ÍNDICE DE CONTENIDOS

CONTEXTO

Los datos recolectados en éste proyecto han sido extraídos del sitio web www.expedia.es con fines académicos. El objetivo académico de la extracción de datos es avanzar en el conocimiento de las técnicas de *web scraping*.

Dado que el enunciado de la práctica daba libertad para escoger el sitio web deseado, hemos decidido extraer información de www.expedia.es dado que es una agencia de viajes que ofrece servicio completo de agencia de viajes como pueden ser: la reserva de billetes de avión, hotel, alquiler de vehículos, cruceros, paquetes vacacionales y varios parques de atracciones, y todo ello a través de internet.

Esta elección estaba supeditada al análisis de las restricciones y características del sitio web para evaluar la legalidad y complejidad de la misma. En este análisis, el primer paso fue acceder al fichero robots.txt del dominio www.expedia.es y este fue el resultado:

```
# This file MUST NOT be edited without approval from the SEO team.
User-agent: *
Disallow: /pub/agent.dll?qscr=mrdt
Disallow: /pub/agent.dll?qscr=mrdt
Disallow: /pub/agent.dll?qscr=mrdr
Disallow: /cd/
Disallow: /pub/agent.dll?qscr=pkgl*
Disallow: /Mates-Rates
Disallow: /Details
Disallow: /pubspec/scripts/
Disallow: /ugc/urs/api/hotelreviews/hotel/HOTELID/
Disallow: /23171577/
Disallow: /recommendations/
User-agent: google-hoteladsverifier
Allow: /
User-agent: AdIdxBot
Allow: /
```

Comprobamos que la ruta www.expedia.es/flights-search no estaba restringida.

Curiosamente, el fichero robots.txt de la versión internacional de Expedia es muy diferente. En el mismo, se establece una regla de Disallow para la ruta /Flights-Search:

```
# This file MUST NOT be edited without approval from the SEO team. Plea

User-agent: *
Disallow: /daily/common/
Disallow: /cd/
Disallow: /pub/agent.dll?qscr=mrdt
Disallow: /pub/agent.dll?qscr=mrdr
Disallow: /daily/vacations/merch/
Disallow: /daily/hotels/all-inclusive.asp
Disallow: /daily/hotels/all_inclusive.asp
Disallow: /pubspec/scripts/
Disallow: /ReviewSubmission
Disallow: /review-inhouse-submission
Disallow: /Hotels/Offers?action=
Disallow: /ads/container
Disallow: /event.ng/
Disallow: /html.ng/
Disallow: /js.ng/
Disallow: /click.ng/
Disallow: /image.ng/
Disallow: /ping.ng/
Disallow: /event.cms/
Disallow: /html.cms/
Disallow: /js.cms/
Disallow: /click.cms/
Disallow: /image.cms/
Disallow: /ping.cms/
Disallow: /pub/agent.dll?qscr=pkgl*
Disallow: /Mates-Rates
Disallow: /Details
Disallow: /carsearch
Disallow: /Flights-Search
Disallow: /Flight-SearchResults
Disallow: /Flights-BagFees
Disallow: /user/
Disallow: /packagesearch
Disallow: /ugc/urs/api/hotelreviews/hotel/HOTELID/
Disallow: /23171577/
```

Por lo tanto, decidimos realizar el scraping sobre la versión española (www.expedia.es).

En éste proyecto nos vamos a centrar en trabajar en los **vuelos de ida únicamente**, dado que realizar la extracción de las demás opciones no aportaría contenido académico al trabajo, y sí redundancia de repetir una misma operación. No queremos decir que no fuese interesante si el fin fuese comercial, pero dado que no es el caso, creemos que no resulta de interés académico.

El objetivo de ésta extracción iría enfocado a poder aplicar un modelo en el que se pueda predecir el mejor momento del año para poder comprar un cierto billete de avión. Para ello es necesario automatizar la extracción de ésta web que combina ofertas de diferentes compañías aéreas, y con dicha información almacenada entrenar un modelo que aprenda de cómo se han comportado los precios en el pasado para poder predecir los del futuro.

TÍTULO DEL DATASET

Título del *data set*: **Precios de un vuelo concreto por mes.**
Siglas: **PVCM**
Título de los archivos: **PVCM_MM1-AAAA1_DD2-MM2-AAAA2.csv**

DESCRIPCIÓN DEL DATASET

Dado que el objetivo que nos hemos fijado es el de saber cuál es el mejor **mes** para ir a un determinado sitio de **nuestra lista de países por visitar** saliendo desde nuestra lista de **aeropuertos cercanos**, se ha decidido extraer información el día 15 de cada mes del año que combine todas las salidas y llegadas de forma que dispongamos de información referente a 36 meses en el momento de la predicción. El motivo de guardar 3 años sería el de poder encontrar patrones, dado que, aunque la media anual cambie, los patrones pueden seguir siendo los mismos, o por lo menos tenemos la oportunidad de comprobarlo.

Por lo tanto, el usuario del hipotético sistema de predicción tendría que indicar el mes en el que quiere viajar, y el país al que quiere viajar, y el sistema le indicaría los mejores meses en orden descendente para comprar el billete a ese sitio en concreto y ese mes en concreto.

En nuestro *data set*, por acotar la información almacenada, vamos a fijar la siguiente lista de países por visitar, y lista de aeropuertos cercanos:

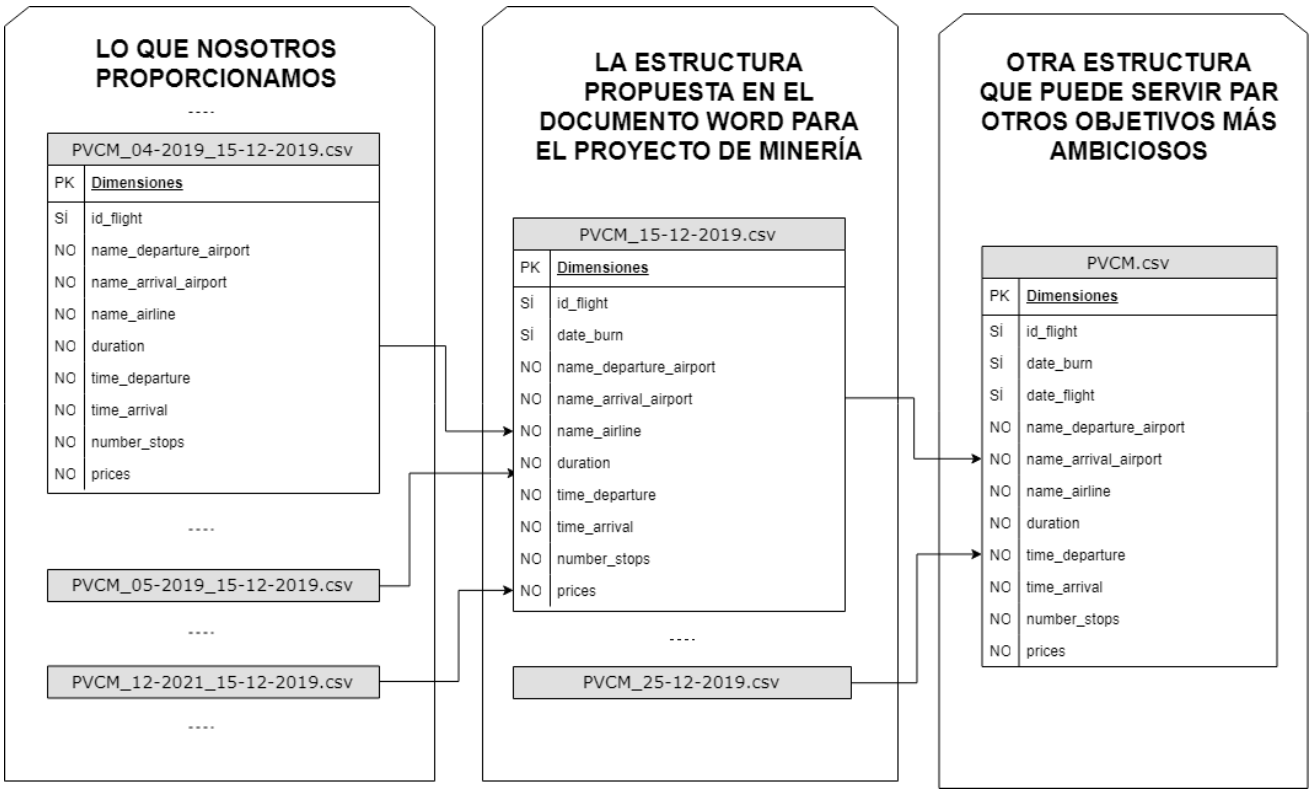
- Lista de países por visitar:
  - Aeropuerto de Nueva York (NYC)
  - Aeropuerto de Londres (LHR-Heathrow)
- Lista de aeropuertos cercanos:
  - Aeropuerto de Bilbao (BIO)
  - Aeropuerto de San Sebastián (EAS)

Los datos extraídos cada mes se almacenarán en un archivo .csv diferente, con un nombre diferente como ya se ha comentado en el apartado *Descripción del Dataset*. Un ejemplo de archivo en un mes concreto sería **PVCM_04-2019_15-12-2019.csv**. Las columnas están separadas por comas “,” y en el propio archivo se ha incluido el nombre de cada columna. A continuación, se muestran las primeras dos líneas de un archivo .csv para que se entienda lo explicado:

date_flight,name_departure_airport,name_arrival_airport,name_airline,duration,time_departure,time_arrival,number_stops,prices

03/10/2019,BILBAO,NEW YORK,TAP Portugal,29 h 55 min,20:05,20:00,(1 escala),250 €

REPRESENTACIÓN GRÁFICA



CONTENIDO

En la siguiente tabla se exponen los diferentes atributos que vamos a extraer o añadir a nuestro *data set*:

NOMBRE COLUMNA	TIPO DE DATOS	ACEPTA NULOS	FORMATO	DÓNDE SE ALMACENA	PRIMARY KEY
Id_flight	INTEGER(5)	NO	-	Columna	SÍ
date_burn	DATE	NO	mm-aaaa	Nombre.csv	SÍ
date_flight	DATE	NO	dd-mm-aaaa	Nombre.csv	NO
name_departure_airport	CHARACTER VARYING(20)	NO	-	Columna	NO
name_arrival_airport	CHARACTER VARYING(20)	NO	-	Columna	NO
name_airline	CHARACTER VARYING(30)	NO	-	Columna	NO
duration	CHARACTER VARYING (15)	NO	Xd Xh Xm	Columna	NO
time_departure	TIME	NO	hh:mm (23:59h)	Columna	NO
time_arrival	TIME	NO	hh:mm (23:59h)	Columna	NO
number_stops	CHAR(10)	NO	(X stop/s) / (Nonstop)	Columna	NO
prices	CHARACTER VARYING (7)	NO	X.XXX €	Columna	NO

Nótese que no se ha añadido la columna *date_extraction* dado que cada **PVCM_MM1-AAAA1_DD2-MM2-AAAA2.csv** hace referencia a un mes diferente de extracción. El motivo de realizar esto así es que en el caso de que no acotemos y queramos guardar todos los datos de todas las combinaciones de países que pueda haber en el sitio web (lo cual sería una aplicación más cerca de la realidad, y menos académica) cada mes ocuparía una extensión de información muy grande, por lo que lo más óptimo para analizar el intervalo de tiempo deseado sería guardarlo en archivos por meses y luego cargar únicamente los archivos deseados para entrenar al modelo.

Los datos van a ser válidos durante 3 años. El motivo es que, aunque los más trascendentes sean los del año anterior, los de los dos años anteriores pueden ayudar al modelo a entender comportamientos a lo largo de los años, es decir, aunque la media anual de precios pueda subir o bajar, es muy posible que se encuentren patrones que nos ayuden a entender si son particulares del año pasado o se repite a lo largo de los años. Almacenar más de 3 años puede resultar inútil dado que incluso las tendencias pueden cambiar por motivos socioeconómicos a nivel global.

La tecnología utilizada para realizar la extracción de los datos ha sido la librería **selenium** dentro de Python, partiendo de una base creada con Anaconda. Hemos escogido un driver de chromium para poder proceder a la extracción. En el apartado de *agradecimientos* se muestran los enlaces visitados para poder operar con las tecnologías mencionadas.

