f58be546 ⓘ	60.2 kB

Adjuntamos link para su acceso: <https://zenodo.org/records/10118490>.

### GIT HUB:

En el git, se han publicados los los archivos referentes a la práctica realizada, adjuntamos el link de acceso y una imagen con la información existente

[https://github.com/albvensuarod/IBEX\\_35\\_WEB\\_SCRAPING/blob/main/README.md](https://github.com/albvensuarod/IBEX_35_WEB_SCRAPING/blob/main/README.md)

The screenshot shows a GitHub repository page. At the top, it displays the repository name 'albvensuarod / IBEX\_35\_WEB\_SCRAPING'. A search bar is present with the placeholder 'Type / to search'. Below the search bar are navigation links: Code, Issues, Pull requests, Actions, Projects, Security, and Insights. The repository title 'IBEX\_35\_WEB\_SCRAPING' is shown in bold, with a 'Private' badge next to it. To the right of the title is a 'Watch' button. Below the title, there are buttons for 'main' (with a dropdown arrow), '1 branch', '0 tags', 'Go to file', 'Add file', and a green 'Code' button. The main content area lists files and commits. A commit by 'albvensuarod' titled 'Update README.md' is at the top, dated '12 hours ago' with '16 commits'. Below it is a list of files added via upload: 'IBEX35\_memoria\_v2.docx', 'INTENTO\_FECHA.py', 'PRA1\_AS\_LP\_IBEX35.py', 'README.md', 'varaiables\_numericas\_IBEX35.csv', and 'variables\_categoricas\_IBEX35.csv'. Each file entry includes its type (file icon), name, action (Add files via upload), and date (e.g., '4 days ago', '17 hours ago', etc.).

## VIDEO

Se ha realizado video de una duración de max 10 minutos, donde se explica el objetivo de este proyecto. El video se ha ubicado en el repositorio ondrive de la UOC y es accesible a través de este enlace.

[https://drive.google.com/drive/folders/18AyZnYfF\\_NBjcv5w3avZS-PSjvGHQ6OD?usp=sharing](https://drive.google.com/drive/folders/18AyZnYfF_NBjcv5w3avZS-PSjvGHQ6OD?usp=sharing)

Contribuciones	Firma
Investigación previa	AS, LP
Redacción de las respuestas	AS, LP
Desarrollo del código	AS, LP