Laboratorio de Datos

Primer Cuatrimestre 2024

Práctica N° 6: Operaciones con DataFrames y transformaciones de datos.

Para estos ejercicios, usar el dataset `penguins`. En la mayoría de los ejercicios se pide realizar varias transformaciones en un DataFrame. En estos ejercicios, ver primero la forma de hacerlo "de cualquier manera", utilizando una o varias instrucciones. Luego, si es posible, hacerlo mediante una sola instrucción encadenando las operaciones.
En todos los ejercicios, cuando se pide alguna condición resolver el ejercicio implementando esa condición, no resolverlo a mano. Por ejemplo, si se pide una lista de los nombres de columnas de 14 caracteres, podemos hacer (ver **Ejercicio 4**)
`[col for col in penguins.columns if len(col) == 14]`,
no tenemos que ponernos a contar a mano cuántas letras tiene cada nombre.

1. Crear un subconjunto de datos que contenga sólo pingüinos de la isla Biscoe y que tengan un pico de 48 mm de largo o más.

   Sugerencia: recordar que para realizar operaciones lógicas coordenada a coordenada con arrays de numpy podemos usar los símbolos & (and) y | (or).

2. Crear otro dataset con la información de pingüinos Adelie machos que no sean de isla Biscoe.

3. Del dataset `penguins` quedarse con todas las variables excepto `year`, `sex` y `body_mass_g`.

   Sugerencia: utilizar el método `.drop()` de DataFrames.

4. Crear un subconjunto de los datos de `penguins` sólo con las obsevaciones de pingüinos machos con aletas (flipper) de más de 200 mm de largo y quedarse con todas las columnas que terminan con "mm".

   Sugerencias:

   - `penguins.columns` nos devuelve una lista con los nombres de las columnas.
   - Si queremos quedarnos con los elementos de una lista que cumplan una cierta propiedad, podemos hacer

     `[col for col in penguins.columns if ???]`

     reemplazando `???` por la condición. Esta forma de definir una lista se conoce como *list comprehension* (definir una lista por comprensión).
   - Pueden utilizar el método `endswith()` aplicado al string.

5. Empezando con `penguins`, crear un dataframe con los siguientes dos requisitos:

   (a) contenga sólo con las observaciones de la isla Dream.

   (b) contenga solo las variables `species` y todas las que empiecen con `bill`.

6. **Restablecer índices.**

   (a) En el dataset `penguins`, eliminar primero todas las filas con datos faltantes. ¿Qué sucede con los índices?

   (b) En el dataset sin datos faltantes, restablecer los índices mediante el comando `reset_index()`.

(c) ¿Cómo podemos hacer todo en un solo comando encadenando operaciones?

7. Renombrar columnas e índices. Para renombrar columnas utilizamos

`.rename(columns = ???)`

y para renombrar índices utilizamos

`.rename(index = ???)`.

Realizar las siguientes operaciones en el dataset `penguins`.

(a) Renombrar la variable `species` a `especies`. En este caso debemos pasarle a columns un diccionario: `{'variable_original' : 'variable nueva'}`.

(b) Renombrar en un solo `rename` la variable `body_mass_g` a `peso_g` y la variable `flipper_length_mm` a `aleta_mm`.

(c) Renombrar el índice 0 a 5000.

(d) Pasar todos los nombres de variables a mayúsculas. Sugerencia: en lugar de un diccionario, podemos pasarle a columnas una función. En este caso, podemos usar la función `str.upper`.

(e) ¿Qué resultado esperan del siguiente comando?

```
penguins.rename(index = np.sqrt)
```

(f) ¿Cómo podemos sumarle uno a todos los índices de `penguins`? Sugerencia: definir primero una función `suma_uno` y utilizar esta función al hacer `rename`.

(g) En Python, al igual que en muchos lenguajes, podemos usar *funciones lambda*, que nos permite crear funciones "al vuelo". ¿Qué resultado esperan del siguiente comando?

```
penguins.rename(index = lambda x : x * 2)
```

(h) ¿Cómo podemos usar funciones lambda para renombrar todos los nombres de columnas a mayúsculas?

8. pd.columns. También podemos renombrar columnas asignando una nueva lista de nombres mediante `penguins.columns = ???`. En este caso, resulta útil definir listas por comprensión como vimos en el **Ejercicio 4**. En el dataset `penguins`:

(a) Convertir todos los nombres de variables a mayúsculas.

(b) Convertir solo los nombres de variables que empiezan con `bill` a mayúsculas.

Sugerencias:

9. Si queremos definir una lista for comprensión aplicando distintas funciones podemos usar `if / else`. Por ejemplo, ¿cuál será la salida de la siguiente instrucción?

```
[x * 10 if x % 2 == 0 else x for x in [1,2,3,4,5,6]]
```

10. Empezando con `penguins` realizar las siguientes operaciones:

(a) Crear una nueva variable que tenga el peso en kg.

(b) Convertir la variable island a minúscula. Sugerencia: aplicar `.str.upper()` a la columna.

11. Empezando con `penguins` crear una tabla resumen que contenga el largo mínimo y máximo de las aletas de los pingüinos de la especie Adelie, agrupados por isla.

