codigo

September 4, 2025

0.1 Instituto Tecnológico y de Estudios Superiores de Monterrey

Análisis de métodos de razonamiento e incertidumbre (Gpo 102)

0.2 Actividad PBL 2

Profesor: Hugo Eduardo Ramírez Jaime

Realizado por:

- Diego Colin Reyes A01666354
- Aldo Reséndiz Cravioto A01625395
- Daniel Alejandro López Martínez A01770442
- Eduardo Ramírez Almanza A01660118

1 Carga de datos y librerias

[34]:
```
import pandas as pd
import itertools

data = pd.read_csv("COVID19MEXICO.csv")
data
```

[34]:
```
       FECHA_ACTUALIZACION ID_REGISTRO  ORIGEN  SECTOR  ENTIDAD_UM  SEXO  \
0               2025-07-29      167f1a       1      12           1     2
1               2025-07-29     ga8a474       1       4          20     2
2               2025-07-29     gb733da       1       6           8     2
3               2025-07-29     g9ff7b3       1       4          32     1
4               2025-07-29     g90b5c8       1       6          10     1
…                      …           …     …     …          …   …
99521           2025-07-29     g897fca       1       3          21     1
99522           2025-07-29     ge40110       1      15          16     2
99523           2025-07-29     g528417       1      15           2     1
99524           2025-07-29     gab8b98       1      12          19     2
99525           2025-07-29     g6709fc       1      15          32     1

       ENTIDAD_NAC  ENTIDAD_RES  MUNICIPIO_RES  TIPO_PACIENTE  …  \
0                1            1              3              1  …
1               20           20            413              1  …
```

```
2                8            8            37             1  …
3               32           32            17             2  …
4               10           10             5             2  …
…              …            …             …             …  …
99521           21           21            53             2  …
99522           15           16           102             2  …
99523            2            2             1             2  …
99524           19           19            44             1  …
99525           32           32            17             2  …

      RESULTADO_PCR RESULTADO_PCR_COINFECCION TOMA_MUESTRA_ANTIGENO  \
0               997                      997                     2
1               997                      997                     2
2               997                      997                     2
3                 5                        5                     2
4                 5                        5                     2
…               …                        …                     …
99521           999                      999                     2
99522           999                      999                     2
99523           999                      999                     2
99524           999                      999                     2
99525           999                      999                     2

       RESULTADO_ANTIGENO  CLASIFICACION_FINAL_COVID  CLASIFICACION_FINAL_FLU  \
0                      97                          6                        6
1                      97                          6                        6
2                      97                          6                        6
3                      97                          7                        7
4                      97                          7                        7
…                     …                          …                        …
99521                  97                          6                        6
99522                  97                          6                        6
99523                  97                          6                        6
99524                  97                          6                        6
99525                  97                          6                        6

       MIGRANTE  PAIS_NACIONALIDAD  PAIS_ORIGEN  UCI
0            99             México           97   97
1            99             México           97   97
2            99             México           97   97
3            99             México           97    2
4            99             México           97    2
…           …                …           …  …
99521        99             México           97    2
99522        99             México           97    2
99523        99             México           97    2
99524        99             México           97   97
```

```
99525        99             México           97    2

[99526 rows x 42 columns]
```

2 Descripción de la base de datos usada

En el contexto de la base de datos de la Secretaría de Salud de México, los códigos de valor son:

1: Sí

2: No

97, 98, 99: Desconocido o no aplica

Edad: Numero de la edad del paciente

NEUMONIA: Paciente que presentó o no diagnóstico de neumonía.

DIABETES: Paciente con un diagnóstico previo de diabetes.

EPOC: Paciente con un diagnóstico previo de enfermedad pulmonar obstructiva crónica.

ASMA: Paciente con un diagnóstico previo de asma.

INMUSUPR: Paciente con un diagnóstico previo de inmunosupresión.

HIPERTENSION: Paciente con un diagnóstico previo de hipertensión.

CARDIOVASCULAR: Paciente con un diagnóstico previo de enfermedad cardiovascular.

OBESIDAD: Paciente con un diagnóstico previo de obesidad.

RENAL_CRONICA: Paciente con un diagnóstico previo de enfermedad renal crónica.

TABAQUISMO: Paciente con un diagnóstico previo de tabaquismo.

