FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 1

[7506/9558] Organización de Datos
Curso 1
Primer cuatrimestre de 2021

Alumnos	Padrón
Elvis, Claros Castro	99879
Lerer, Joaquín	
Bach, Alan	
Reinaudo, Dante	102848

Índice

1. Introducción

El presente informe reúne la documentación de la solución del primer trabajo práctico de la materia Organización de Datos, el cual consiste en desarrollar un análisis exploratorio de un set de datos a partir de los conocimientos adquiridos en la materia. En particular, vamos a realizar el análisis sobre un set de datos con información acerca del impacto del terremoto Gorkha, ocurrido en Nepal durante el año 2015. Particularmente el dataset se enfoca en cómo eran las condiciones de una determinada vivienda y cuál fue su grado de daño luego del accidente.

2. Desarrollo

Para facilitar la lectura, a partir de este momento se hará referencia al set de datos como "DataFrame".

2.1. Preprocesamiento de los datos

Con el fin de optimizar el espacio utilizado y facilitar la futura investigación, se procedió a hacer una limpieza de los datos sobre los dos DataFrame involucrados, cada uno derivado de un archivo csv.

2.1.1. Columnas sin información

En primer lugar, se buscó si en los DataFrame existían columnas que no aporten ningún tipo de información o que tuvieran un único valor repetido a lo largo de todas sus filas, con el objetivo de descartarlas, ya que serían irrelevantes para la investigación e indiferente incluirlas. Luego se prosiguió a eliminar los valores nulos que pudieran existir en las columnas. Asombrosamente, en ninguno de los DataFrame analizados se encontraron columnas sin información, ni tampoco columnas con valores nulos.

2.1.2. Análisis del tipo de datos de las columnas

Con el objetivo de administrar el espacio utilizado, se analizó el valor de los tipos de datos de todas las columnas de nuestros dos DataFrame y se realizaron modificaciones sobre estos.

Al leer una columna del tipo entero con pandas, esta se almacena por default con un int64, este tipo de dato ocupa 64 bits para representar valores enteros. Dado que el lector de este informe, puede desconocer sobre temas relacionados a la informatica, cabe aclarar que en el sistema binario, con un número entero n de bits es posible representar valores de hasta 2^n. Si el valor de los números a guardar es pequeño, se puede optimizar bastante el espacio utilizado empleando un tipo de dato entero que use menos bits para la representación. Se procedió a buscar el valor máximo de cada una de las columnas numéricas, para saber con cuantos bits es posible representarlas.

Columna	Valor Máximo	Tipo de dato utilizado	Tipo de dato suficiente
building_id	1052934	Int64	Int32
geo_level_1_id	30	Int64	Int8
geo_level_2_id	1427	Int64	Int16
geo_level_3_id	12567	Int64	Int16
count_flours_pre_eq	9	Int64	Int8
age	995	Int64	Int16
area_percentage	100	Int64	Int8
height_percentage	32	Int64	Int8
count_families	9	Int64	Int8

Cuadro 1: Tipo de dato entero suficiente para almacenar la columna

Además, las columnas booleanas, inicialmente se almacenaron con un tipo de dato int64, pero fueron transformadas a un tipo de dato booleano (bool), para mejorar la perfomance de nuestro código.

Por último, las siguiente columnas fueron tratadas como tipo categórico (category):

- damage_grade
- land_surface_condition
- foundation_type
- roof_type
- ground_floor_type
- other_floor_type
- position
- plan_configuration
- legal_ownership_tatus

2.1.3. Limpieza de los datos

Luego de efectuar el análisis previo sobre los dos set de datos, se puede observar cómo se redujo considerablemente el espacio utilizado. Se pasó de trabajar con dos DataFrame, donde uno de ellos tenía un peso de 71.3 Mb, a trabajar con un unico DataFrame limpio con un peso de 11.4 Mb. Queda al descubierto la importancia de realizar una limpieza de los datos.

Tras las modificaciones mencionadas, se puede observar que el DataFrame cuenta con 40 columnas y 260601 filas, sumando un total de 10424040 celdas.

2.1.4.

