Introducción a la computación

Taller Nº 4

Fecha de entrega:

1. Problema

La Secretaría de Ambiente de la Ciudad Autónoma de Buenos Aires quiere hacer un estudio sobre la contaminación que hay en la ciudad. Dado que tenemos fama mundial en el tema de análisis de datos, nos convocan como consultores. Luego de firmar los contratos respectivos, les solicitamos los datos que debemos analizar y nos respondieron con el siguiente archivo.

Los datos incluidos en este archivo corresponden con distintas mediciones realizadas en estaciones meteorológicas EPA de la Ciudad Autónoma de Buenos Aires y, como el nombre del archivo lo indica, está en formato CSV (*Comma Separated Values*). Este tipo de archivos está organizado de manera tal de que tiene una serie de columnas con datos expresados como texto. El primer renglón del archivo tiene el *encabezado* que incluye los nombres de las columnas que nos pueden guiar para interpretar lo que incluyen (si son números, caracteres, etc).

El archivo en sí mismo no tiene información adicional para interpretar cada dato (habitualmente conocida como *metadata*), por lo que hay dos opciones: i) Alguien con el conocimiento nos dice qué hay en cada columna o ii) Lo tratamos de deducir mirando los nombres de las columnas y los valores que contiene.

Luego de pedir una reunión con el equipo técnico que nos contrató, nos dieron algo más de información. Lo primero es que la Ciudad de Buenos Aires cuenta con cuatro estaciones (sí solo cuatro) que miden distintos parámetros de contaminación: Monóxido de Carbono (CO), Óxidos de Nitrógeno (NO2) y Material Particulado respirable menor a 10 micrones (PM10).

Además, estas estaciones meteorológicas tienen sensores meteorológicos asociados que registran: Temperatura, Humedad, Presión atmosférica, Velocidad y dirección del viento, Radiación solar y Precipitaciones. Las cuatro estaciones se identifican como: Parque Centenario, Córdoba, La Boca y Palermo. Sus ubicaciones (coordenadas) se encuentran en el siguiente archivo.

El trabajo que nos encargaron es obtener un promedio semanal de óxido de nitrógeno (NO2) en las distintas estaciones meteorológicas. Luego, nos piden ubicar un punto (o *marker*) en un mapa de la ciudad cuyo tamaño sea proporcional al promedio semanal.

2. Preparando los datos

Lamentablemente los datos que tiene el archivo no tienen un formato cómodo para poder realizar el análisis pedido. Vamos a tener que procesar estos datos para poder utilizarlos.

Lo primero a hacer es leer el archivo, para esto podemos usar la biblioteca `csv` que con su función `csv.reader` permite leer un archivo de este tipo y acceder a su contenido fila por fila. La primer fila tiene los nombres de las columnas y las siguientes contienen los datos de las mediciones. Como podrán confirmar al inspeccionar el archivo, hay valores que están marcados como `s/d` que significa *sin datos* y otros que ni siquiera tienen un valor. Eventualmente cuando

se calculen los promedios, deberán incluir solo aquellos datos que contengan valores medidos, es decir, que representen un número. Lo siguiente que pueden ver es que el formato contiendo fecha y hora de la medición tiene una forma rara, la primer columna contiene el día y la segunda la hora.

Veamos las cosas que debemos hacer:

1. Lo primero que debemos hacer es generar una lista llamada `fechas` conteniendo la conversión de las dos primeras columnas al tipo de dato fecha-hora de `Python` (disponible en el módulo `datetime`. Deberán implementar la función `convertir_fecha` que tome el contenido de las dos primeras columnas (procesando fila por fila, es decir, recibe dos cadenas de caracteres), y devuelva la fecha-hora como valor (no como cadena de caracteres).

   La función `datetime.strptime('23/05/2010 15:45:00', '%d/%m/%Y %H:%M:%S')` sirve para esto es (en este ejemplo recibe una cadena de caracteres y la convierte a fecha-hora usando el formato indicado en la otra cadena). Deberán preparar la cadena de caracteres para utilizar en esta función en base a los datos que están en el archivo. El módulo `datetime` provee la función `replace` que permite actualizar cualquiera de los campos fecha-hora (por ejemplo, se puede cambiar la hora o el día una vez que se obtuvo la fecha con la función anterior). Luego, deberán procesar todas las filas del archivo agregando las fechas generadas a la lista `fechas`. Para controlar, esta lista deberá tener tantos elementos como filas-1 tiene el archivo. Un detalle importante es que hay horas que no están en el rango entre 0 y 23, en los casos en los que ocurra esto, deberán ajusta la fecha-hora para incrementar un día por cada 24 horas adicionales que indique la columna hora (por ejemplo, si la hora fuera 29, debería incrementarse un día la fecha leída y poner 5 en la hora). El módulo `datetime` tiene ofrece `timedelta` que permite crear una valor que luego se puede sumar a una fecha-hora directamente (la suma de `time` maneja los detalles de ajuste de meses, años, etc).

