Práctica 1: Web Scrapping

Integrantes

- Richard Jácome Guanoluisa
- Andrea Martínez Espinosa

Esta práctica se encuentra disponible en el siguiente link:
https://github.com/aemartineze/Web_Scrapping_MTA_Data

1 Contexto

Realizar el web Scrapping de la información de los torniquetes administrados por la MTA (Metropolitan Transportation Authority). La página web contiene archivos semanales con la información de los valores registrados y acumulados de ingresos y salidas en los torniquetes administrados por la MTA (Metropolitan Transportation Authority), así como la descripción de la estación en la que se encuentran los torniquetes y las líneas que se pueden abordar desde la estación.

2 Título para el dataset

El titulo escogido para el dataset es: Registros_torniquetes_MTA

3 Descripción del dataset

Se generará un dataset que dispondrá de la información de las entradas y salidas de los torniquetes administrados por la MTA disponible en el link http://web.mta.info/developers/turnstile.html y se analizará el movimiento de las personas dentro de la red de transporte.

4 Representación gráfica

La información es recogida del conteo acumulado de los torniquetes de las estaciones del metro o trenes de Nueva York.

Figura 1. Torniquetes en Nueva York
Fuente: https://commons.wikimedia.org/wiki/File:PATH_33_Street_turnstiles_vc.jpg

5 Contenido

La descripción original de los campos según la MTA se encuentra en la web: http://web.mta.info/developers/resources/nyct/turnstile/ts_Field_Description.txt

Se disponen de 11 campos de acuerdo al siguiente detalle:

C/A = Área de Control
UNIT = Unidad remota para la estación
SCP = Posición del canal que representa la dirección específica del dispositivo
STATION = Nombre de la estación
LINENAME = Líneas que llevan a la estación
DIVISION = Línea original de la estación
DATE = Fecha (MM-DD-YY)
TIME = Hora (hh:mm:ss) para el evento de auditoría programado
DESC = Descripción del evento de auditoría
ENTRIES = Registro acumulativo de entradas
EXIST = Registro acumulativo de salidas

La información se encuentra disponible desde el 25 de mayo del 2010 hasta el 03 de abril de 2021 en archivos semanales almacenados en un enlace individual a un archivo .txt.

Para efectos de este trabajo se tomará la información mensual desde el año 2019 para analizar el efecto de la pandemia COVID-19 en los viajes realizados. Se tomará la información del primer día del mes para calcular el número de pasajeros mensuales en cada estación.

Para efectos de esta práctica se depuró la información de las columnas por lo que el dataset resultante Registros_torniquetes_MTA tiene los siguientes registros:

Id_Torniquete = Id único del torniquete (se compone de los campos originales C/A, UNIT, SCP)
Fecha = Fecha y hora (AAAA-MM-DD hh:mm:ss)
Estacion = Nombre de la estación
Linea = Líneas que llevan a la estación
Division = Línea original de la estación
Entradas = Registro acumulativo de entradas
Salidas = Registro acumulativo de salidas

6 Agradecimientos

La información ha sido recolectada desde la página web del MTA http://web.mta.info/developers/turnstile.html donde existen archivos semanales de extensión .txt con la información de los valores registrados y acumulados de ingresos y salidas en los torniquetes dentro de la zona de la ciudad de Nueva York a través de Long Island, sureste del estado de Nueva York y Connecticut. El organismo que publica la información es la MTA por lo cual se dispone de información gratuita, accesible y oficial.

7 Inspiración

Con este dataset se pretende responder el impacto en la movilidad de las personas de Nueva York debido a la pandemia COVID-19, es interesante tomar en cuenta el detalle del flujo de personas de una de las ciudades más pobladas del mundo para realizar contrastes con otras ciudades y de esta manera obtener conocimiento o indicar que comportamiento se está dando, que cambios se pueden hacer o evidenciar se hallan patrones en el flujo de acuerdo a un orden cronológico.

8 Licencia

La licencia que aplica para el dataset de esta práctica es: **CC BY-SA 4.0**, ya que el dataset deberá ser gratuito porque los datos son públicos, de esta manera facilitamos a usuarios el acceso sin permiso de nosotros (autores). Además, se podrá dar un enfoque comercial, crear derivaciones, esto siempre y cuando se referencie a los autores, se debe tomar en cuenta que si se puede distribuir este dataset y sus derivaciones, estas derivaciones tienen que ser distribuidas adjuntando una licencia igual a la que rige en el dataset original.

9 Código

Se adjunta el código en Python en el archivo código_dataset.py y en el Anexo 1.

10 Dataset

El dataset ha sido publicado en Zenodo en el link https://doi.org/10.5281/zenodo.4678313

11 Recursos

- Subirats, L., Calvo, M. (2018). Web Scraping. Editorial UOC.
- Creative Commons (2021). CC BY-SA 4.0. Disponible en: https://creativecommons.org/licenses/by-sa/4.0/deed.es

12 Contribuciones

Contribuciones	Firma
Investigación previa	Richard J. - Andrea M.
Redacción de las respuestas	Richard J. - Andrea M.
Código	Richard J. - Andrea M.

