

# **"Ciencia de datos aplicada al estudio de la Obesidad y otras enfermedades crónicas en Córdoba"**

*Práctico N°2 Curación de Datos.*

*Integrantes del grupo X:*

*Basmadjian, Osvaldo Martín*

*Fernández, María Emilia*

*Romero, Fernando*

**1) Verificar los tipos de datos detectados por pandas. Asegurar que el tipo de datos sea correcto para las variables que son continuas y aquellas que son categóricas. En el caso de no coincidir se deberán modificar la naturaleza de esas variables.**

Consideramos que para realizar un análisis más eficiente de los diferentes tipos de datos de la base de datos es conveniente analizar cada tipo de dato por separado. Para ello definimos, una función que nos permite extraer todas las columnas para un tipo de dato determinado, permitiendo un análisis más específico.

int16
int8
category
object
float64
datetime64[ns]
float32

**Tabla 1.** Tipos de datos únicos.

De la Tabla 1, podemos observar que sobre el dataset se detectaron 7 tipos de datos únicos, trabajaremos sobre ellos individualmente.

### 1.1) Tipo int8

Teniendo en cuenta que una variable de tipo signada puede representar valores entre -128 y 127, realizamos un análisis para determinar si alguna entrada del dataset contaba con valores negativos. Cuando no se encontraron valores negativos, realizamos el casteo del tipo de dato de int8 a uint8, que se podría utilizar para representar valores de 0 a 255, y que al no existir valores negativos, representan mejor el tipo de variable. Además, verificamos que la edad sea mayor a 18 en todos los registros.

### 1.2) Tipo int16

Para este tipo de dato aplicamos un criterio similar al descrito en el apartado anterior (int8). En primer lugar, verificamos que no existan valores negativos, y luego convertimos los datos a tipo uint16. Sobre estos datos, aquellas que no contenían valores mayores a 256, los re-convertimos a tipo uint8, reduciendo a la mitad el almacenamiento en memoria.

### 1.3) Tipo float32

El tipo float se caracteriza por ser capaz de representar datos con valores decimales. Para ello verificamos si las variables cuentan con valores decimales. De caso contrario se podrán convertir aplicando alguno de los criterios anteriores.

Haciendo la resta en la variable de tipo float32 y su versión de int32, si el resultado es cero, significa que la variable puede ser casteada a su versión entera sin punto decimal. A su vez, podemos reajustar el tipo de dato de las variables.

Para las variables dni e id, encontramos que el tipo de dato de mejor se ajusta es uint16 y finalmente para cod\_enc una uint8 es lo que mejor se ajusta.

#### 1.4) Tipo float64

Para las variables con este tipo de dato, encontramos que tiene valores NaN, y que Pandas utiliza, por defecto, float64, hasta no eliminar los valores NaN, no se podrá hacer ningún tipo de casteo con estas variables.

#### 1.5) Tipo Category

Sobre este tipo de variable, creemos que no hay tipo de conversión que se pueda hacer por el momento, solo tener presente que pueden contener valores nulos o inválidos

#### 1.6) Tipo Object

Aquí encontramos que hay existen variables que podrían ser representadas con otro tipo de datos. Al ser de tipo object, los métodos, min() y max(), no se pueden utilizar para determinar el rango de la variable. Entonces con una inspección visual de los valores, podemos determinar que la variable 'est' se encuentra mejor ajustada con un tipo uint8, mientras que peso, talla y esof con un float16, al contener valores con coma. Lo demás valores presentan valores como 'x' o '\*', por lo tanto quedan como están hasta avanzar con la curación.

#### 1.7) Tipo datetime

Como las fechas no se corresponden con una fecha reciente, elegimos pasar todas las variables a horas.



### 2) Verificar y asegurar que no existan datos duplicados.


Utilizando los métodos de pandas, identificamos que el dataset tiene 4298 entradas, y con el método duplicated() obtenemos la cantidad de ellas que están repetidas, en este caso 6. Luego el método drop\_duplicates(), nos genera un dataset con las filas repetidas que se han eliminado.

En todos los casos consideramos que las filas duplicadas son aquellas que tienen el mismo valor en las columnas de dni, dni\_, id y n\_encuesta.

### 3) Verificar que no existan caracteres "raros" para los datos de los features "ubic" y "cod\_encuestador".

Para verificar que no existan caracteres raros, utilizamos una librería de Python, llamada “string” que nos permite identificar aquellos caracteres ASCII imprimibles. Luego, tomamos como criterio que los caracteres raros son aquellos no imprimibles.