12. Empezando con `penguins`, agrupar los datos por especie y año, luego crear una tabla de resumen que contenga el ancho del pico (llamarla `bill_depth_mean`) y el largo del pico (llamarla `bill_length_mean`) para cada grupo

13. Empezando con `penguins`, hacer una secuencia de operaciones que:

- **(a)** Agregue una nueva columna llamada `bill_ratio` que sea el cociente entre el largo y el ancho del pico.
- **(b)** Quedarse sólo con las columnas `species` y `bill_ratio`.
- **(c)** Agrupar los datos por especie.
- **(d)** Crear una tabla de resumen que contenga el promedio de la variable `bill_ratio` por especie y que el nombre de la columna en la tabla sea `bill_ratio_mean`.

14. Usar el método `rename()` de DataFrames para cambiarle el nombre a la variable `body_mass_g` y llamarla `masa_corporal_g`.

15. Agregar una columna a `penguins` con la mediana de la masa corporal de los pingüinos de cada especie usando `group_by()` y `agg()`.

16. Empezando con `penguins`, escribir una secuencia de operaciones que:

- **(a)** Excluya a los pingüinos observados en la isla Biscoe.
- **(b)** Sólo se quede con las variables que están entre `species` y `body_mass_g` inclusive.
- **(c)** Renombre la variable species a `especie_pinguino`.
- **(d)** Agrupe los datos por la variable `especie_pinguino`.
- **(e)** Calcule el valor medio de las variables que contienen el string "length", separando por la especie del pingüino, y llamando a las columnas como las originales pero agregando "_mean" al final.

17. Empezando con `penguins`, contar cuántas observaciones hay por especie, isla y año.

18. Empezando con `penguins`, quedarse sólo con los pingüinos de las especies Adelie y Gentoo. Luego contar cuántos hay por cada especie y sexo.

19. Agregar una nueva columna a la base de datos llamada `peso_bin` que contenga:

- "chico" si la masa corporal es menos que 4000 gramos.
- "grande" si la masa corporal es mayor que 4000 gramos.

20. Empezando con `penguins`,

- **(a)** Quedarse sólo con las observaciones correspondientes a pingüinos Chinstrap.
- **(b)** Luego, quedarse sólo con las variables `flipper_length_mm` y `body_mass_g`.
- **(c)** Agregar una nueva columna llamada `fm_ratio` que contenga el cociente entre el largo de la aleta y el peso del pingüino.
- **(d)** Luego quedarse solo con las observaciones que no tienen NaN en ninguna columna (ayuda: `dropna()`)
- **(e)** Agregar otra columna llamada `ratio_bin` que contenga la palabra "alto" si `fm_ratio` es mayor o igual que 0.05 y "bajo" si el cociente es menor que 0.05.

Limpieza de datos. En los siguientes ejercicios, utilizamos el dataset `macro_full_columns.csv`.

21. Queremos arreglar los nombres de algunas columnas y eliminar columnas inútiles.

- **(a)** Cargar el archivo en un DataFrame `macroFull` utilizando la columna `anio` como index.
- **(b)** Listar todos los nombres de columnas, y eliminar del DataFrame la columna `Unnamed: 0`.
- **(c)** Observamos que algunas columnas terminan con el prefijo `vari_Porc` y otras con el prefijo `variPorc`. Cambiar el final de todas las columnas terminadas en `vari_Porc` a `variPorc`.
- **(d)** Modificar también todos los nombres de columnas terminados en `_Per_Cap` a `_perCap`.

22. Datos faltantes.

- **(a)** ¿En qué columnas hay datos faltantes? Podemos usar `df.isnull().any(axis = ???)`. ¿Cómo podemos generar una lista que tenga solamente los nombres de las columnas con datos faltantes?
- **(b)** ¿En qué años hay datos faltantes? Listar todos los años con datos faltantes.
- **(c)** Convertir todos los datos faltantes a 0.

23. Variables de oferta.

- **(a)** Generar un DataFrame que contenga solo las variables que terminan con "oferta".
- **(b)** Queremos explicar la variable `PBI_a_precios_de_mercado.oferta` utilizando el resto de las variables de oferta. Crear un DataFrame X que contenga todas las variables de oferta excepto la de PBI y una Seris y que contenga solo esa variable.
- **(c)** Ajustar un modelo de regresión lineal ordinaria o Ridge y definir el vector de predicciones.
- **(d)** Graficar, en un mismo gráfico, la variable respuesta original y la predicha en función del año. Sugerencia: prestar atención a los índices de cada variable.

24.

- **(a)** Quedarse sólo con las observaciones correspondientes a pingüinos Chinstrap.
- **(b)** Luego, quedarse sólo con las variables `flipper_length_mm` y `body_mass_g`.
- **(c)** Agregar una nueva columna llamada `fm_ratio` que contenga el cociente entre el largo de la aleta y el peso del pingüino.
- **(d)** Luego quedarse solo con las observaciones que no tienen NaN en ninguna columna (ayuda: `dropna()`)
- **(e)** Agregar otra columna llamada `ratio_bin` que contenga la palabra "alto" si `fm_ratio` es mayor o igual que 0.05 y "bajo" si el cociente es menor que 0.05.