Limpieza de datos de National Health and Nutrition Survey para análisis estadístico

---

El presente notebook se centra en la limpieza y normalización de las variables para crear el dataset posteriormente utilizado en el archivo Practica_estadistica. Los datos han sido extraidos de la web de NHANES, en la cual se han seleccionado los periodos desde 2007 hasta 2023.

Como el objetivo es poder crear un modelo que prediga el nivel de depresion, los dataset escogidos para cada periodo han sido:

- *Demographic Variables and Sample Weights*
- *Mental Health - Depression Screener*
- *Phisical Activity*
- *Sleep Disorders*

0. Importaciones necesarias

In [1]:
```
import pandas as pd
import numpy as np

import sys
from pathlib import Path

sys.path.insert(0, str(Path("..").resolve()))

import src.transformaciones as tr
```

```
Funciones ejecutadas correctamente
```

1. Carga y visualización de los datos

Para optimizar el código y evitar cargar cada dataset de forma individual, se ha optado por crear un diccionario en el que se ha incluido el periodo representativo de cada archivo para usarlo posteriormente como columna. También se han seleccionado las columnas necesarias para la construcción de este dataset tras un previo análisis manual.

In [2]:
```
cycles = [
    {"cycle": "2007-2008", "phq": "../data/2007-2008/DPQ_E.xpt", "demo": "../dat
    {"cycle": "2009-2010", "phq": "../data/2009-2010/DPQ_F.xpt", "demo": "../dat
    {"cycle": "2011-2012", "phq": "../data/2011-2012/DPQ_G.xpt", "demo": "../dat
    {"cycle": "2013-2014", "phq": "../data/2013-2014/DPQ_H.xpt", "demo": "../dat
    {"cycle": "2015-2016", "phq": "../data/2015-2016/DPQ_I.xpt", "demo": "../dat
    {"cycle": "2017-2020", "phq": "../data/2017-2020/P_DPQ.xpt", "demo": "../dat
    {"cycle": "2021-2023", "phq": "../data/2021-2023/DPQ_L.xpt", "demo": "../dat

]
```

```
phq_cols = [
    "SEQN",
    "DPQ010",
    "DPQ020",
    "DPQ030",
    "DPQ040",
    "DPQ050",
    "DPQ060",
    "DPQ070",
    "DPQ080",
    "DPQ090",
    "DPQ100"
]

demo_cols = [
    "SEQN",
    "RIDAGEYR",
    "RIAGENDR",
    "RIDRETH1",
    "DMDEDUC2",
    "DMDMARTL",
    "DMDMARTZ",
    "INDFMPIR"
]

sleep_cols = [
    "SEQN",
    "SLD010H",
    "SLD012"
]

activity_cols = [
    "SEQN",
    "PAD800",
    "PAD820",
    "PAD675",
    "PAD660"

]

dfs = []


for c in cycles:
    # Cargar PHQ-9, demografía, sueñi y actividad física
    df_phq = pd.read_sas(c["phq"], format="xport")
    df_demo = pd.read_sas(c["demo"], format="xport")
    df_sleep = pd.read_sas(c["sleep"], format="xport")
    df_activity = pd.read_sas(c["activity"], format="xport")


    # Seleccionar solo columnas indicadas
    df_phq = df_phq[[col for col in phq_cols if col in df_phq.columns]]
    df_demo = df_demo[[col for col in demo_cols if col in df_demo.columns]]
    df_sleep = df_sleep[[col for col in sleep_cols if col in df_sleep.columns]]
    df_activity = df_activity[[col for col in activity_cols if col in df_activit

    # Merge por SEQN(id)
    df_merged = df_phq.merge(df_demo, on="SEQN", how="left")
```

```
    df_merged = df_merged.merge(df_sleep, on="SEQN", how="left")
    df_merged = df_merged.merge(df_activity, on="SEQN", how="left")

    # Añadir columna ciclo
    df_merged["ciclo"] = c["cycle"]

    # Guardar en la lista
    dfs.append(df_merged)


# Concatenar todos los ciclos
df_all = pd.concat(dfs, ignore_index=True)

print(df_all.shape)
```

```
(44931, 25)
```

In [3]:
```
df_all.head()
```

Out[3]:

	SEQN	DPQ010	DPQ020	DPQ030	DPQ040	DPQ050	
0	41475.0	1.000000e+00	5.397605e-79	1.000000e+00	1.000000e+00	1.000000e+00	1.0
1	41477.0	5.397605e-79	5.397605e-79	5.397605e-79	5.397605e-79	5.397605e-79	5.3
2	41479.0	5.397605e-79	5.397605e-79	5.397605e-79	5.397605e-79	5.397605e-79	5.3
3	41481.0	5.397605e-79	5.397605e-79	1.000000e+00	1.000000e+00	5.397605e-79	5.3
4	41482.0	9.000000e+00	1.000000e+00	1.000000e+00	5.397605e-79	5.397605e-79	5.3