1. Código Fuente

El código fuente se encuentra disponible en: https://github.com/DatosOrga2021/orga_datos_1c_2021

2. Módulos

2.1. Módulo Filter

A modo de poder realizar un buen análisis, es necesario tener una base de datos limpia y organizada de la cual extraer información. Para conseguir esto, se implementó un módulo capaz de modificar el *DataFrame* original con *filterf ull_correction(df)*. Ejecutar esta función, entre otras cosas:

1. Elimina las columnas “Last_Activity” y “Actual_Delivery_Date”, pues no tienen información.
2. Modifica el tipo de las columnas a *int64* o *datetime* si corresponde
3. Modifica el nombre de las columnas “TRF” y “Source ” a los indicados en el enunciado.
4. Convierte columnas categóricas de texto en categóricas numéricas, eliminando prefijos a la categoría.

Además de automatizar la corrección completa del *DataFrame*, se puede hacer un llamado a cada función individual para corregir solo ciertas columnas o inconvenientes.

2.2. Módulo Print

Para poder generar salidas de información útiles, presentables y organizadas, se implementó este módulo capaz de escribir a consola o a un archivo la información solicitada. A partir de sus funciones nos fue posible acelerar el proceso de generación de salidas y facilitar la lectura de ellas. Las funciones son:

Configuración

- *et_output(file)*: Borra el archivo *file*, y lo selecciona como salida.
- *eset_output()*: Selecciona a la salida estándar como salida.

Impresión

- *print_title(title)*: Imprime un título.
- *print_subtitle(subtitle)*: Imprime un subtítulo.
- *printf(text)*: Imprime un texto.
- *printt(text)*: Imprime un texto con indentación.
- *print_series(series)*: Imprime una serie.

Formateo

- *ewline()*: Imprime un salto de línea.
- *liv()*: Imprime un divisor.
- *pretty_f(float, n)*: Devuelve un string generado a partir de formatear el float con “n”.

3. Librerías Externas

3.1. Pandas

Pandas es una librería para análisis y manipulación de datos. Fue la principal herramienta utilizada para este proyecto, siendo la base de todos los análisis aquí presentados.

3.2. Numpy

Numpy es una librería para análisis numérico muy conocida. Dado su practicalidad estuvo involucrada de fondo en muchas operaciones que realizamos, así como también forma parte de otras de las librerías usadas.

3.3. Matplotlib

Matplotlib es una librería para creación de visualizaciones estáticas y animadas. Fue la principal herramienta utilizada para crear gráficos, partiendo de nuestro DataFrame que era modificado para llegar a información que le cargaríamos a los gráficos.

3.4. SeaBorn

Seaborn es una librería para visualización de datos estadísticos basada en Matplotlib. Fue utilizada para la creación de algunas visualizaciones más complejas, como los HeatMaps.

3.5. GeoPandas

GeoPandas es una librería para trabajar con datos geoespaciales. Fue utilizada para la creación de las visualizaciones de mapas. Para esto fue requerido modificar varios datos para adaptarse a las especificaciones de GeoPandas.

Referencias

[1] Benford's law. En.wikipedia.org. (2020). Retrieved from https://en.wikipedia.org/wiki/Benford

[2] GeoPandas 0.8.0. Geopandas.org. (2013). Retrieved from https://geopandas.org/.

[3] Hunter, J. (2007). Matplotlib: A 2D graphics environment. Matplotlib: Python plotting. Re- trieved from https://app.flourish.studio/visualisation/4347256/edit.

[4] Michael Waskom and the seaborn development team. (2020). mwaskom/seaborn. seaborn: statistical data visualization. Retrieved from https://seaborn.pydata.org/.

[5] NumPy. Numpy.org. (2019). Retrieved from https://numpy.org/.

[6] Regla de Sturges. Es.wikipedia.org. (2020). Retrieved from https://es.wikipedia.org/wiki/Regla_de_Sturges.

[7] The pandas development team. (2020). pandas - Python Data Analysis Library. Pan- das.pydata.org. Retrieved from https://pandas.pydata.org/.

[8] Mlxtend. http://rasbt.github.io/mlxtend/. En especial se utilizo http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/.

[9] Florecer https://app.flourish.studio/visualisation/4347256/ed it?