AGRADECIMIENTOS

El propietario de los datos antes de la extracción era expedia.es, una empresa nacida en el seno de Microsoft y que en 2001 fue vendida a USA Networks. Agradecemos a ésta empresa su permisividad a la hora de compartir datos en su portal español expedia.es como hemos mostrado en el apartado de *Contexto* cuando hemos tratado el tema de www.expedia.es/robots.txt.

En cuanto a la base de datos generada de la extracción, ofrecemos acceso libre a todo usuario de la web que quiera acceder a él.

Cuando se nos presentó el enunciado de la práctica, dimos inicio a un proceso de investigación para descubrir las librerías, herramientas y tecnologías existentes para realizar Web Scraping. A continuación listamos las librerías que hemos analizado en este proceso:

- Scrapy
- Requests
- BeautifulSoup
- Selenium

El siguiente enlace web nos ayudó mucho a la hora de entender las principales características y diferencias entre ellas: https://info.scrapinghub.com/web-scraping-guide/python-web-scraping-libraries-and-frameworks

Se adjunta la captura de la tabla comparativa que se presenta en el enlace anterior:

	Scrapy	Requests	BeautifulSoup	Selenium
What is it?	Web scraping framework	Library	Library	Library
Purpose	Complete web scraping solution	Simplifies making HTTP requests	Data parser	Scriptable web browser to render javascript
Ideal Use Case	Development of recurring or large scale web scraping projects	Simple non-recurring web scraping tasks	Simple non-recurring web scraping tasks	Small-scale web scraping of javascript heavy websites
Built-in Data Storage Supports	JSON, JSON lines, XML, CSV	Need to develop your own	Need to develop your own	Customizable
Available Selectors	CSS & Xpath	N/A	CSS	CSS & Xpath
Asynchronous	Yes	No	No	No
Javascript support	Yes, via Splash library	N/A	No	Yes
Documentation	Excellent	Excellent	Excellent	Good
Learning Curve	Easy	Very Easy	Very Easy	Easy
Ecosystem	Large ecosystem of developers contributing projects and support on Github and StackOverflow	Few related projects or plugins	Few related projects or plugins	Few related projects or plugins
Github Stars	29,239	34,727	-	11,926

Para poder llevar a cabo éste proyecto se ha accedido a al siguiente link para descargar el programa chromedriver.exe que nos permite acceder al sitio de expedia.es haciendo uso de chromium:

http://chromedriver.chromium.org/downloads

Como ya hemos comentado en el apartado de *Contenido*, como hemos utilizado selenium en Python, nos ha sido de utilidad la siguiente referencia para entender la librería mejor:

https://selenium-python.readthedocs.io/

La forma de trabajar ha sido instalando anaconda que incluye las librerías más utilizadas en la realización de la mayoría de programas mediante el siguiente enlace:

https://www.anaconda.com/distribution/

Para poder trabajar con anaconda hemos instalado mediante el siguiente comando la librería selenium:

conda install -c conda-forge selenium

Encontrado en el sitio:

https://anaconda.org/conda-forge/selenium

INSPIRACIÓN

Como ya hemos adelanta en el contexto, el objetivo sería **predecir el mes del año en el que resulta más interesante comprar un billete a un destino de nuestra lista de ciudades a visitar desde un aeropuerto que nos quede cerca de casa en un vuelo que salga un día concreto del año**.

Para poder obtener dicha respuesta, hemos de aplicar un **modelo de regresión** debido a que podríamos decir que es un problema de clasificación con clases contínuas. Es decir, nuestro objetivo sería predecir la curva (o modelo) de un valor numérico (el precio) para posteriormente calcular los mínimos y decidir si llegamos a tiempo para comprar el billete en un mínimo global o nos tenemos que esperar y conformar con el siguiente mínimo local de nuestro modelo.

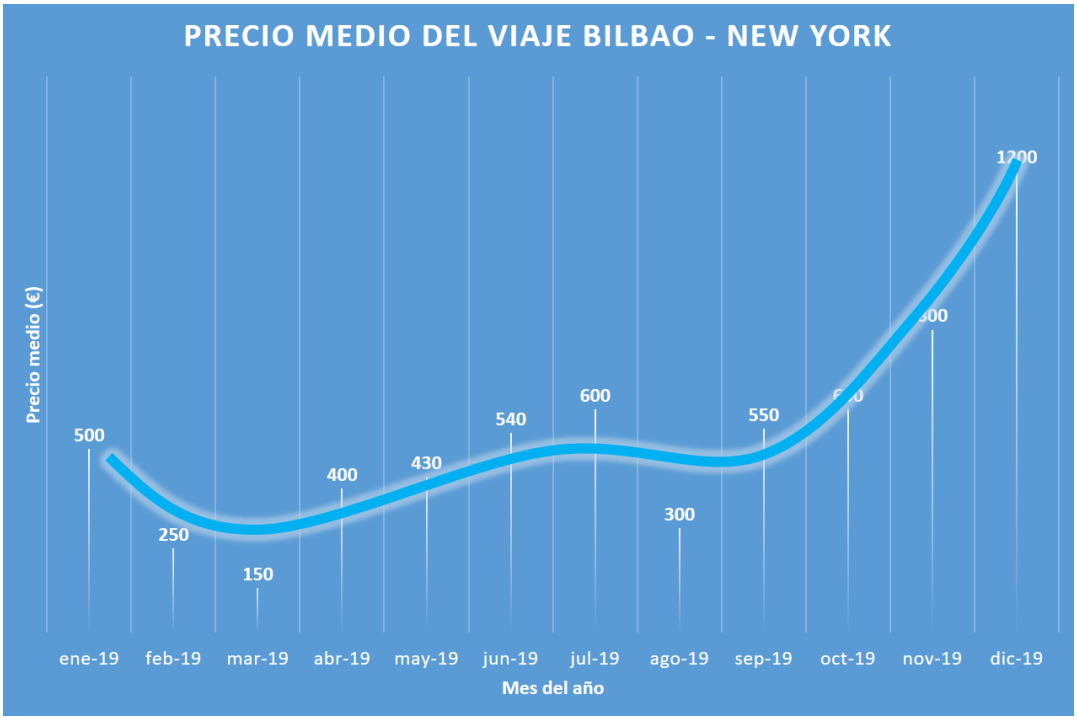
Dicho de otra forma, nuestro objetivo es encontrar la función de regresión:

$$f : X \rightarrow \mathbb{R}$$

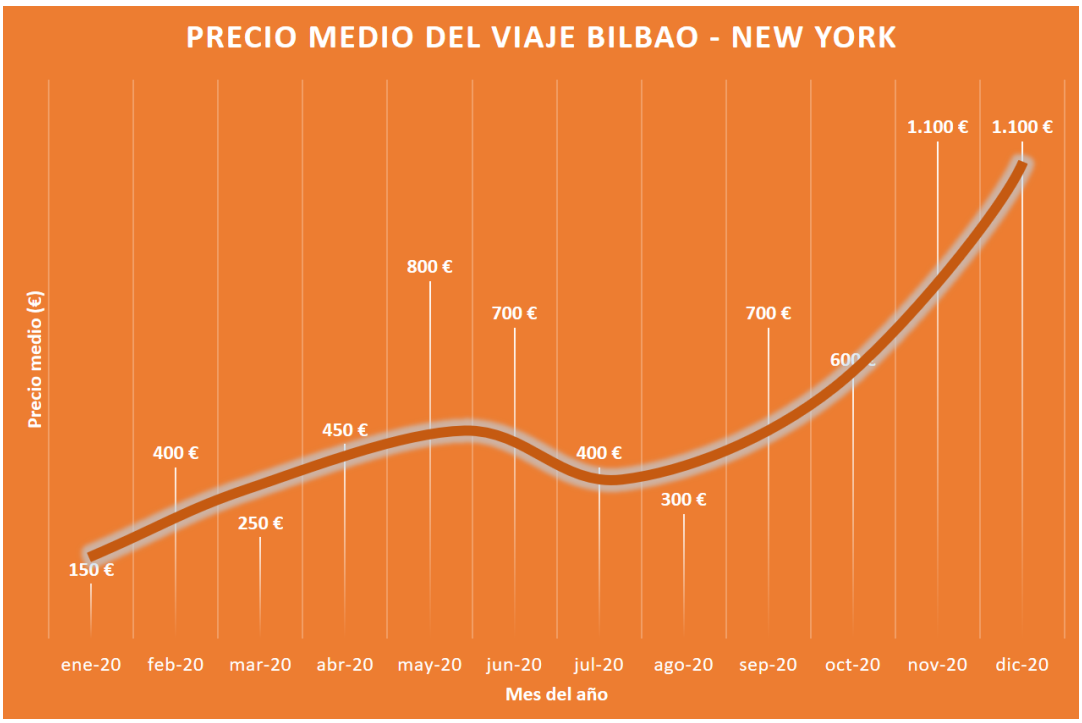
Donde f representa la función de regresión, X el conjunto de atributos que forman una instancia y R un valor en el dominio de los números reales.

A continuación, se va a describir un ejemplo de modelo regresión que podría servirnos para obtener nuestro propósito. Para realizar la descripción se va a usar una representación esquemática de lo que sería el modelo dado que para poder ponerlo en práctica hacen falta por lo menos 1 año, y si son 3 mejor. Además, si los datos en vez de guardarse cada mes se guardan más veces, la calidad del modelo mejoraría, aunque no mucho.

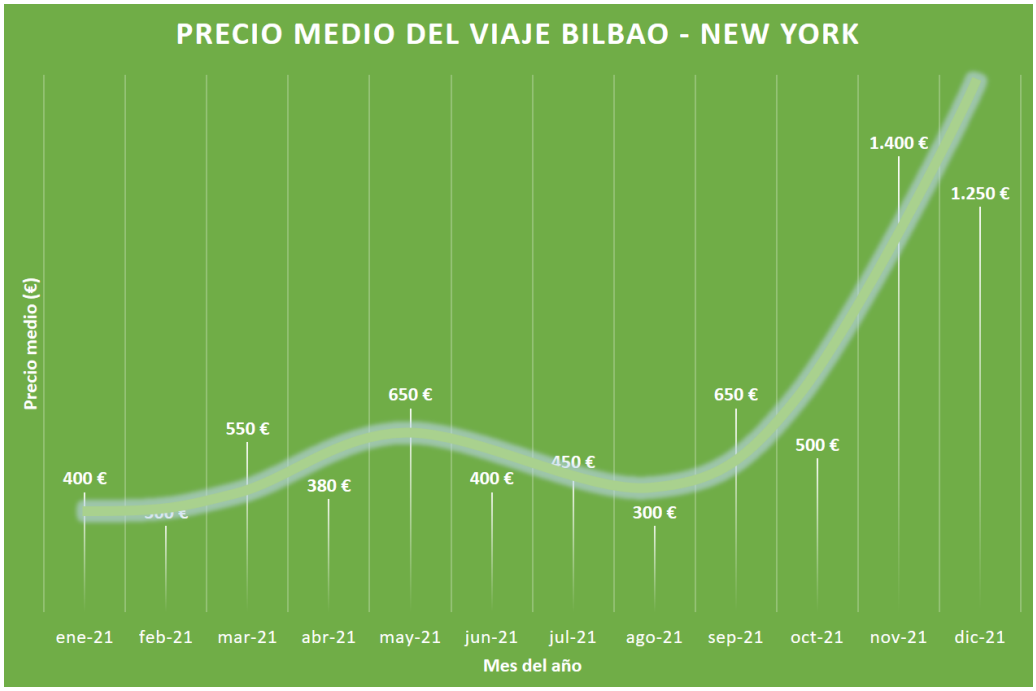
En primer lugar, vamos a finarnos lo que pasaría si empezando desde enero a diciembre del 2019 guardamos todos los meses información acerca de la fecha en la que estamos interesados en viajar:



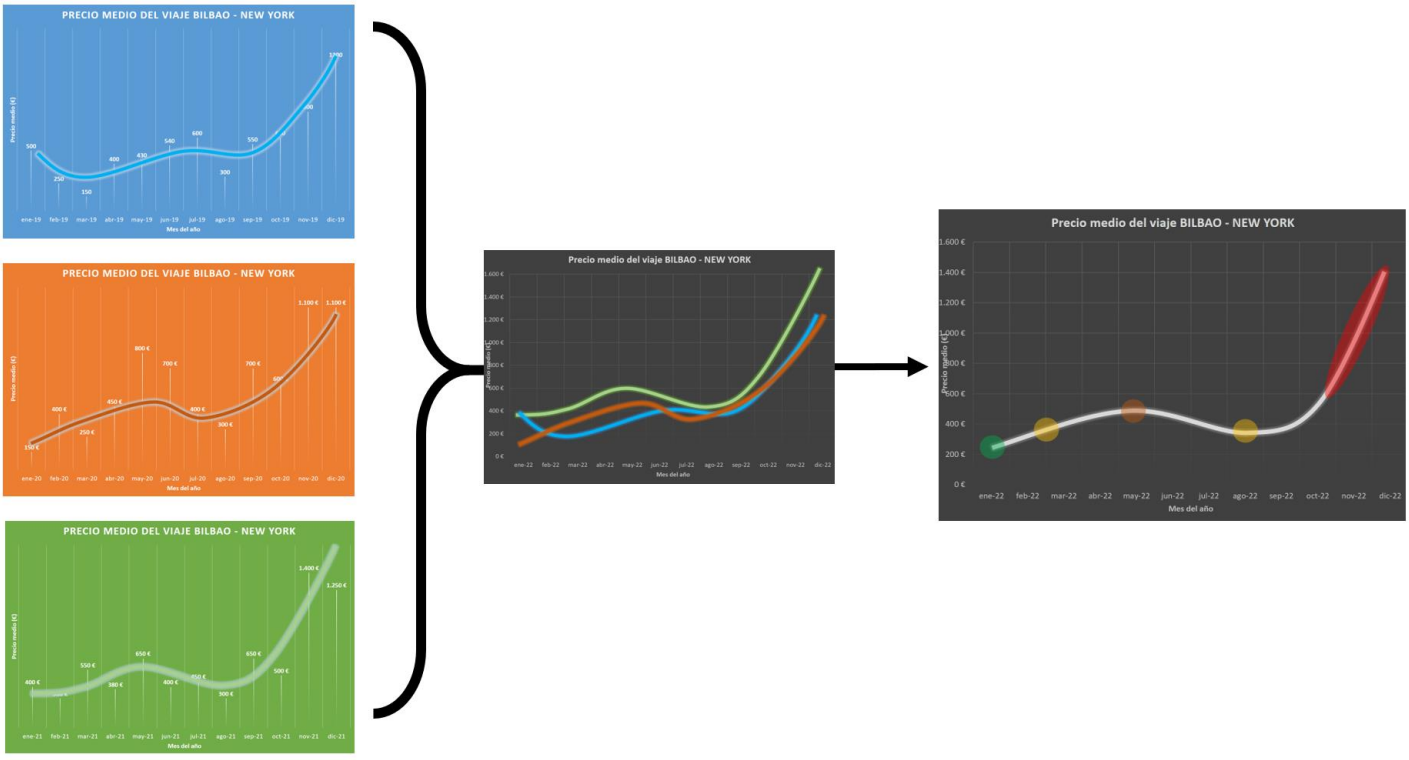
Después hacemos lo mismo para el año 2020:



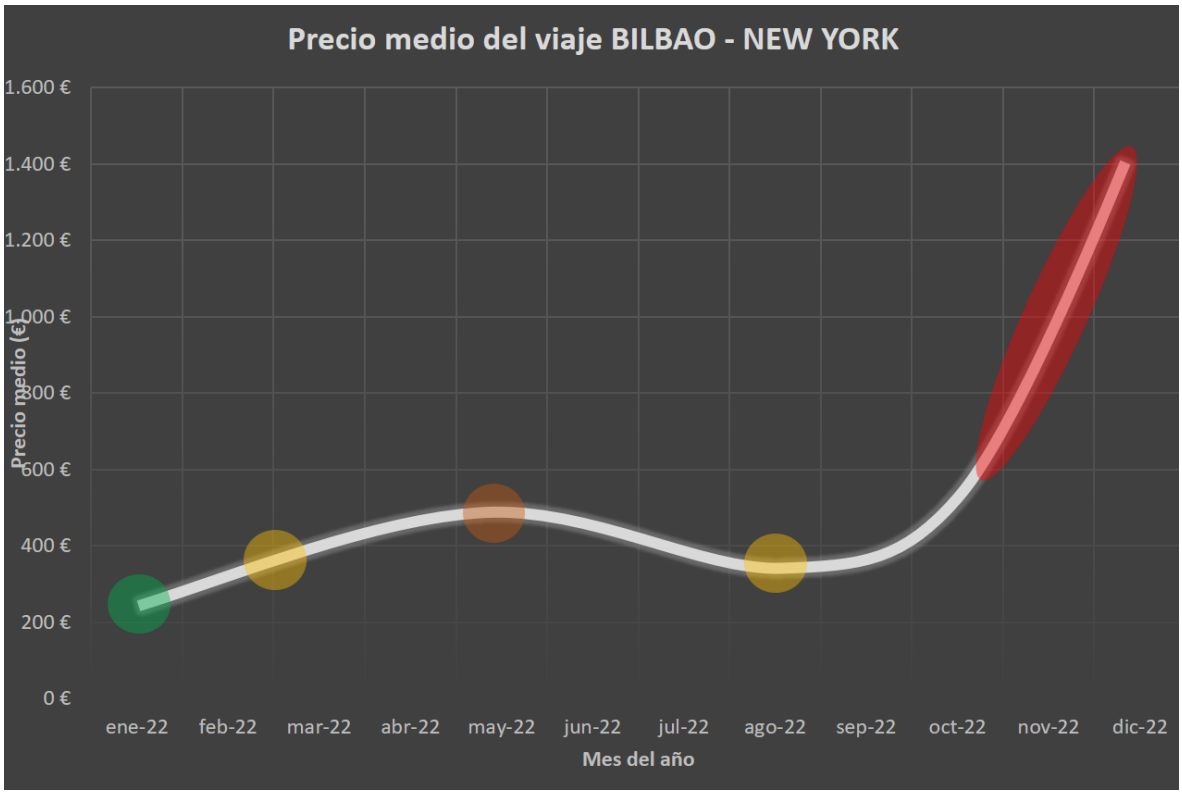
Y para el año 2021 finalmente:



Una vez tenemos los tres modelos de los tres años, podríamos juntarlos en uno solo y conocer una tendencia, que luego podría ser escalada teniendo en cuenta la recopilación de la misma información en los primeros meses del 2022. A continuación, se muestra el esquema del proceso (sin escalado):



Por lo tanto, de éste hipotético modelo que hemos generado, el resultado sería algo así como:



Además, dado que no solo nos interesa como destino NEW YORK, también hemos añadido otros destinos para que el usuario pueda hacer mejores valoraciones. Para esto, había que repetir el mismo proceso explicado generando diferentes modelos para cada destino.