5 rows × 25 columns

In [4]:
```
df_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 44931 entries, 0 to 44930
Data columns (total 25 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   SEQN      44931 non-null  float64
 1   DPQ010    40368 non-null  float64
 2   DPQ020    40360 non-null  float64
 3   DPQ030    40356 non-null  float64
 4   DPQ040    40349 non-null  float64
 5   DPQ050    40348 non-null  float64
 6   DPQ060    40342 non-null  float64
 7   DPQ070    40338 non-null  float64
 8   DPQ080    40337 non-null  float64
 9   DPQ090    40329 non-null  float64
 10  DPQ100    27609 non-null  float64
 11  RIDAGEYR  44931 non-null  float64
 12  RIAGENDR  44931 non-null  float64
 13  RIDRETH1  44931 non-null  float64
 14  DMDEDUC2  42755 non-null  float64
 15  DMDMARTL  28147 non-null  float64
 16  INDFMPIR  40111 non-null  float64
 17  SLD010H   23877 non-null  float64
 18  PAD675    15306 non-null  float64
 19  PAD660    8824 non-null   float64
 20  ciclo     44931 non-null  object
 21  SLD012    20865 non-null  float64
 22  DMDMARTZ  14607 non-null  float64
 23  PAD800    5031 non-null   float64
 24  PAD820    2877 non-null   float64
dtypes: float64(24), object(1)
memory usage: 8.6+ MB
```

2. Limpieza de valores no válidos y/o nulos

Tal y como se muestra en la exploración previa, una de las características de estos archivos es que presentan valores ínfimos los cuales no pertenecen a ninguna de las escalas indicadas en la documentación de los datos. Estos datos aparecen al leer archivos SAS en Python y representan un valor cercano a 0 o un error en la lectura de los datos.

A pesar de que en la documentación existe 0 como valor, no se han imputado como 0 ya que no se puede asegurar que sean datos válidos. Pasándolos a Nan, se evita crear sesgos.

En cuanto al orden de la limpieza se ha ido haciendo por las columnas correspondientes a cada fichero para una mayor facilidad de entendimiento. Además, la interpretación de valores correspondientes a 'prefiero no contestar' o 'desconocido' van variando entre los ficheros, por ejemplo, para el caso de las preguntas del test de Depresión son valores no contestados aquellos pertenecientes a 7,9,77 y 99.

In [5]:
```
# Reemplazar códigos inválidos por NaN para las columnas de phq
df_all[phq_cols] = df_all[phq_cols].replace([7, 9, 77, 99], np.nan)

# Reemplazar valores extremadamente pequeños por NaN
df_all[phq_cols] = df_all[phq_cols].map(lambda x: np.nan if abs(x)<1e-10 else x)
```

In [6]:
```
tr.calculo_nulos(df_all, df_all.columns)
```

```
Porcentaje de nulos de SEQN = 0.0
Porcentaje de nulos de DPQ010 = 76.16
Porcentaje de nulos de DPQ020 = 76.63
Porcentaje de nulos de DPQ030 = 64.0
Porcentaje de nulos de DPQ040 = 54.39
Porcentaje de nulos de DPQ050 = 76.45
Porcentaje de nulos de DPQ060 = 83.11
Porcentaje de nulos de DPQ070 = 83.38
Porcentaje de nulos de DPQ080 = 89.72
Porcentaje de nulos de DPQ090 = 96.36
Porcentaje de nulos de DPQ100 = 82.36
Porcentaje de nulos de RIDAGEYR = 0.0
Porcentaje de nulos de RIAGENDR = 0.0
Porcentaje de nulos de RIDRETH1 = 0.0
Porcentaje de nulos de DMDEDUC2 = 4.84
Porcentaje de nulos de DMDMARTL = 37.36
Porcentaje de nulos de INDFMPIR = 10.73
Porcentaje de nulos de SLD010H = 46.86
Porcentaje de nulos de PAD675 = 65.93
Porcentaje de nulos de PAD660 = 80.36
Porcentaje de nulos de ciclo = 0.0
Porcentaje de nulos de SLD012 = 53.56
Porcentaje de nulos de DMDMARTZ = 67.49
Porcentaje de nulos de PAD800 = 88.8
Porcentaje de nulos de PAD820 = 93.6
```

Los nombres de las columnas pueden ir variando a lo largo de los periodos para datasets que hacen referencia a lo mismo. Es por ello que se imputarn dichos valores en la columna que menos nulos tengan y se eliminará aquella que duplica la información y que presenta un mayor porcentaje de nulos.