Reemplazamos los caracteres raros según los siguientes criterios:

- Todas las letras con tilde, se reemplazan con su versión sin tilde.
- Todas las “ñ”, se reemplazan por “nie” 

#### **4) Codificar las categorías de la variable "sexo", "ubic" y "nes" para su posterior uso en los próximos prácticos.**

Utilizamos para este paso el método LabelEncoder() de SciKitLearn para codificar las variables “sexo”, “ubic” y “nes”. Particularmente, como la variable “ubic” era de tipo “category”, la convertimos previamente al tipo string, y todos sus valores NaN, los pasamos a variables string con valor ‘NaN’, luego pudimos aplicar la transformación.

#### **5) Verificar espacios y normalizar los nombres de al menos 10 features que crean convenientes para un mejor entendimiento de su significado.**

Utilizamos la función .rename() de pandas para renombrar las variables que identificamos como susceptibles de ser normalizadas para un mejor entendimiento. A continuación se presentan los nombres originales de las variables y su normalización:

Original	Normalizado
tipo	act_fis_tipo
veces	act_fis_veces
riesgo	cat_cintura
eent	nivel_edu
Nombre	act_fis_nombre
tiempo	dbt_duracion
carga	act_fis_carga
ubic	ubic_tumor
imagen	imagen_juzgada
perc	imagen_percibida

**Tabla 2.** Variables cuyo nombre fue normalizado.

Cabe mencionar que renombramos cuatro variables referidas a la actividad física realizada por los encuestados, para las cuales utilizamos como primera parte del nombre "act\_fis\_" para poder identificarlas en su conjunto de una manera fácil si fuera necesario (“mismo prefijo”). De modo semejante, asignamos como primera parte del nombre "imagen\_" a las variables referidas a la imagen física de la persona encuestada.

## 6) Supongamos que necesitamos compartir públicamente todo el dataset y que los features "cod\_encuestador" y "dni" son sensibles y confidenciales.

Para despersonalizar los datos utilizamos los métodos de hashing **md5** para la variable "cod\_encuestador" y **sha256** para "dni\_" de la librería hashlib. El primero, convierte un string a una expresión única (fingerprint) de 128-bit como '0322599fc63702617f294d5f702e0a01', mientras que el segundo a una expresión como 'bacf3b56f68bedaf9dbab5956e3ce642197b4966d447995fa1a34939d5c2f851'. La aplicación de cualquiera de estos métodos por separado sobre un mismo string siempre arrojará como output la misma expresión, lo cual es útil para poder evaluar la integridad de los datos originales de cada variable.

## 7) Analizar y normalizar los datos faltantes (nulos).

Utilizando la función isnull() y ordenando los outputs para cada variable de manera descendente, identificamos que aquellas con mayor cantidad de registros faltantes fueron "grupoedad", "fgins1" y "carga" con 100, **67** 55 y 4,54 % de valores nulos respectivamente.

**Análisis causa grupoedad:** creemos que es probable que haya sido una variable a completar a posteriori (luego de terminar la encuesta) para tener identificado el grupo etéreo al que pertenecían los encuestados, dado que la manifestación de la enfermedad "obesidad" podría tener implicaciones diferentes según el mismo.

Los valores nulos de la variable "grupoedad" fueron reemplazados por números del 0 al 10, representando cada uno de ellos intervalos consecutivos y fijos de 10 años **(desde 0 a 110 años)**

**Análisis causa "fgins1":** teniendo en cuenta que "fgins2" representa el consumo de grasas insaturadas en gramo y es una variable continua con una distribución no normal y que la variable "fgins1" posee tres categorías (bajo medio y alto) según la frecuencia de consumo de grasas insaturadas. Decidimos imputarle los valores categóricos faltantes a "fgins1" según si el valor que tomaba la observación en "fgins2" se encontraba entre los percentiles 0 - 33 (categoría 1), 33 - 66 (categoría 2) y 66 - 100 (categoría 3).


**Análisis causa "carga":** asumimos que faltaba una categoría para indicar que los individuos no realizan actividad física (o que podría haberse confundido con la categoría sin carga), como se indica para las demás variables referidas a actividad física.


Los valores nulos de la variable "carga", cuando las variables "actfis" y "veces" tomaban valor 1 (no realiza actividad física) fueron reemplazados por el valor **4** que de ahora en adelante significa que los individuos no realizan actividad física.


