Estadística Descriptiva — Pima Indians Diabetes

Autor: Fabiola Ochoa y Aaron Cuevas · **Fecha:** 2025-10-31

Objetivo de la actividad

Replicar los pasos de salón sobre el dataset `diabetes.csv` (**Pima Indians Diabetes**):

1. **Cargar** los datos.
2. Verificar **cantidad de datos** (filas/columnas) y **nombres de variables**.
3. Revisar **tipos de dato** e **identificar valores nulos**.
4. **Seleccionar** tres variables por integrante; aquí se analizan **Pregnancies**, **DiabetesPedigreeFunction** y **Outcome** (común para todos).
5. Para cada variable seleccionada: **tipo/subtipo**, **rangos (mín–máx)**, **media**, **mediana**, **desviación estándar (DE)** y **comentarios**.
6. Realizar **3 consultas** sobre los datos con las variables asignadas.

1. Carga de datos

In [ ]:
```
import pandas as pd, numpy as np
from pathlib import Path

candidates = [Path("data/diabetes.csv"), Path("diabetes.csv")]
for p in candidates:
    if p.exists():
        CSV_PATH = p
        break
else:
    raise FileNotFoundError("No se encontró diabetes.csv.")

df = pd.read_csv(CSV_PATH)

# Ceros imposibles como NA en columnas clínicas estándar
cols_zero_na = [c for c in ["Glucose","BloodPressure","SkinThickness","Insul
df[cols_zero_na] = df[cols_zero_na].replace(0, np.nan)

df.head()
```

2. Cantidad de datos y variables

In [ ]:
```
df.shape, df.columns.tolist()
```

3. Información general y valores nulos

In [ ]:
```
df.info()
print("\nValores nulos por columna:")
df.isna().sum()
```

4. Variables seleccionadas y tipología

- **Pregnancies**: *cuantitativa discreta* (conteo de embarazos; 0 es posible).
- **DiabetesPedigreeFunction (DPF)**: *cuantitativa continua* (índice de antecedentes familiares).
- **Outcome**: *categórica binaria* (0 = no diabetes, 1 = diabetes). Su media equivale a la prevalencia muestral.

5. Estadísticos descriptivos

In [ ]:
```
sel = ["Pregnancies","DiabetesPedigreeFunction","Outcome"]
stats = df.agg({
    "Pregnancies": ["min","max","mean","median","std"],
    "DiabetesPedigreeFunction": ["min","max","mean","median","std"],
    "Outcome": ["min","max","mean"]
})
stats
```

In [ ]:
```
def iqr(s):
    return s.quantile(0.75) - s.quantile(0.25)
pd.DataFrame({
    "Pregnancies_IQR":[iqr(df["Pregnancies"])],
    "DPF_IQR":[iqr(df["DiabetesPedigreeFunction"])]
})
```

6. Visualización

In [ ]:
```
import matplotlib.pyplot as plt
df["Pregnancies"].dropna().plot(kind="hist", bins=30, alpha=0.8, title="Hist
plt.xlabel("Pregnancies"); plt.ylabel("Frecuencia"); plt.show()

df["DiabetesPedigreeFunction"].dropna().plot(kind="hist", bins=30, alpha=0.8
plt.xlabel("DiabetesPedigreeFunction"); plt.ylabel("Frecuencia"); plt.show()

df["Outcome"].value_counts().sort_index().plot(kind="bar", title="Distribuci
plt.xlabel("Outcome"); plt.ylabel("Conteo"); plt.show()
```

7. Consultas

In [ ]:
```python
# Q1: Pacientes con ≥5 embarazos y Outcome=1
q1 = (df.query("Pregnancies >= 5 and Outcome == 1")
        [["Pregnancies","DiabetesPedigreeFunction","Outcome"]]
        .sort_values(["Pregnancies","DiabetesPedigreeFunction"], ascending=F
q1.head(10)
```