Para finalizar con éste apartado, cabe destacar que éste conjunto de datos, además de para contestar a la pregunta que hemos realizado, nos puede servir para contestar a unas cuantas preguntas más que pueden realizarse incluso con otros tipos de modelos diferentes a los de regresión, dado que el número de dimensiones que se almacenan en dicha base de datos así lo permite.

Por poner un ejemplo, y sin profundizar tanto como en el anterior, podríamos utilizar un **modelo de clasificación** previo a una **discretización de los precios** en bloques de: muy barato, barato, medio, caro, y muy caro (o en menos grupos, esto sería cuestión de probar) de forma que podamos clasificar las aerolíneas que más vuelos ofrecen con el tipo de billete que solemos comprar, de forma que si la diferencia entre los precios de dos aerolíneas es parecido, pero sabemos que una de ellas está dentro de las que suelen adaptarse a mi rango de precios, me compense comprar el billete con esa aerolínea para beneficiarme de los puntos que ésta pueda darme, y por tanto conseguir descuentos más importantes.

LICENCIA

Hemos seleccionado la licencia *Open Database License (ODbL)* para nuestro dataset. Esto quiere decir que los usuarios pueden compartir sus datos con libertad y sin temor a los derechos de autor o cuestiones de propiedad. Cualquier puede hacer uso libre de los datos contenidos en este dataset sin miedo a ninguna infracción de derechos de autor y se permite que utilicen los datos para añadirlos a sus bases de datos. El objetivo de esta licencia es el de comprar y compartir información y datos de forma fácil y libre.

Estas son las condiciones que define la licencia *Open Database License*:

- Atribución: La referencia a la base de datos se debe incluir en todo momento en cualquier emisión pública con información del dataset y conclusiones derivadas.
- Compartir igual: las adaptaciones, revisiones o adiciones a los contenidos o estructura de la organización de la base de datos también deben estar a libre disposición de los otros usuarios bajo la Licencia Abierta de Bases de Datos.

Mantener abierta: la redistribución de cualquier versión o de cualquier parte de la base de datos debe estar restringida. Ninguna sección de la base de datos puede estar detrás de un PayWall.

CÓDIGO

```
#################################################################
################  ExpediaScraper.py  ############################
#################################################################


from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.chrome.options import Options


import pandas as pd
import time
import datetime
import myconstants as mc
import from_cities as fc
import to_cities as tc
import urllib.robotparser


opts = Options()
opts.add_argument("user-agent=Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) \
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36")
browser = webdriver.Chrome(executable_path='chromedriver',chrome_options=opts)

# User space
day = '15'
month = '12'
year = '2019'


debug = True
csv_index = 0;


# Tickets
## Ticket type path
one_way_ticket = "//label[@id='flight-type-one-way-label-hp-flight']"
#return_ticket = "//label[@id='flight-type-roundtrip-label-hp-flight']"
#multi_ticket = "//label[@id='flight-type-multi-dest-label-hp-flight']"


## Choose ticket type
def ticket_chooser(ticket):
    try:
        ticket_type = browser.find_element_by_xpath(ticket)
        ticket_type.click()
    except Exception as e:
        pass
```