OTRO_CASO: Indica si el paciente tuvo contacto con otro caso confirmado de COVID-19.

FECHA_DEF: Contiene la fecha del fallecimiento del paciente en caso de tener 9999-99-99 o algo parecido no ha muerto

Y la más Importante

2.0.1 CLASIFICACION_FINAL_COVID

3: Negativo a SARS-CoV-2: El caso fue descartado tras obtener un resultado de laboratorio negativo.

4: Caso Descartado por Dictaminación: El caso fue descartado por un comité de expertos sin una prueba de laboratorio concluyente.

5: Sospechoso: Es el estado inicial y transitorio de un caso, cuya clasificación final está pendiente de confirmación o descarte.

6: Confirmado por Laboratorio: El caso fue confirmado mediante una prueba de laboratorio, típicamente RT-PCR, el método de mayor fiabilidad.

7: Positivo por Dictaminación o Asociación: El caso fue catalogado como positivo por un comité de expertos basándose en un nexo epidemiológico con un caso confirmado o en criterios clínicos, sin una prueba de laboratorio concluyente.

2.0.2 CLASIFICACION_FINAL_FLU

(3): Caso confirmado

(7): Caso negativo

(4, 5 ó 6): Sin información (sin muestra o muestra inválida)

3 Limpieza

[35]:
```
# Seleccionando las columnas de interes
data = data[[
    "EDAD",
    "NEUMONIA",
    "DIABETES",
    "EPOC",
    "ASMA",
    "INMUSUPR",
    "HIPERTENSION",
    "CARDIOVASCULAR",
    "OBESIDAD",
    "RENAL_CRONICA",
    "TABAQUISMO",
    "OTRO_CASO",
    "FECHA_DEF",
    "CLASIFICACION_FINAL_COVID",
    "CLASIFICACION_FINAL_FLU"
]].copy()
# Lista de valores que representan datos faltantes
missing_codes = [97, 99, 997, 999, 9999, '97', '99', '997', '999', '9999',␣
 ↪'9999-99-99']

# Reemplazar por 0 en todo el DataFrame
data = data.replace(missing_codes, 0)

# Crear una columana adulto mayor para centrarse solo en esa la probabilidad
adulto_mayor = data["EDAD"].apply(
    lambda x: 1 if pd.notna(x) and x >= 60 else 2) #se toma 60 que es la edad␣
 ↪para ser adulto mayor en México
data.insert(1, 'AM', adulto_mayor)

# Esta funcion es para en lugar de tener fecha de muerte, tener si se murio o no
def convertir_no_cero_a_uno(valor):
    """Convierte valores no cero a 1, mantiene ceros como 2"""
```

```
    return 1 if valor != 0 else 2

muertos = data["FECHA_DEF"]
data["FECHA_DEF"] = muertos.apply(convertir_no_cero_a_uno)

def crear_columna_enfermo(df):
    df = df.copy()

    df['ENFERMO'] = (
        (df['CLASIFICACION_FINAL_COVID'].isin([6, 7])) |  # COVID positivo
        (df['CLASIFICACION_FINAL_FLU'] == 3)              # Influenza positivo
    )
    # Igual usamos 1 para casos positivos y 2 para negativos
    df['ENFERMO'] = df['ENFERMO'].map({True: 1, False: 2})

    return df
data = crear_columna_enfermo(data)

data
```

[35]:

	EDAD	AM	NEUMONIA	DIABETES	EPOC	ASMA	INMUSUPR	HIPERTENSION	\
0	8	2	2	2	2	2	2	2	
1	23	2	2	2	2	2	2	2	
2	18	2	2	2	2	1	2	2	
3	24	2	2	2	2	1	2	2	
4	47	2	2	2	2	2	2	2	
…	…	..	…	…	…	…	…	…	
99521	83	1	2	2	2	2	2	1	
99522	50	2	1	1	2	2	2	1	
99523	13	2	2	2	2	2	2	2	
99524	37	2	2	2	2	2	2	2	
99525	3	2	1	2	2	2	2	2	