In [ ]:
```python
# Q2: Pr(Outcome=1) por categorías de Pregnancies
preg_bins = [0,1,3,5,10,100]
preg_labels = ["0","1–2","3–4","5–9","10+"]
q2 = (df.assign(preg_cat=pd.cut(df["Pregnancies"], bins=preg_bins, labels=pr
        .groupby("preg_cat")["Outcome"].mean()
        .rename("Pr(Outcome=1)")).to_frame()
q2
```

In [ ]:
```python
# Q3: Pr(Outcome=1) por cuartiles de DPF
q3 = (df.assign(dpf_q=pd.qcut(df["DiabetesPedigreeFunction"], q=4, duplicate
        .groupby("dpf_q")["Outcome"].mean()
        .rename("Pr(Outcome=1)")).to_frame()
q3
```

8. Conclusiones

a) Tipos y limpieza. `Pregnancies` es **discreta** (conteo), `DiabetesPedigreeFunction` es **continua** (índice) y `Outcome` es **binaria** (0/1). Los ceros imposibles se trataron como faltantes en variables clínicas no analizadas aquí (Glucose, BloodPressure, SkinThickness, Insulin, BMI), evitando sesgos en estadísticas agregadas.

b) Rango y tendencia central.

- `Pregnancies`: **mín 0**, **máx 17**, **media 3.85**, **mediana 3.00**, **DE 3.37**, **IQR 5.00**. La **media ≥ mediana** sugiere **asimetría a la derecha**: abundan valores bajos con una cola de conteos altos.
- `DiabetesPedigreeFunction`: **mín 0.078**, **máx 2.420**, **media 0.472**, **mediana 0.372**, **DE 0.331**, **IQR 0.382**. También asimétrica a la derecha: muchos valores pequeños y algunos grandes.
- `Outcome`: **prevalencia muestral ≈ 34.9%** (la media de `Outcome`).

c) Asociación empírica con Outcome.

- Por **categorías de embarazos** (`Pr(Outcome=1)`):
- 0: 0.272
- 1–2: 0.258
- 3–4: 0.352
- 5–9: 0.492
- 10+: 0.588 Se observa un **patrón creciente**: a mayor número de embarazos, mayor proporción de casos positivos.

- Por **cuartiles de DPF** (`Pr(Outcome=1)`):
- (0.077, 0.244]: 0.255
- (0.244, 0.372]: 0.333
- (0.372, 0.626]: 0.323
- (0.626, 2.42]: 0.484 La proporción **aumenta de cuartil en cuartil**, coherente con que valores altos del índice familiar se asocian a mayor riesgo.

d) Lectura integrada y límites.

- Ambas variables (`Pregnancies` , `DPF`) muestran relación **monótona creciente** con `Outcome` , pero de **magnitud moderada** (fenómeno multifactorial).
- Posibles **confusores** (no modelados aquí): edad, IMC y glucosa; podrían explicar parte de los incrementos observados.
- La prevalencia reportada es **muestral**; no debe interpretarse como tasa poblacional sin muestreo representativo.

e) Recomendación inmediata.

- Mantener estas variables en un modelo logístico base `Outcome ~ Pregnancies + DPF` y, si procede, ajustar por edad, IMC y glucosa para estimar efectos **parciales**.

Visualización y Análisis de Datos

Variables categóricas

In [ ]:
```
import matplotlib.pyplot as plt

# Barras para Outcome (única categórica estricta)
datos_outcome = df['Outcome'].value_counts().sort_index()
datos_outcome.plot(kind='bar')
plt.title('Distribución de Outcome (0=no, 1=sí)')
plt.xlabel('Outcome'); plt.ylabel('Conteo')
plt.show()
```

In [ ]:
```
import pandas as pd, numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
try:
    df
except NameError:
    import pathlib
    df = pd.read_csv(pathlib.Path('data/diabetes.csv'))
    for c in ['Glucose','BloodPressure','SkinThickness','Insulin','BMI']:
        if c in df.columns: df[c]=df[c].replace(0, np.nan)
```