Comprensión de los Datos

Emiliano Hervert de la Cruz | A01412606 | Carrera: IDM

In [1]:
```
#importa librerías
import pandas as pd
```

Descripción de Variables

A continuación se presentan todas las variables que tiene el dataset, así como el significado de cada una:

- **Pregnancies** - Número de veces embarazada
- **Glucose** - Concentración de glucosa plasmática a las 2 horas en una prueba de tolerancia a la glucosa oral
- **BloodPressure** - Presión arterial diastólica (mm Hg)
- **SkinThickness** - Espesor del pliegue cutáneo del tríceps (mm)
- **Insulin** - Insulina sérica de 2 horas (mu U/ml)
- **BMI** - Índice de masa corporal (peso en kg/(altura en m)^2)
- **DiabetesPedigreeFunction** - Función de pedigrí de diabetes
- **Age** - Edad (años)
- **Outcome** - Resultado (tiene diabetes o no)

De igual manera, es importante mencionar que todos los datos provienen de pacientes mujeres de al menos 21 años de edad y de ascendencia indígena Pima.

In [2]:
```
#Leer archivo csv con los datos
df = pd.read_csv("diabetes.csv")
```

In [3]:
```
#Verificar cuántas filas y columnas se tienen, respectivamente
df.shape
```

Out[3]:
```
(768, 9)
```

El dataset tienen 768 filas (incluyendo encabezados) y 9 columnas.

In [4]:
```
#Primeros 5 renglones del dataset
df.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunc
0	6	148	72	35	0	33.6	0
1	1	85	66	29	0	26.6	0
2	8	183	64	0	0	23.3	0
3	1	89	66	23	94	28.1	0
4	0	137	40	35	168	43.1	2

In [5]:
```
#Últimos 5 renglones del dataset
df.tail()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

In [6]:
```
#Revisa la información mas completa del conjunto de datos usando la función info()
#Muestra el total de datos, las columnas y su tipo correspondiente, dice si contien
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Como se puede ver, ninguno de los atributos tiene valores nulos, ya que 768 de las 768 filas son non-null. Asimismo, todas las columnas usan números, específicamente datos de tipo entero a excepción del *BMI* y *DiabetesPedigreeFunction* que usan flotantes.

In [7]:
```
#Revisar cuántos valores únicos tiene cada atributo del archivo
df.nunique()
```

Out[7]:
```
Pregnancies                 17
Glucose                    136
BloodPressure               47
SkinThickness               51
Insulin                    186
BMI                        248
DiabetesPedigreeFunction   517
Age                         52
Outcome                      2
dtype: int64
```

Variables Seleccionadas:

1. BMI
2. DiabetesPedigreeFunction
3. Outcome

In [8]:
```
#Nuevo dataframe solamente con mis variables seleccionadas
df_misvars = df[['BMI','DiabetesPedigreeFunction','Outcome']]
```

In [9]:
```
df_misvars.head()
```

Out[9]:

	BMI	DiabetesPedigreeFunction	Outcome
0	33.6	0.627	1
1	26.6	0.351	0
2	23.3	0.672	1
3	28.1	0.167	0
4	43.1	2.288	1

Exploración de Datos

In [10]:
```
#Estadísticas básicas de los atributos como cantidad de datos totales, su media, de
df_misvars.describe()
```

Out[10]:

	BMI	DiabetesPedigreeFunction	Outcome
count	768.000000	768.000000	768.000000
mean	31.992578	0.471876	0.348958
std	7.884160	0.331329	0.476951
min	0.000000	0.078000	0.000000
25%	27.300000	0.243750	0.000000
50%	32.000000	0.372500	0.000000
75%	36.600000	0.626250	1.000000
max	67.100000	2.420000	1.000000

In [11]:
```
#Revisar valores nulos con funcion isnull().sum()
df_misvars.isnull().sum()
```

Out[11]:
```
BMI                         0
DiabetesPedigreeFunction    0
Outcome                     0
dtype: int64
```

De nuevo, confirmamos que la suma total de los valores nulos para cada uno de estos atributos es 0.

In [12]:
```
#Revisar valores únicos por columna usando función unique(): nombre-columna.unique(
df_misvars.Outcome.unique()
```

Out[12]:
```
array([1, 0])
```

Los valores únicos para BMI y para DiabetesPedigreeFunction son muchos, ya que dependen de cada persona y no es común que coincidan. Sin embargo, aquí podemos ver que los valores únicos de Outcome son solo 0 y 1, es decir, si la paciente tiene o no Diabetes.

Variables Cuantitativas

Medidas de tendencia central

In [13]:
```
#DiabetesPedigreeFunction
#Se obtiene la media, mediana y moda
mean_pedigree = df_misvars['DiabetesPedigreeFunction'].mean()
median_pedigree = df_misvars['DiabetesPedigreeFunction'].median()
mode_pedigree = df_misvars['DiabetesPedigreeFunction'].mode()
print("Mean_pedigree:",mean_pedigree)
print("Median_pedigree:",median_pedigree)
print("Mode_pedigree:",mode_pedigree)
```