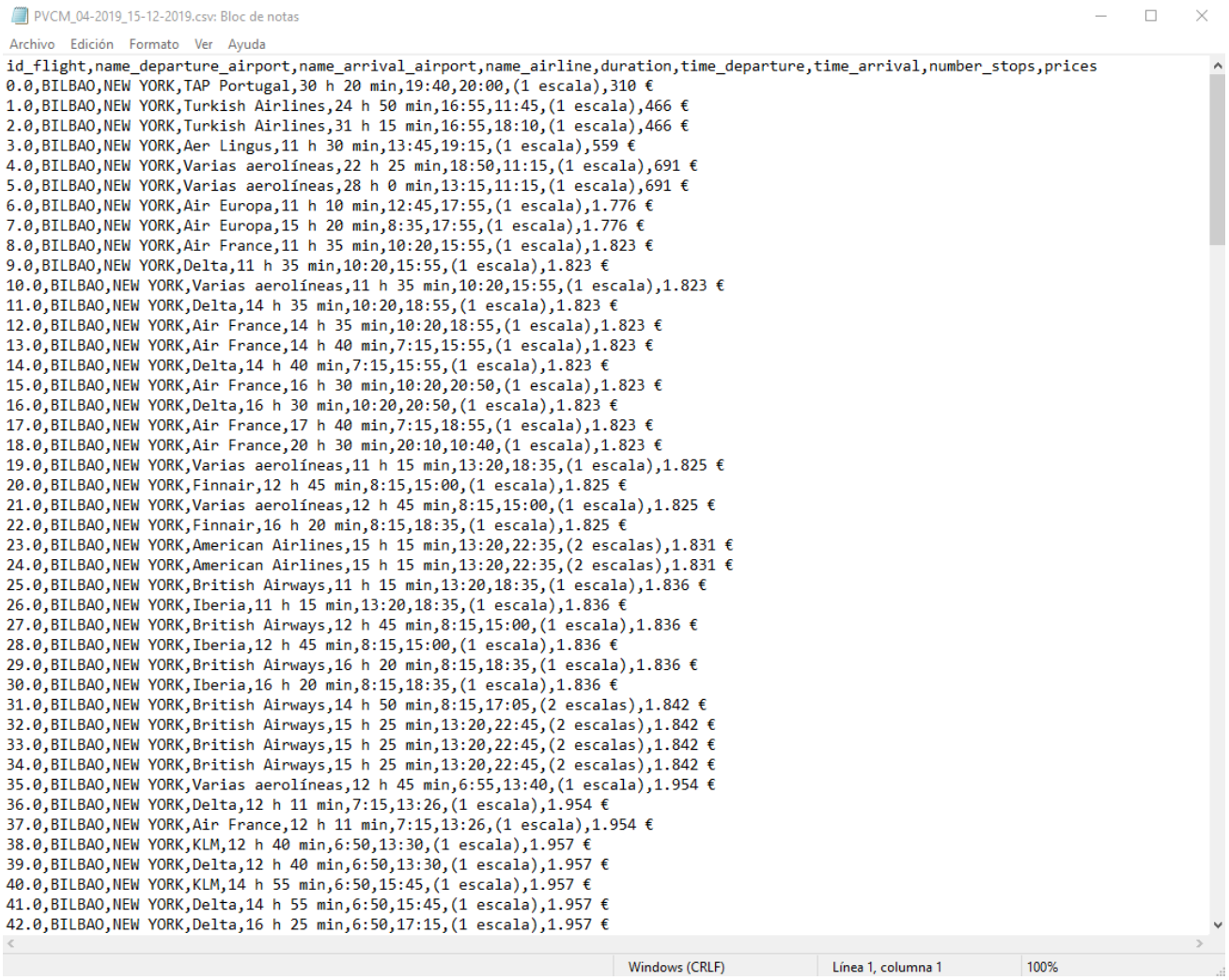
```
# Choose departure country
def departure_city_chooser(departure_city):
    fly_from = browser.find_element_by_xpath("//input[@id='flight-origin-hp-flight']")
    time.sleep(mc.wait_next_step)
    fly_from.clear()
    time.sleep(mc.wait_next_step)
    fly_from.send_keys('  ' + departure_city)
    time.sleep(mc.wait_next_step)
    first_item = browser.find_element_by_xpath("//a[@id='aria-option-0']")
    time.sleep(mc.wait_next_step)
    first_item.click()

# Choose arrival country
def arrival_city_chooser(arrival_city):
    fly_to = browser.find_element_by_xpath("//input[@id='flight-destination-hp-flight']")
    time.sleep(mc.wait_next_step)
    fly_to.clear()
    time.sleep(mc.wait_next_step)
    fly_to.send_keys('  ' + arrival_city)
    time.sleep(mc.wait_next_step)
    first_item = browser.find_element_by_xpath("//a[@id='aria-option-0']")
    time.sleep(mc.wait_next_step)
    first_item.click()

# Choose departure date
def departure_date_chooser(day, month, year):
    dep_date_button = browser.find_element_by_xpath("//input[@id='flight-departing-single-hp-
flight']")
    dep_date_button.clear()
    dep_date_button.send_keys(day + '/' + month + '/' + year)

# Search
def search():
    search = browser.find_element_by_xpath("//button[@class='btn-primary btn-action gcw-submit']")
    search.submit()
    time.sleep(mc.wait_search)

# Create data frame
df = pd.DataFrame()
def compile_data(day, month, year, from_city, to_city):
    global df
    global dep_times_list
    global arr_times_list
    global airlines_list
    global price_list
    global durations_list
    global stops_list
    global price_list

    #add new rows after existing ones
    csv_index = len(df.index)

    #name_airline
    airlines = browser.find_elements_by_xpath("//span[@data-test-id='airline-name']")
    airlines_list = [value.text for value in airlines]
```

```
#duration
durations = browser.find_elements_by_xpath("//span[@data-test-id='duration']")
durations_list = [value.text for value in durations]

#time_departure
dep_times = browser.find_elements_by_xpath("//span[@data-test-id='departure-time']")
dep_times_list = [value.text for value in dep_times]

#time_arrival
arr_times = browser.find_elements_by_xpath("//span[@data-test-id='arrival-time']")
arr_times_list = [value.text for value in arr_times]

#number_stops
stops = browser.find_elements_by_xpath("//span[@class='number-stops']")
stops_list = [value.text for value in stops]

#prices
prices = browser.find_elements_by_xpath("//span[@data-test-id='listing-price-dollars']")
price_list = [value.text for value in prices]

for i in range(len(dep_times_list)):
    try:
        df.loc[int(i) + csv_index, 'id_flight'] = int(i)
    except Exception as e:
        pass

    try:
        df.loc[i + csv_index, 'name_departure_airport'] = from_city
    except Exception as e:
        pass

    try:
        df.loc[i + csv_index, 'name_arrival_airport'] = to_city
    except Exception as e:
        pass

    try:
        df.loc[i + csv_index, 'name_airline'] = airlines_list[i]
    except Exception as e:
        pass

    try:
        df.loc[i + csv_index, 'duration'] = durations_list[i]
    except Exception as e:
        pass

    try:
        df.loc[i + csv_index, 'time_departure'] = dep_times_list[i]
    except Exception as e:
        pass

    try:
        df.loc[i + csv_index, 'time_arrival'] = arr_times_list[i]
    except Exception as e:
        pass

    try:
```

```
            df.loc[i + csv_index, 'number_stops'] = stops_list[i]
        except Exception as e:
            pass

        try:
            df.loc[i + csv_index, 'prices'] = price_list[i]
        except Exception as e:
            pass

# Run code
rp = urllib.robotparser.RobotFileParser()
rp.set_url("https://www.expedia.es/robots.txt")
rp.read()
if (rp.can_fetch("*", "https://www.expedia.es/Flights-Search")):
    for from_city in fc.from_cities:
        for to_city in tc.to_cities:
            link = 'https://www.expedia.es/'
            browser.get(link)
            time.sleep(mc.wait_open_url)

            #choose flights only
            flights_only = browser.find_element_by_xpath("//button[@id='tab-flight-tab-hp']")
            flights_only.click()
            ticket_chooser(one_way_ticket)
            departure_city_chooser(from_city)
            arrival_city_chooser(to_city)
            departure_date_chooser(day, month, year)
            search()
            compile_data(day, month, year, from_city, to_city)
            if (debug):
                print(df)
            time.sleep(mc.wait_next_step)
    df.to_csv('PVCM_' + time.strftime("%m-%Y") + '_'+ day + '-'+ month + '-' + year +'.csv',
header=True, index=False)
else:
    print('Error: No tiene permisos para extraer ésta información según
https://www.expedia.es/robots.txt')


###################################################################
#####################  from_cities.py  ############################
###################################################################
from_cities = ['BILBAO','SAN SEBASTIAN']


###################################################################
#####################  to_cities.py  ##############################
###################################################################
to_cities = ['NEW YORK','LONDON HEATHROW']


###################################################################
#################  myconstants.py  ################################
###################################################################

wait_next_step = 1.5
wait_open_url = 5
wait_search = 5
```

