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 6**)
`[col for col in penguins.columns if len(col) == 14]`,
no tenemos que ponernos a contar a mano cuántas letras tiene cada nombre.

1. A partir del dataset `penguins`, 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. **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?

5. **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 `flipper_length_mm` a `aleta_mm` y la variable `body_mass_g` a `peso_g`.

   (c) Renombrar el índice 0 a "cero".

   (d) Pasar todos los nombres de variables a mayúsculas. Sugerencia: en lugar de un diccionario, podemos pasarle a `columns` 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 una función lambda para renombrar todos los nombres de columnas a mayúsculas?

6. df.columns. También podemos renombrar columnas asignando una nueva lista de nombres mediante `penguins.columns = ???`. En este caso, resulta útil definir *listas por comprensión*. La sintaxis es `[f(x) for x in LIST]`.

(a) ¿Qué resultado esperan de ejecutar este comando?

```
[saludo + '!' for saludo in ['hola', 'chau']]
```

(b) En el dataset `penguins`, convertir todos los nombres de variables a mayúsculas utilizando listas por comprensión.

7. En el dataset `penguins`, convertir solo los nombres de variables que empiezan con `bill` a mayúsculas.

Sugerencia: Si queremos definir una lista for comprensión aplicando distintas funciones podemos crear una función partida utilizando `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]]
```

Nota: la función que utilizamos para definir listas por comprensión es una función *lambda* (pero no necesitamos escribir `lambda`). Esta sintaxis para definir funciones partidas es parte de la sintaxis de las funciones lambda.

8. Utilizando una función lambda, 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.

9. 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:

- 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.
- Pueden utilizar el método `endswith()` aplicado al string.

10. 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`.

11. 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.

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

13. 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.

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

14. 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`.

15. 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.

16. Variables de oferta.

(a) Generar un DataFrame que contenga solo las variables que terminan con `.oferta`. ¿Había datos faltantes en estas columnas?

(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 Series `y` que contenga solo la variable de PBI.

(c) Ajustar un modelo de regresión lineal ordinaria o Ridge y generar el vector de predicciones (hacerlo sobre el conjunto de todos los datos, no es necesario separar en entrenamiento y testeo).

(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.