	CARDIOVASCULAR	OBESIDAD	RENAL_CRONICA	TABAQUISMO	OTRO_CASO	\
0	2	2	2	2	2	
1	2	2	2	2	2	
2	2	2	2	2	2	
3	2	2	2	2	2	
4	2	2	2	2	2	
…	…	…	…	…	…	
99521	2	2	2	2	2	
99522	2	2	1	2	1	
99523	2	2	2	2	2	
99524	2	2	2	2	1	
99525	2	2	2	2	2	

	FECHA_DEF	CLASIFICACION_FINAL_COVID	CLASIFICACION_FINAL_FLU	ENFERMO

```
0              2                       6                       6        1
1              2                       6                       6        1
2              2                       6                       6        1
3              2                       7                       7        1
4              2                       7                       7        1
…          …                       …                       …       …
99521          2                       6                       6        1
99522          2                       6                       6        1
99523          2                       6                       6        1
99524          2                       6                       6        1
99525          2                       6                       6        1

[99526 rows x 17 columns]
```

4 Enfermedades/Sintomas mas mortíferos

[36]:
```python
s = data["ENFERMO"].value_counts()
total_poblacion = data["ENFERMO"].count()
print(f"{s}, total de población: {total_poblacion} ")
sintomas = ["AM", "NEUMONIA", "DIABETES", "EPOC", "ASMA",
            "INMUSUPR", "HIPERTENSION", "CARDIOVASCULAR",
            "OBESIDAD", "RENAL_CRONICA", "TABAQUISMO", "OTRO_CASO"]

# Filtrar pacientes muertos
muertos = data[data['FECHA_DEF'] == 1]

# Contar cuántos tienen todos los síntomas = 0
todos_0 = muertos[(muertos[sintomas] == 0).all(axis=1)]
num_todos_0 = len(todos_0)

# Contar cuántos tienen todos los síntomas = 2
todos_2 = muertos[(muertos[sintomas] == 2).all(axis=1)]
num_todos_2 = len(todos_2)

print(f"Muertos los cuales no sabemos ninguno de sus sintomas: {num_todos_0}")
print(f"Muertos asintomaticos: {num_todos_2}")
muertos = 0
for i in range(len(data["FECHA_DEF"])):
    if data["FECHA_DEF"][i] == 1:
        muertos +=1
print(f"Total de muertos: {muertos}")
Pct_muertos = muertos/99526
print(f"Porcentaje  de muertos: {round(Pct_muertos*100,2)}%")
```

```
ENFERMO
1    92054
2     7472
```

```
Name: count, dtype: int64, total de población: 99526
Muertos los cuales no sabemos ninguno de sus sintomas: 0
Muertos asintomaticos: 279
Total de muertos: 3710
Porcentaje  de muertos: 3.73%
```

[37]:
```python
for sintoma in sintomas:
    # Contar cuantas veces coincide el sintoma con que este enfermo el paciente
    sintoma_y_enfermo = len(data[(data[sintoma] == 1) & (data['ENFERMO'] == 1)])
    total_con_sintoma = len(data[data[sintoma] == 1])

    #Calcular cuantas personas enfermas tienen el sintoma
    pct_enfermos_con_sintoma = (sintoma_y_enfermo / total_poblacion) * 100
    # Calcular de las personas con el sintoma cuantos estan enfermos
    pct_sintoma_que_son_enfermos = (sintoma_y_enfermo / total_con_sintoma) * 100
    s

    if sintoma == sintomas[0]:
        print(f"{'Síntoma':15} {'% ' 'Enfermos con síntoma en la población␣
↪total':>25} {' % ' 'Si tiene sintoma esta enfermo ':>30}")
    print(f"{sintoma:15} {pct_enfermos_con_sintoma:25.2f}%␣
↪{pct_sintoma_que_son_enfermos:40.2f}%")
```

```
Síntoma         % Enfermos con síntoma en la población total  % Si tiene sintoma
esta enfermo
AM                                19.69%
91.15%
NEUMONIA                          21.95%
92.90%
DIABETES                          12.12%
91.69%
EPOC                               3.11%
92.46%
ASMA                               3.67%
93.23%
INMUSUPR                           2.84%
89.64%
HIPERTENSION                      14.62%
91.34%
CARDIOVASCULAR                     3.11%
92.25%
OBESIDAD                           6.24%
92.09%
RENAL_CRONICA                      3.17%
91.24%
TABAQUISMO                         4.15%
92.09%
OTRO_CASO                         16.86%
```