## 8) Aplicar reglas de integridad sobre los datos.

Construimos dos condiciones restrictivas para evaluar la integridad de los datos respecto a la variable cáncer. La primera, permite extraer del conjunto de datos aquellas entradas en

las que se indica la presencia de cáncer pero no se tiene ubicación para el mismo (variable: `has_cancer_but_no_ubic`). Mientras que la segunda permite extraer aquellas entradas para las que se indica ausencia de cáncer pero se indica una ubicación (`has_ubic_but_no_cancer`). La primera condición arrojó 7 registros y la segunda 9. Para resolver las inconsistencias utilizamos complementariamente otras tres variables del conjunto de datos que se refieren al diagnóstico de cáncer (`tratinst` que debería ser 999 cuando hay ausencia de tumor y, `tummal` y `tumben` que indican la presencia o ausencia de tumor maligno y benigno, respectivamente).

Para aquellas entradas para las que se indicó la presencia de cáncer pero no  ubicación, utilizando las variables complementarias, pudimos inferir que la mayoría se trataban efectivamente de casos de cáncer. Con lo cual, asignamos NaN en la variable ubicación de las entradas correspondientes (`set_to_cure.ubic[(set_to_cure.cancer == 2) & (set_to_cure.ubic == 21)] = 'NaN'`)

Particularmente en la entrada 2351, se indicaba ausencia de cáncer pero en ubicación se indicaba cuello uterino. Utilizando las variables complementarias inferimos que no se trata de un caso de cáncer, entonces asignamos “no tiene” (21 en la codificación)  a ese registro en la columna ubicación (`set_to_cure.loc[2351, 'ubic'] = 21`).

Aquellas entradas en las que se indicó ausencia de cáncer y una ubicación, fueron curadas considerando la definición de cáncer propuesta por el Instituto Nacional del Cáncer de Estados Unidos (la presencia de un tumor maligno se diagnostica como cáncer). Utilizando las variables complementarias `tummal` y `tumben` (además de `tratinst`), se confirmó que sólo uno de estos casos correspondía efectivamente a cáncer (1073), con lo cual, se asignó un 2 a esa entrada en la columna correspondiente a cáncer (presencia  de cáncer) (`set_to_cure.cancer[(set_to_cure.cancer == 1) & (set_to_cure.tummal == 2)] = 2`)

Por otro lado, encontramos que no existen registros para los meses de abril, mayo y junio, ni para la estación otoño (codificado como 2).

Construimos las condiciones que nos permitan curar errores en la correspondencia entre cada mes y su respectiva estación. Por ejemplo, si se tiene un registro para el mes de diciembre, las únicas estaciones que pueden corresponder son verano (codificado como 1) o primavera (codificado como 4). Entonces, a todos los registros que no cumplen con esta condición se les asignó un 1 en la columna correspondiente a estación (`set_to_cure.est[((set_to_cure.mes == 'diciembre') & (set_to_cure.est != 1)) | (set_to_cure.est != 4)] = 1`). Se realizó el mismo procedimiento para cada mes.

## 9) Outliers.

Curamos los outliers de las variables cuantitativas continuas “mets” (nivel de actividad física de los individuos), “fgr2” (ingesta de grasas consumidas por los individuos) e “imc” (el índice de masa corporal). Considerando que en el trabajo práctico anterior hallamos que ninguna de estas variables tiene distribución normal, utilizamos una fórmula que involucra la distancia de cada uno de los registros para estas variables con respecto a los percentiles 0.25 y 0.75 de la distribución. Específicamente, se identificaron los outliers con valores extremos muy altos del siguiente modo:

```
high_outliers = (set_to_cure[var]) >=
((set_to_cure[var].quantile(0.75)) + 1.5 * range)
```

de modo que se consideran outliers aquellos valores de la variable mayores a el percentil 0.75 en una vez y media el recorrido intercuartílico

De modo semejantes, los outliers con valores extremos muy bajos se identificaron como:

```
low_outliers = (set_to_cure[var]) <=
((set_to_cure[var].quantile(0.25)) - 1.5 * range)
```

de modo que se consideran outliers aquellos valores de la variable menores que el percentil 0.25 en una vez y media el recorrido intercuartílico

Luego de la identificación de outliers, los mismos fueron reemplazados por valores “NaN”, de modo que el resultado fue la eliminación de 154, 132 y 114 outliers para “mets”, “fgr2” e “imc”, respectivamente.

## 10) Guardar el dataset.

El nuevo dataset generado se guardó en formato .csv.