```
Mean_pedigree: 0.47187630208333325
Median_pedigree: 0.3725
Mode_pedigree: 0    0.254
1    0.258
Name: DiabetesPedigreeFunction, dtype: float64
```

In [14]:
```
#BMI
#Se obtiene la media, mediana y moda
mean_bmi = df_misvars['BMI'].mean()
median_bmi = df_misvars['BMI'].median()
mode_bmi = df_misvars['BMI'].mode()
print("Mean_bmi:",mean_bmi)
print("Median_bmi:",median_bmi)
print("Mode_bmi:",mode_bmi)
```

```
Mean_bmi: 31.992578124999998
Median_bmi: 32.0
Mode_bmi: 0    32.0
Name: BMI, dtype: float64
```

Conclusiones:

Función de pedigrí de diabetes:

1. El promedio fue 0.47188
2. El valor al centro es 0.3725
3. Los valores más repetidos fueron 0.254 y 0.258

Índice de masa corporal:

1. El BMI promedio fue 31.99258 (el promedio está dentro de la categoría de obesidad leve)
2. El BMI al centro es 32 (al menos la mitad de las personas está en la categoría de obesidad leve o más arriba)
3. El BMI más repetido fue 32

Variables Categóricas

In [15]:
```
#Para conteo  de cada valor en una columna, en orden descendente usar función value
df_misvars['Outcome'].value_counts()
```

Out[15]:
```
Outcome
0    500
1    268
Name: count, dtype: int64
```

De todos los datos, se obtiene que 500 personas no tienen diabetes, mientras que 268 sí tienen.

In [48]:
```
# Crear columna con variable Obesidad que clasifique como True a las personas que t
# Debido a una advertencia que dice "A value is trying to be set on a copy of a sli
```