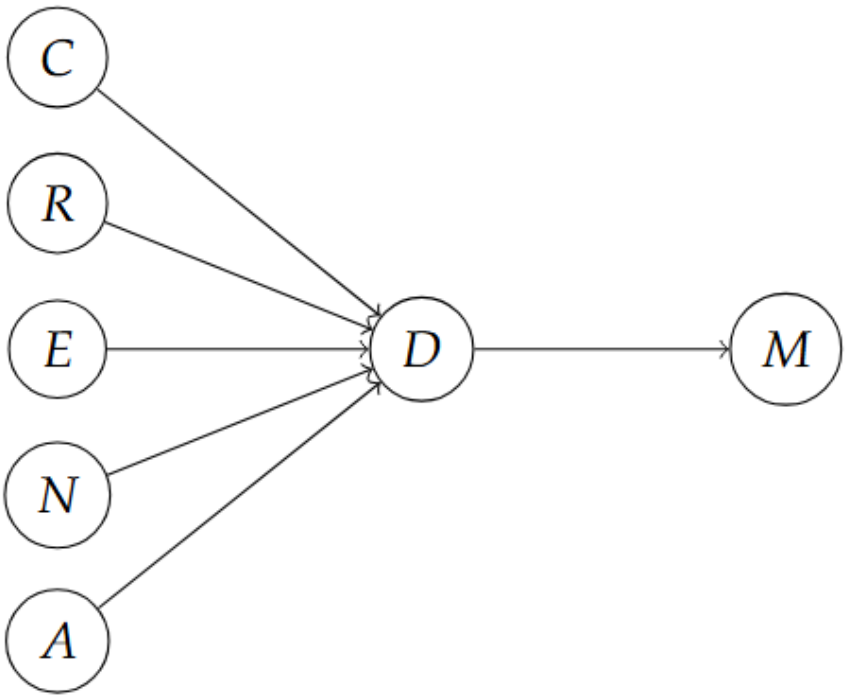
90.45%

[38]:
```
# Calcular y mostrar % de muertos por síntoma (sobre población total y sobre quienes tienen el síntoma)
resultados_top_sintomas_mortiferos = []
for sintoma in sintomas:
    total_con_sintoma = len(data[data[sintoma] == 1])
    sintoma_y_muerto = len(data[(data[sintoma] == 1) & (data['FECHA_DEF'] ==1)])

    if sintoma == sintomas[0]:
        print(f"{'Síntoma':15} {'% Muertos que presentaron el sintoma (sobre población total)':>45} {'% Peronas con sintoma que estan muertos':>55}")

    pct_muertos_con_sintoma = (sintoma_y_muerto / total_poblacion) * 100
    pct_sintoma_que_estanmuertos = (sintoma_y_muerto / total_con_sintoma * 100) if total_con_sintoma > 0 else 0.0

    print(f"{sintoma:15} {pct_muertos_con_sintoma:25.2f}% {pct_sintoma_que_estanmuertos:60.2f}%")

    resultados_top_sintomas_mortiferos.append({
        'sintoma': sintoma,
        'pct_muertos_entre_con_sintoma': pct_sintoma_que_estanmuertos,
        'muertos': sintoma_y_muerto,
        'total_con_sintoma': total_con_sintoma
    })

# Ordenar de mayor a menor y mostrar top 5
top5 = sorted(resultados_top_sintomas_mortiferos, key=lambda x: x['pct_muertos_entre_con_sintoma'], reverse=True)[:5]
print("\nTop 5 síntomas por % de quienes los tienen que murieron (de mayor a menor):")
for i, r in enumerate(top5, start=1):
    print(f"{i}. {r['sintoma']}: {r['pct_muertos_entre_con_sintoma']:.2f}% ({r['muertos']}/{r['total_con_sintoma']})")
```

```
Síntoma         % Muertos que presentaron el sintoma (sobre población total)
% Peronas con sintoma que estan muertos
AM                                    2.16%
10.00%
NEUMONIA                              2.44%
10.32%
DIABETES                              1.25%
9.49%
EPOC                                  0.41%
12.20%
ASMA                                  0.09%
```

```
2.25%
INMUSUPR                        0.26%
8.08%
HIPERTENSION                    1.43%
8.93%
CARDIOVASCULAR                  0.42%
12.49%
OBESIDAD                        0.43%
6.33%
RENAL_CRONICA                   0.43%
12.35%
TABAQUISMO                      0.40%
8.80%
OTRO_CASO                       0.36%
1.94%

Top 5 síntomas por % de quienes los tienen que murieron (de mayor a menor):
1. CARDIOVASCULAR: 12.49% (419/3356)
2. RENAL_CRONICA: 12.35% (427/3458)
3. EPOC: 12.20% (408/3344)
4. NEUMONIA: 10.32% (2426/23519)
5. AM: 10.00% (2149/21496)
```