De este modo, las columnas a mantener después de impurtar los valores son:

- DMDMARTL es equivalente a DMDMARTZ.
- PAD800 es equivalente a PAD675
- PAD820 es equivalente a PAD660
- SLD012 es equivalente a SLD010H

In [7]:
```
df_all["DMDMARTL"] = df_all["DMDMARTL"].fillna(df_all["DMDMARTZ"])
df_all["PAD675"] = df_all["PAD675"].fillna(df_all["PAD800"])
df_all["PAD660"] = df_all["PAD660"].fillna(df_all["PAD820"])
df_all["SLD010H"] = df_all["SLD010H"].fillna(df_all["SLD012"])
```

In [8]:
```
df_all.drop(columns=["DMDMARTZ", "PAD800", "PAD820", "SLD012" ], inplace=True)
```

In [9]:
```
tr.calculo_nulos(df_all, df_all.columns)
```

```
Porcentaje de nulos de SEQN = 0.0
Porcentaje de nulos de DPQ010 = 76.16
Porcentaje de nulos de DPQ020 = 76.63
Porcentaje de nulos de DPQ030 = 64.0
Porcentaje de nulos de DPQ040 = 54.39
Porcentaje de nulos de DPQ050 = 76.45
Porcentaje de nulos de DPQ060 = 83.11
Porcentaje de nulos de DPQ070 = 83.38
Porcentaje de nulos de DPQ080 = 89.72
Porcentaje de nulos de DPQ090 = 96.36
Porcentaje de nulos de DPQ100 = 82.36
Porcentaje de nulos de RIDAGEYR = 0.0
Porcentaje de nulos de RIAGENDR = 0.0
Porcentaje de nulos de RIDRETH1 = 0.0
Porcentaje de nulos de DMDEDUC2 = 4.84
Porcentaje de nulos de DMDMARTL = 4.85
Porcentaje de nulos de INDFMPIR = 10.73
Porcentaje de nulos de SLD010H = 0.42
Porcentaje de nulos de PAD675 = 54.74
Porcentaje de nulos de PAD660 = 73.96
Porcentaje de nulos de ciclo = 0.0
```

In [10]:
```
df_all.shape
```

Out[10]:
```
(44931, 21)
```

Siguiendo con la limpieza de nulos perteneciente a las columnas de las preguntas del test de Depresión se opta por eliminar todas aquellas que presentan valores nulos en todas las filas ya que no aportan ningún tipo de información.

In [11]:
```
preguntas_phq = ['DPQ010',
 'DPQ020',
 'DPQ030',
 'DPQ040',
 'DPQ050',
 'DPQ060',
 'DPQ070',
 'DPQ080',
 'DPQ090',
 'DPQ100']

# Eliminar las filas con nulos en todas las columnas del test PHQ_9
df = df_all.dropna(subset=preguntas_phq, how="all")

# Comprobamos de nuevo la cantidad de filas tras su eliminación
print(df.shape)
```

```
(27638, 21)
```

In [12]:
```
tr.calculo_nulos(df_all, df_all[preguntas_phq].columns)
```

```
Porcentaje de nulos de DPQ010 = 76.16
Porcentaje de nulos de DPQ020 = 76.63
Porcentaje de nulos de DPQ030 = 64.0
Porcentaje de nulos de DPQ040 = 54.39
Porcentaje de nulos de DPQ050 = 76.45
Porcentaje de nulos de DPQ060 = 83.11
Porcentaje de nulos de DPQ070 = 83.38
Porcentaje de nulos de DPQ080 = 89.72
Porcentaje de nulos de DPQ090 = 96.36
Porcentaje de nulos de DPQ100 = 82.36
```

Pasando a la limpieza de los datos demográficos usando el mismo procedimiento, se observa que en este caso los datos no válidos corresponden a 77 y 99.

In [13]:
```
# Limpiamos las columnas de los datos demograficos, previa copia de seguridad
df = df.copy()

demo_cols = ["INDFMPIR","DMDEDUC2","DMDMARTL"]

df[demo_cols] = df[demo_cols].replace([77,99], np.nan)
df[demo_cols] = df[demo_cols].map(lambda x: np.nan if abs(x)<1e-10 else x)
```

Para las columnas representativas de Problemas de Sueño y Actividad Física se debe tener en cuenta también sus peculiaridades respecto a los valores no válidos que corresponden a 99.0 y 7777 y 9999 respectivamente.