2. Una vez obtenida la lista con todas las fecha-hora de las mediciones, deberemos generar cuatro listas que llamaremos `NO2_centenario`, `NO2_cordoba`, `NO2_boca` y `NO2_palermo`. En estas listas deberán incluir los valores de cada medición haciendo la conversión de cadena de caracteres a número, siempre y cuando haya un dato válido, sino se pondrá el string `NaN`. La idea es tener cuatro listas que tengan la misma cantidad de elementos entre sí y que, a su vez, tengan la misma cantidad de elementos que la lista `fechas`. Se recomienda utilizar el mismo ciclo del punto anterior para hacer este procesamiento y aprovecharse del manejo de excepciones para capturar aquellas entradas que no tengan un número válido (recuerden `try...except...else`).

3. Para poder generar los promedios semanales, vamos a tener que ordenar la lista de fechas en forma creciente, pero para no perder relación con los valores de las mediciones, deberemos implementar una función que reciba las cinco listas y las ordene de acuerdo a la lista `fechas`. Es decir, al permutar elementos de la lista `fechas`, deberán realizar la misma permutación en el resto de las listas. Recomendamos modificar `upsort` para esto.

4. Para separar los datos semanalmente, va a ser necesario identificar cuáles fechas corresponden a una misma semana. Para eso, debemos definir una función `extraer_semanas()` que tome una lista de fechas y agrupe los índices de la `lista_fechas` por semana. Debe devolver una lista de listas, donde cada lista corresponda con los índices de cada semana. Las semanas empiezan en Lunes y las fechas previas al primer lunes deben descartarse.

5. Una vez ordenados los datos, tenemos que generar los promedios semanales. Para esto, vamos a programar una función que se llame `obtener_promedios` que tome la lista de fechas, la lista de semanas y las cuatro listas con datos de NO2 de las estaciones, y genere cinco listas. Una primera lista con las fechas de inicio de cada semana, y otras cuatro con los promedios semanales de cada semana. Estas cinco listas deben tener el mismo largo. Si para una semana no hubiera ninguna medición, deberán poner un 0 (no es lo mejor, pero no nos queremos complicar).

6. Para mostrar un mapa, se puede usar la biblioteca `folium` que permite mostrar una zona de cualquier lugar del mundo, agregarle marcadores para coordenadas de interés y después guardarlo como una página web. Tiene funcionalidades y plugins adicionales que te permiten hacer muchas cosas interesantes.

   La forma más sencilla de crear un mapa es usar la función `Map` que debe recibir como argumento una `location` que debe ser una lista con dos números que representan latitud y longitud del centro del mapa que se va a mostrar. En este caso, podrían usar el punto medio entre las estaciones meteorológicas. Otro parámetro importante es `zoom_start` que hace un acercamiento para que resulte práctica la vista inicial, recomendamos usar 16. Para agregar un marcador se puede usar la función `CircleMarker` que toma como argumento una `location` (en nuestro la ubicación de la estación meteorológica), `popup` y `tooltip` (que son texto que se mostrarán cuando se interactúa con el mouse sobre el marker), `radius` (que es el radio del círculo y es lo que debería ser proporcional al promedio semanal) y `color` (que permite elegir el color del marker). En este punto, deberemos generar un mapa con cuatro markers que muestren las ubicaciones de las cuatro estaciones meteorológicas con `tooltip` y `popup` que muestren el nombre de la estación correspondiente. Pueden guardar el mapa en un archivo `html` con la función `save` y verlo con un browser (`Firefox`, `Chrome`, el que tengan).

7. Una vez probada la generación del mapa, vamos a generar los mapas semanales de contaminación de óxido de nitrógeno en CABA. Para esto deberán colocar un marker en la ubicación de cada estación y su radio deberá ser proporcional al valor promedio de esa semana. Les recomendamos usar la media anual medida para usar como valor de referencia y hacer el marker más grande o más pequeño de acuerdo a la diferencia con esta media. Queda claro, entonces, que tendremos un mapa y un archivo html por cada semana. Si una estación no tuviera datos para una semana, se deberá elegir otro marcador (a elección) y la leyenda deberá ser ``N/A''.

8. Además de generar los mapas, nos interesa tener los gráficos de la variación del promedio de óxidos de nitrógeno en función del tiempo. Para esto deberemos graficarlo usando matplotlib. Deberán encontrar una manera de mostrar esta información de forma clara que permita comparar los valores de las distintas estaciones meteorológicas.

9. Finalmente, además de procesar los datos para mostrar el promedio semanal en el mapa, nos piden responder ¿cuál es la estación que tuvo más tiempo el sensor sin funcionar?

Condiciones de entrega:

- El grupo debe estar conformado por 2 o 3 estudiantes de la cohorte de los encuentros semanales.
- Se debe escribir c'odigo `Python` con la solución y con comentarios correspondientes. Se debe incluir el código de prueba realizado para verificar el funcionamiento.
- Se evaluará la correctitud del código producido, su claridad y legibilidad; y el uso de la herramienta `git`.
- Asegurarse de que su docente tenga permisos de lectura (*Reporter*) en el repositorio de `git`. Se descargará la última versión de los archivos directamente de ahí, luego de ser informados de que el taller se encuentra listo.

  Importante: Solo se admite la entrega por medio de `git`. No se aceptarán las entregas de código por mail u otro medio.