En base a estos 5 síntomas/Enfermedades se generó la red bayesiana que se muestra a continuación:



En la que cada variable aleatoria (nodo) representa:

- C: Tiene enfermedad cardiovascular
- R: Tiene enfermedad renal crónica
- E: Tiene EPOC
- N: Tiene neumonía
- A: Es adulto mayor ($edad \geq 60$)
- D: Resultado del diagnóstico
- M: Resultado de muerte

Cada una de estas variables aleatorias tiene dos posibles resultados True ó False

Para el cálculo de cada probabilidad de los síntomas se usó la propia base de datos para determinar la probabilidad de que una persona aleatoria en México tenga uno de los 5 síntomas elegidos, esto en base a frecuencias y la siguiente fórmula:

$$P(s_1) = \frac{\#\text{ de personas con síntoma } s_1}{\#\text{ de personas en la base de datos}}$$

[39]:
```
def calculo_sintoma(df,sintoma):
    total_con_sintoma = len(df[df[sintoma] == 1])
    Pct_Sintoma = total_con_sintoma/total_poblacion
    return Pct_Sintoma

# La probabilidad de tener cada sintoma
Probabilidad_AM = round(calculo_sintoma(data,"AM"),2)
Probabilidad_Neumonia = round(calculo_sintoma(data,"NEUMONIA"),2)
Probabilidad_EPOC = round(calculo_sintoma(data,"EPOC"),2)
Probabilidad_Cardiovascular = round(calculo_sintoma(data,"CARDIOVASCULAR"),2)
Probabilidad_Renal_Cronica = round(calculo_sintoma(data,"RENAL_CRONICA"),2)

Ndata = {
    'Sintoma': ["AM", "NEUMONIA", "EPOC", "CARDIOVASCULAR", "RENAL_CRONICA"],
    'Probabilidad_Enfermedad': [Probabilidad_AM, Probabilidad_Neumonia,␣
 ↪Probabilidad_EPOC, Probabilidad_Cardiovascular, Probabilidad_Renal_Cronica],
    'Pct_de_enfermos_con_sintoma': [0.2129, 0.2373, 0.336, 0.336, 0.343],␣
 ↪#Porcentajes agregados manualmente de los codigos anteriores
    'Pct_de_persomas_con_sintoma_Enfermos': [0.9115, 0.929, 0.9246, 0.9225, 0.
 ↪9124] #Porcentajes agregados manualmente de los codigos anteriores
}



dfNdata = pd.DataFrame(Ndata)
dfNdata
```

[39]:

	Sintoma	Probabilidad_Enfermedad	Pct_de_enfermos_con_sintoma	\
0	AM	0.22	0.2129	
1	NEUMONIA	0.24	0.2373	
2	EPOC	0.03	0.3360	

```
3  CARDIOVASCULAR                    0.03                          0.3360
4   RENAL_CRONICA                    0.03                          0.3430

   Pct_de_persomas_con_sintoma_Enfermos
0                                0.9115
1                                0.9290
2                                0.9246
3                                0.9225
4                                0.9124
```

4.0.1 Cálculo de la Tabla de Probabilidad Condicional (CPT) a partir de Datos

Para construir la Tabla de Probabilidad Condicional (CPT) de nuestro nodo `Diagnóstico`, hemos calculado cada probabilidad condicional directamente a partir de la frecuencia de los casos en nuestro conjunto de datos.

El principio es sencillo: la probabilidad condicional de tener la enfermedad, dada una combinación específica de síntomas, se estima como la **proporción** de personas con esa combinación de síntomas que efectivamente tuvieron un diagnóstico positivo.