Anexo 1

```
#Importamos las librerias necesarias
from bs4 import BeautifulSoup
import requests
import pandas as pd
from datetime import datetime, timedelta
import time


#Dirección web de "The Metropolitan Transportation Authority" con las estadísticas de los pasajeros que pasan por los torniquetes del metro de NYC
dir_web = "http://web.mta.info/developers/turnstile.html"

#Utilizamos la libreria BeautifulSoup para acceder a la información
page = requests.get(dir_web)
soup = BeautifulSoup(page.text, "html.parser")

#Visualizamos todas las referencias de objetos en la página que en este caso son los archivos txt
#Guardamos en una lista
lista_txt =[]
for link in soup.find_all("a"):
   lista_txt.append(link.get('href'))

#Eliminamos valores nulos
lista_txt = [x for x in lista_txt if x]

#Filtramos los links de fechas
lista_txt = [x for x in lista_txt if 'turnstile_' in x]

#Generamos la lista con los links
lista_txt = ['http://web.mta.info/developers/'+x for x in lista_txt]


#Establecemos un rango de fecha para leer los archivos
fecha_inicial = datetime(2019,1,1)
fecha_final = datetime(2021,3,31)
rango_fechas = [(fecha_inicial + timedelta(days=d))
                for d in range((fecha_final - fecha_inicial).days + 1)]

#Filtramos solo los sábados
rango_fechas = [fecha.strftime("%y%m%d") for fecha in rango_fechas if fecha.weekday()  in [5]]

#Filtramos los primeros días de cada mes
s = pd.Series(pd.to_datetime(rango_fechas, format="%y%m%d"))
rango_fechas = s.groupby(s.dt.strftime('%y%m')).min().tolist()
rango_fechas = [dt.strftime('%y%m%d') for dt in rango_fechas]
```

```
#Filtrar la lista_txt_ en funcion del rango_fechas
lista_txt_final = [x for x in lista_txt if any(substring in x for substring in rango_fechas)]

df_mta = pd.DataFrame()
for fecha in lista_txt_final:
  aux = pd.read_csv(fecha)
  time.sleep(2)
  df_mta = pd.concat([df_mta,aux],ignore_index=True)

#Creamos una columna con el indicativo de cada torniquete
df_mta['Id_Torniquete'] = df_mta['C/A'] + '-' + df_mta['UNIT'] + '-' + df_mta['SCP']

#Unimos en una sola columna la fecha y la hora
df_mta['Fecha'] =  df_mta['DATE'] + " " + df_mta['TIME']
df_mta.Fecha = pd.to_datetime(df_mta.Fecha)

# limpiando los meses solo tomando el primer día de cada mes
df_mta = df_mta.loc[df_mta['Fecha'].dt.day == 1]


#Conservamos solo los valores Register de la columna DESC
df_mta = df_mta[df_mta.DESC != 'RECOVR AUD']

#Eliminamos las columnas que no son necesarias
df_mta.drop(['C/A', 'UNIT','SCP','TIME'], axis=1, inplace=True)

# creando dataframe para guardar lo limpio
frame_primera_hora = pd.DataFrame()
for i in df_mta.Id_Torniquete.unique():  # recorro cada Id unico
  torniquete = df_mta.loc[:,'Id_Torniquete'] == i
  torniquete_cl = df_mta.loc[torniquete]  # creo nuevo dataframe solo un torniquete id unico
  for j in df_mta.DATE.unique():  # recorro cada una de las 14 fechas (strings)
    fecha_torniquete = torniquete_cl.loc[:,'DATE'] == j
    fecha_torniquete_cl = torniquete_cl.loc[fecha_torniquete]
    try:  # cuando no hay dato en esa instancia de fecha
      fecha_corta = fecha_torniquete_cl.loc[:,'Fecha'] == min(fecha_torniquete_cl.Fecha)
      final = fecha_torniquete_cl.loc[fecha_corta]
    except ValueError:
      continue
    # concateno cada dataframe que voy sacando
    frame_primera_hora = frame_primera_hora.append(final, ignore_index=True)

#Eliminamos las columnas que no son necesarias
frame_primera_hora.drop(['DATE', 'DESC'], axis=1, inplace=True)

#Cambiamos los nombres de las columnas
Registros_torniquetes_MTA = frame_primera_hora
Registros_torniquetes_MTA.columns=['Estacion','Linea','Division', 'Entradas', 'Salidas', 'Id_Torniquete',
'Fecha']
```

```
#Cambiamos el orden de las columnas
Registros_torniquetes_MTA = Registros_torniquetes_MTA[['Id_Torniquete', 'Fecha', 'Estacion','Linea','Division','Entradas', 'Salidas']]

#Guardamos el dataset en uno
path = input("Ingrese el path para guardar el archivo ejemplo C:/Users/richa/Google Drive/TIPOLOGÍA Y CICLO DE VIDA DE LOS DATOS/PRACTICA 1 ")
Registros_torniquetes_MTA.to_csv(path + '/Registros_torniquetes_MTA.csv', header=True, index=False)
```