```
df_misvars.loc[:,'Obesidad'] = df_misvars['BMI'] >= 30
```

In [49]:
```
df_misvars
```

Out[49]:

	BMI	DiabetesPedigreeFunction	Outcome	Obesidad
0	33.6	0.627	1	True
1	26.6	0.351	0	False
2	23.3	0.672	1	False
3	28.1	0.167	0	False
4	43.1	2.288	1	True
...	...	...	...	...
763	32.9	0.171	0	True
764	36.8	0.340	0	True
765	26.2	0.245	0	False
766	30.1	0.349	1	True
767	30.4	0.315	0	True

768 rows × 4 columns

Se agregó una nueva columna para poder categorizar fácilmente a las personas que tienen obesidad y a las que no.

Consulta

In [50]:
```
# Acceder a la primera fila
df_misvars.iloc[0]
```

Out[50]:
```
BMI                         33.6
DiabetesPedigreeFunction    0.627
Outcome                         1
Obesidad                     True
Name: 0, dtype: object
```

In [51]:
```
# Acceder a las dos primeras filas
df_misvars.iloc[:2]
```

Out[51]:

	BMI	DiabetesPedigreeFunction	Outcome	Obesidad
0	33.6	0.627	1	True
1	26.6	0.351	0	False

In [52]:
```
#Seleccionar columnas, indicando entre corchetes [nombreColumna, nombreColumna]
df_misvars[['BMI','DiabetesPedigreeFunction']]
```

Out[52]:

	BMI	DiabetesPedigreeFunction
0	33.6	0.627
1	26.6	0.351
2	23.3	0.672
3	28.1	0.167
4	43.1	2.288
...	...	...
763	32.9	0.171
764	36.8	0.340
765	26.2	0.245
766	30.1	0.349
767	30.4	0.315

768 rows × 2 columns

In [53]:
```
#Selección de filas [indicar dataframe[columna] operador valor]
personas_sin_infrapeso = df_misvars[df_misvars['BMI'] >= 18.5]
```

In [54]:
```
personas_sin_infrapeso
```

Out[54]:

	BMI	DiabetesPedigreeFunction	Outcome	Obesidad
0	33.6	0.627	1	True
1	26.6	0.351	0	False
2	23.3	0.672	1	False
3	28.1	0.167	0	False
4	43.1	2.288	1	True
...	...	...	...	...
763	32.9	0.171	0	True
764	36.8	0.340	0	True
765	26.2	0.245	0	False
766	30.1	0.349	1	True
767	30.4	0.315	0	True

753 rows × 4 columns

Aquí se hace una selección de solamente aquellas filas que NO contienen a pacientes con infrapeso (BMI < 18.5)

In [55]:
```
#ordenar usando funcion sort_values(by=atributo, ascending=True/false)
personas_sin_infrapeso.sort_values(by='DiabetesPedigreeFunction', ascending=True)
```

Out[55]:

	BMI	DiabetesPedigreeFunction	Outcome	Obesidad
268	25.1	0.078	0	False
180	23.2	0.084	0	False
149	27.3	0.085	0	False
567	32.0	0.085	0	True
135	33.8	0.088	0	True
...	...	...	...	...
45	42.0	1.893	1	True
370	38.4	2.137	1	True
4	43.1	2.288	1	True
228	36.7	2.329	0	True
445	59.4	2.420	1	True

753 rows × 4 columns

La selección de personas sin infrapeso es ordenada en base al valor de la función de pedigrí de diabetes de cada paciente, de manera ascendente.

In [56]:
```
#Agrupar por un atributo y calcular función de agregación utilizando groupby(atribu
personas_sin_infrapeso.groupby('Obesidad')['DiabetesPedigreeFunction'].mean()
```

Out[56]:
```
Obesidad
False    0.429267
True     0.498642
Name: DiabetesPedigreeFunction, dtype: float64
```

De esta selección de personas, se agrupan aquellos datos de pacientes con Obesidad y de aquellos sin Obesidad para obtener el promedio de la función de pedigrí de cada grupo.

Posteriormente, se consigue un subconjunto de las personas con su **valor de la función de pedigrí de diabetes** por **arriba del promedio**:

In [57]:
```
personas_con_pedigree_arribadelpromedio = df_misvars[df_misvars['DiabetesPedigreeFu
```

In [58]:
```
personas_con_pedigree_arribadelpromedio
```

Out[58]:

	BMI	DiabetesPedigreeFunction	Outcome	Obesidad
0	33.6	0.627	1	True
2	23.3	0.672	1	False
4	43.1	2.288	1	True
11	38.0	0.537	1	True
12	27.1	1.441	0	False
...	...	...	...	...
745	30.0	0.488	0	True
747	46.3	1.096	0	True
750	31.2	1.182	1	True
755	36.5	1.057	1	True
760	28.4	0.766	0	False

295 rows × 4 columns