La fórmula general que aplicamos es:

Sea $e_i = \{s_1, s_2, s_3, s_4, s_5\}$ una combinación de síntomas dada.

$$P(\text{D=True} \mid e_i) = \frac{\text{Nº de casos con D=True y síntomas } e_i}{\text{Nº de casos con síntomas } e_i}$$

Proceso de Cálculo Este cálculo se repite para cada una de las $2^5 = 32$ combinaciones posibles de los cinco síntomas (donde cada síntoma puede estar presente `True` o ausente `False`).

Ejemplo Concreto:

Para determinar la probabilidad de diagnóstico si un paciente presenta el síntoma 1 ($S_1 = T$) pero no presenta el síntoma 2 ($S_2 = F$), la fórmula específica sería:

$$P(D = T \mid S_1 = T, S_2 = F, \dots) = \frac{\text{Nº de personas con } (D = T, S_1 = T, S_2 = F, \dots)}{\text{Nº total de personas con } (S_1 = T, S_2 = F, \dots)}$$

Al aplicar este procedimiento para todas las combinaciones posibles, construimos la CPT completa que define la relación entre los síntomas y la probabilidad del diagnóstico en nuestra red bayesiana.

5 Caclular P(S)

[40]:
```python
def p_s(AM: bool, NEUMONIA: bool , EPOC: bool, CARDIOVASCULAR: bool,␣
 ↪RENAL_CRONICA: bool, dfData):
    if AM == True:
        Valor_am = dfData["Probabilidad_Enfermedad"][0]
    else:
```

```
        Valor_am = 1 - dfData["Probabilidad_Enfermedad"][0]
    if NEUMONIA == True:
        Valor_NEUMONIA = dfData["Probabilidad_Enfermedad"][1]
    else:
        Valor_NEUMONIA = 1 - dfData["Probabilidad_Enfermedad"][1]
    if EPOC == True:
        Valor_EPOC= dfData["Probabilidad_Enfermedad"][2]
    else:
        Valor_EPOC = 1 - dfData["Probabilidad_Enfermedad"][2]
    if CARDIOVASCULAR == True:
        Valor_CARDIOVASCULAR = dfData["Probabilidad_Enfermedad"][3]
    else:
        Valor_CARDIOVASCULAR = 1 - dfData["Probabilidad_Enfermedad"][3]
    if RENAL_CRONICA == True:
        Valor_RENAL_CRONICA = dfData["Probabilidad_Enfermedad"][4]
    else:
        Valor_RENAL_CRONICA = 1 - dfData["Probabilidad_Enfermedad"][4]
    p_s =␣
 ↪Valor_am*Valor_NEUMONIA*Valor_EPOC*Valor_CARDIOVASCULAR*Valor_RENAL_CRONICA
    return p_s

Probabilidad_S = p_s(True,True,True,True,True, dfNdata)
#print(f"P(D) = {Probabilidad_S}")
combinaciones = list(itertools.product([False, True], repeat=5))

# Crear tabla con resultados
tabla = []
for comb in combinaciones:
    prob = p_s(*comb, dfNdata)
    fila = dict(zip(dfNdata["Sintoma"], comb))
    fila["Probabilidad_S"] = prob
    tabla.append(fila)

df_resultados = pd.DataFrame(tabla)
df_resultados
```

[40]:

	AM	NEUMONIA	EPOC	CARDIOVASCULAR	RENAL_CRONICA	Probabilidad_S
0	False	False	False	False	False	0.541033
1	False	False	False	False	True	0.016733
2	False	False	False	True	False	0.016733
3	False	False	False	True	True	0.000518
4	False	False	True	False	False	0.016733
5	False	False	True	False	True	0.000518
6	False	False	True	True	False	0.000518
7	False	False	True	True	True	0.000016
8	False	True	False	False	False	0.170852
9	False	True	False	False	True	0.005284

10	False	True	False	True	False	0.005284
11	False	True	False	True	True	0.000163
12	False	True	True	False	False	0.005284
13	False	True	True	False	True	0.000163
14	False	True	True	True	False	0.000163
15	False	True	True	True	True	0.000005
16	True	False	False	False	False	0.152599
17	True	False	False	False	True	0.004720
18	True	False	False	True	False	0.004720
19	True	False	False	True	True	0.000146
20	True	False	True	False	False	0.004720
21	True	False	True	False	True	0.000146
22	True	False	True	True	False	0.000146
23	True	False	True	True	True	0.000005
24	True	True	False	False	False	0.048189
25	True	True	False	False	True	0.001490
26	True	True	False	True	False	0.001490
27	True	True	False	True	True	0.000046
28	True	True	True	False	False	0.001490
29	True	True	True	False	True	0.000046
30	True	True	True	True	False	0.000046
31	True	True	True	True	True	0.000001