DATASET

A continuación, mostramos el *dataset* obtenido de la extracción de datos mencionada en los apartados anteriores:



PVCM_04-2019_15-12-2019.csv: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
id_flight,name_departure_airport,name_arrival_airport,name_airline,duration,time_departure,time_arrival,number_stops,prices
0.0,BILBAO,NEW YORK,TAP Portugal,30 h 20 min,19:40,20:00,(1 escala),310 €
1.0,BILBAO,NEW YORK,Turkish Airlines,24 h 50 min,16:55,11:45,(1 escala),466 €
2.0,BILBAO,NEW YORK,Turkish Airlines,31 h 15 min,16:55,18:10,(1 escala),466 €
3.0,BILBAO,NEW YORK,Aer Lingus,11 h 30 min,13:45,19:15,(1 escala),559 €
4.0,BILBAO,NEW YORK,Varias aerolíneas,22 h 25 min,18:50,11:15,(1 escala),691 €
5.0,BILBAO,NEW YORK,Varias aerolíneas,28 h 0 min,13:15,11:15,(1 escala),691 €
6.0,BILBAO,NEW YORK,Air Europa,11 h 10 min,12:45,17:55,(1 escala),1.776 €
7.0,BILBAO,NEW YORK,Air Europa,15 h 20 min,8:35,17:55,(1 escala),1.776 €
8.0,BILBAO,NEW YORK,Air France,11 h 35 min,10:20,15:55,(1 escala),1.823 €
9.0,BILBAO,NEW YORK,Delta,11 h 35 min,10:20,15:55,(1 escala),1.823 €
10.0,BILBAO,NEW YORK,Varias aerolíneas,11 h 35 min,10:20,15:55,(1 escala),1.823 €
11.0,BILBAO,NEW YORK,Delta,14 h 35 min,10:20,18:55,(1 escala),1.823 €
12.0,BILBAO,NEW YORK,Air France,14 h 35 min,10:20,18:55,(1 escala),1.823 €
13.0,BILBAO,NEW YORK,Air France,14 h 40 min,7:15,15:55,(1 escala),1.823 €
14.0,BILBAO,NEW YORK,Delta,14 h 40 min,7:15,15:55,(1 escala),1.823 €
15.0,BILBAO,NEW YORK,Air France,16 h 30 min,10:20,20:50,(1 escala),1.823 €
16.0,BILBAO,NEW YORK,Delta,16 h 30 min,10:20,20:50,(1 escala),1.823 €
17.0,BILBAO,NEW YORK,Air France,17 h 40 min,7:15,18:55,(1 escala),1.823 €
18.0,BILBAO,NEW YORK,Air France,20 h 30 min,20:10,10:40,(1 escala),1.823 €
19.0,BILBAO,NEW YORK,Varias aerolíneas,11 h 15 min,13:20,18:35,(1 escala),1.825 €
20.0,BILBAO,NEW YORK,Finnair,12 h 45 min,8:15,15:00,(1 escala),1.825 €
21.0,BILBAO,NEW YORK,Varias aerolíneas,12 h 45 min,8:15,15:00,(1 escala),1.825 €
22.0,BILBAO,NEW YORK,Finnair,16 h 20 min,8:15,18:35,(1 escala),1.825 €
23.0,BILBAO,NEW YORK,American Airlines,15 h 15 min,13:20,22:35,(2 escalas),1.831 €
24.0,BILBAO,NEW YORK,American Airlines,15 h 15 min,13:20,22:35,(2 escalas),1.831 €
25.0,BILBAO,NEW YORK,British Airways,11 h 15 min,13:20,18:35,(1 escala),1.836 €
26.0,BILBAO,NEW YORK,Iberia,11 h 15 min,13:20,18:35,(1 escala),1.836 €
27.0,BILBAO,NEW YORK,British Airways,12 h 45 min,8:15,15:00,(1 escala),1.836 €
28.0,BILBAO,NEW YORK,Iberia,12 h 45 min,8:15,15:00,(1 escala),1.836 €
29.0,BILBAO,NEW YORK,British Airways,16 h 20 min,8:15,18:35,(1 escala),1.836 €
30.0,BILBAO,NEW YORK,Iberia,16 h 20 min,8:15,18:35,(1 escala),1.836 €
31.0,BILBAO,NEW YORK,British Airways,14 h 50 min,8:15,17:05,(2 escalas),1.842 €
32.0,BILBAO,NEW YORK,British Airways,15 h 25 min,13:20,22:45,(2 escalas),1.842 €
33.0,BILBAO,NEW YORK,British Airways,15 h 25 min,13:20,22:45,(2 escalas),1.842 €
34.0,BILBAO,NEW YORK,British Airways,15 h 25 min,13:20,22:45,(2 escalas),1.842 €
35.0,BILBAO,NEW YORK,Varias aerolíneas,12 h 45 min,6:55,13:40,(1 escala),1.954 €
36.0,BILBAO,NEW YORK,Delta,12 h 11 min,7:15,13:26,(1 escala),1.954 €
37.0,BILBAO,NEW YORK,Air France,12 h 11 min,7:15,13:26,(1 escala),1.954 €
38.0,BILBAO,NEW YORK,KLM,12 h 40 min,6:50,13:30,(1 escala),1.957 €
39.0,BILBAO,NEW YORK,Delta,12 h 40 min,6:50,13:30,(1 escala),1.957 €
40.0,BILBAO,NEW YORK,KLM,14 h 55 min,6:50,15:45,(1 escala),1.957 €
41.0,BILBAO,NEW YORK,Delta,14 h 55 min,6:50,15:45,(1 escala),1.957 €
42.0,BILBAO,NEW YORK,Delta,16 h 25 min,6:50,17:15,(1 escala),1.957 €
```

Windows (CRLF) Línea 1, columna 1 100%

CONTRIBUCIONES DE INTEGRANTES

A continuación, se muestra las contribuciones realizadas por cada uno de los integrantes del grupo:

Contribuciones	Firma
Investigación previa	B., J.A.
Redacción de las respuestas	B., J.A.
Desarrollo código	B., J.A.

REFERENCIAS

- https://www.expedia.es
- https://www.draw.io/
- Libro: “Minería de datos: Modelos y algoritmos”: Jordi Gironés Roig, Jordi Casas Roma, Julià Minguillón Alfonso, Ramon Caihuelas Quiles.