In [14]:
```
# limpiamos las columnas de los datos de sueño y actividad
cols_sueño = ['SLD010H']
cols_act = ['PAD675','PAD660']

df[cols_sueño] = df[cols_sueño].replace([99.0], np.nan)
df[cols_sueño] = df[cols_sueño].map(lambda x: np.nan if abs(x)<1e-10 else x)

df[cols_act] = df[cols_act].replace([7777,9999], np.nan)
df[cols_act] = df[cols_act].map(lambda x: np.nan if abs(x)<1e-10 else x)
```

Centrándonos en la parte de actividad física existen dos columnas que representan tiempo dedicado a hacer ejercicio de forma moderada o intensa. Como el dato que interesa es saber los minutos que dedican al día y no su intensidad, se procede a sumar ambas variables para obtener una nueva y eliminamos las anteriores para evitar ruido.

In [15]:
```
# Se crea una variable que unifique el tiempo en minutos dedicado a hacer activi
df["min_actividad_diaria"] = np.where(
    df[["PAD675", "PAD660"]].isna().all(axis=1),
    np.nan,
    df["PAD675"].fillna(0) + df["PAD660"].fillna(0)
)

# Eliminamos las columnas que ya no nos interesan:
df.drop(columns=['PAD675','PAD660'], inplace=True)
```

In [16]:
```
df.isnull().mean() * 100
```

Out[16]:
```
SEQN                     0.000000
DPQ010                  61.238150
DPQ020                  62.012447
DPQ030                  41.468268
DPQ040                  25.852088
DPQ050                  61.715754
DPQ060                  72.545047
DPQ070                  72.979231
DPQ080                  83.280266
DPQ090                  94.076995
DPQ100                  71.325711
RIDAGEYR                 0.000000
RIAGENDR                 0.000000
RIDRETH1                 0.000000
DMDEDUC2                 5.546711
DMDMARTL                 5.608221
INDFMPIR                10.521745
SLD010H                  0.539113
ciclo                    0.000000
min_actividad_diaria    48.748100
dtype: float64
```

3. Creación target

Antes de continuar imputando o limpiando nulos, necesitamos obtener la columna objetivo.

Mediante la encuesta PHQ9 se consigue el valor numérico respecto al nivel de depresión que puede padecer una persona. Dicho dato se obtiene sumando las preguntas realizadas en ese test sobre sus percepciones a excepción de la columna DPQ100, que es una representación de cómo le afecta esa sintomatología.

Es por ello que para obtener el datos será necesario tener las columnas correspondientes a las respuestas respondidas.

Como solo hay 436 columnas completas al 100% se opta por imputar la media individual a aquellos valores faltantes siempre y cuando tengan completas 8 de las nueve preguntas, es decir, que al datos faltante de una columna de ese test se le imputará la media respecto a lo que haya completado en las otras preguntas.

In [17]:
```
# Seleccionamos las columnas que se usan para el calculo del PHQ_9 score.
phq = ['DPQ010',
 'DPQ020',
 'DPQ030',
 'DPQ040',
 'DPQ050',
 'DPQ060',
 'DPQ070',
 'DPQ080',
 'DPQ090']

# Número de filas con PHQ-9 completo
n_completas = df[phq].notna().all(axis=1).sum()

print("Filas con PHQ-9 completo:", n_completas)
```

```
Filas con PHQ-9 completo: 436
```

In [18]:
```
# Implementamos puntuaciones medias individuales para aquellos que han rellenado
# Para ello comprobamos primeros cuales son dichas filas y el resto las eliminam

df_valid = df.copy()

df_valid["phq_answered"] = df_valid[phq].notna().sum(axis=1)

df_valid = df_valid[df_valid["phq_answered"] >= 8].copy()

# Eliminamos la columna que indica el número de columnas respondidas
df_valid.drop(columns=["phq_answered"], inplace=True)


# Aplicamos la media y redondeamos porque los valores solo pueden ir de 0 a 3, s
df_valid[phq] = (df_valid[phq].apply(lambda row: row.fillna(row.mean()), axis=1)

# Obtenemos el target
df_valid["PHQ9_score"] = df_valid[phq].sum(axis=1)
```

4. Normalización de columnas

Se procede a cambiar los tipos de datos a enteros para aquellos que representan respuestas de 0 a 3 ó 6, según la pregunta. el índice que identifica a cada encuestado también pasa a ser un entero y la categoría ciclo pasa a ser 'category'.

Esto permite reflejar mejor la naturaleza de los datos, además de mejorar la consistencia y el uso de memoria.

In [19]:
```
df_valid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1529 entries, 14 to 44906
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   SEQN                  1529 non-null   float64
 1   DPQ010                1529 non-null   float64
 2   DPQ020                1529 non-null   float64
 3   DPQ030                1529 non-null   float64
 4   DPQ040                1529 non-null   float64
 5   DPQ050                1529 non-null   float64
 6   DPQ060                1529 non-null   float64
 7   DPQ070                1529 non-null   float64
 8   DPQ080                1529 non-null   float64
 9   DPQ090                1529 non-null   float64
 10  DPQ100                1369 non-null   float64
 11  RIDAGEYR              1529 non-null   float64
 12  RIAGENDR              1529 non-null   float64
 13  RIDRETH1              1529 non-null   float64
 14  DMDEDUC2              1433 non-null   float64
 15  DMDMARTL              1431 non-null   float64
 16  INDFMPIR              1347 non-null   float64
 17  SLD010H               1507 non-null   float64
 18  ciclo                 1529 non-null   object
 19  min_actividad_diaria  605 non-null    float64
 20  PHQ9_score            1529 non-null   float64
dtypes: float64(20), object(1)
memory usage: 262.8+ KB
```

In [20]:

```
columnas_int = phq_cols + ['RIDAGEYR', 'RIAGENDR', 'RIDRETH1', 'DMDEDUC2', 'DMDM
df_valid = tr.cambiar_tipos(df_valid,['ciclo'],columnas_int)
df_valid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1529 entries, 14 to 44906
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   SEQN                  1529 non-null   Int64
 1   DPQ010                1529 non-null   Int64
 2   DPQ020                1529 non-null   Int64
 3   DPQ030                1529 non-null   Int64
 4   DPQ040                1529 non-null   Int64
 5   DPQ050                1529 non-null   Int64
 6   DPQ060                1529 non-null   Int64
 7   DPQ070                1529 non-null   Int64
 8   DPQ080                1529 non-null   Int64
 9   DPQ090                1529 non-null   Int64
 10  DPQ100                1369 non-null   Int64
 11  RIDAGEYR              1529 non-null   Int64
 12  RIAGENDR              1529 non-null   Int64
 13  RIDRETH1              1529 non-null   Int64
 14  DMDEDUC2              1433 non-null   Int64
 15  DMDMARTL              1431 non-null   Int64
 16  INDFMPIR              1347 non-null   float64
 17  SLD010H               1507 non-null   float64
 18  ciclo                 1529 non-null   category
 19  min_actividad_diaria  605 non-null    float64
 20  PHQ9_score            1529 non-null   float64
dtypes: Int64(16), category(1), float64(4)
memory usage: 276.6 KB
```

El último paso antes de exportar el fichero es pasar las columnas a títulos que sean significativos para poder realizar el análisis.

In [21]:
```
df_valid=df_valid.rename(columns = {'SEQN': 'id',
                                    'DPQ010': 'perdida_interes',
                                    'DPQ020': 'estado_animo',
                                    'DPQ030': 'alteraciones_sueño',
                                    'DPQ040': 'fatiga',
                                    'DPQ050': 'cambios_apetito',
                                    'DPQ060': 'sentimiento_inutilidad/culpa',
                                     'DPQ070': 'dificultades_concentracion',
                                     'DPQ080': 'inquietud/lentitud_motora',
                                     'DPQ090': 'ideacion_suicida',
                                     'DPQ100': 'impacto_funcional_sintomas',
                                     'RIDAGEYR': 'edad',
                                     'RIAGENDR': 'genero',
                                     'RIDRETH1': 'etnia',
                                     'DMDEDUC2': 'educacion',
                                     'DMDMARTL': 'estado_civil',
                                     'INDFMPIR': 'indice_ingresos_familiares',
                                     'SLD010H': 'horas_sueño'
                                    })
```

In [22]:
```
df_valid.columns
```

Out[22]:
```
Index(['id', 'perdida_interes', 'estado_animo', 'alteraciones_sueño', 'fatiga',
       'cambios_apetito', 'sentimiento_inutilidad/culpa',
       'dificultades_concentracion', 'inquietud/lentitud_motora',
       'ideacion_suicida', 'impacto_funcional_sintomas', 'edad', 'genero',
       'etnia', 'educacion', 'estado_civil', 'indice_ingresos_familiares',
       'horas_sueño', 'ciclo', 'min_actividad_diaria', 'PHQ9_score'],
      dtype='object')
```

4. Exportación

El último paso antes de realizar el análisis estadístico es transportar los datos a CSV.

In [24]:
```
df_valid.to_csv("../data/nhanes_depression_dataset.csv", index = False)
```