6 Calculos de P(¬S|D) y P(S|D)

[41]:
```
resultados = []

# Primero calculamos el total de muertes para los porcentajes
total_enfermos = len(data[data['ENFERMO'] == 1])

for comb in itertools.product([0, 1], repeat=len(dfNdata["Sintoma"])):
    # Construir la query string
    condiciones_query = []
    for i, col in enumerate(dfNdata["Sintoma"]):
        if comb[i] == 1:
            condiciones_query.append(f"{col} == 1")
        else:
            condiciones_query.append(f"{col} != 1")

    query_str = " & ".join(condiciones_query) + " & ENFERMO == 1"

    # Calcular el conteo
    conteo = len(data.query(query_str))

    # Calcular porcentaje
    porcentaje = (conteo / total_enfermos * 100) if total_enfermos > 0 else 0
```

```
    # Crear diccionario con resultados
    resultado = {col: comb[i] for i, col in enumerate(dfNdata["Sintoma"])}
    resultado['ENFERMOS'] = conteo
    resultado['PORCENTAJE'] = round(porcentaje, 2)
    resultados.append(resultado)

# Crear DataFrame final
tabla_resultados = pd.DataFrame(resultados)

print(tabla_resultados)
print(f"\nTotal de enfermos: {total_enfermos}")
```

	AM	NEUMONIA	EPOC	CARDIOVASCULAR	RENAL_CRONICA	ENFERMOS	PORCENTAJE
0	0	0	0	0	0	56475	61.35
1	0	0	0	0	1	763	0.83
2	0	0	0	1	0	477	0.52
3	0	0	0	1	1	74	0.08
4	0	0	1	0	0	249	0.27
5	0	0	1	0	1	16	0.02
6	0	0	1	1	0	21	0.02
7	0	0	1	1	1	15	0.02
8	0	1	0	0	0	13107	14.24
9	0	1	0	0	1	508	0.55
10	0	1	0	1	0	414	0.45
11	0	1	0	1	1	69	0.07
12	0	1	1	0	0	205	0.22
13	0	1	1	0	1	18	0.02
14	0	1	1	1	0	46	0.05
15	0	1	1	1	1	3	0.00
16	1	0	0	0	0	9737	10.58
17	1	0	0	0	1	552	0.60
18	1	0	0	1	0	522	0.57
19	1	0	0	1	1	136	0.15
20	1	0	1	0	0	919	1.00
21	1	0	1	0	1	70	0.08
22	1	0	1	1	0	141	0.15
23	1	0	1	1	1	38	0.04
24	1	1	0	0	0	4746	5.16
25	1	1	0	0	1	597	0.65
26	1	1	0	1	0	629	0.68
27	1	1	0	1	1	156	0.17
28	1	1	1	0	0	914	0.99
29	1	1	1	0	1	82	0.09
30	1	1	1	1	0	297	0.32
31	1	1	1	1	1	58	0.06

```
Total de enfermos: 92054
```

El primero que no incluye P(¬S|D) siendo 61.35% y P(S(i)|D) es el resto

7 Calculo de P(D | Sintomas) y P(D | ¬Sintomas)

Para eso obtenemos para cada P(S(i)) buscamos cuantos enfermos tiene y lo dividimos entre su población total

[42]:
```
resultados = []

total_poblacion = len(data)
total_enfermos = len(data[data['ENFERMO'] == 1])

for comb in itertools.product([0, 1], repeat=len(dfNdata["Sintoma"])):
    # Construir la query base (síntomas)
    condiciones_query = []
    for i, col in enumerate(dfNdata["Sintoma"]):
        if comb[i] == 1:
            condiciones_query.append(f"{col} == 1")
        else:
            condiciones_query.append(f"{col} != 1")

    query_str = " & ".join(condiciones_query)

    # Conteos
    conteo_poblacion = len(data.query(query_str))
    conteo_enfermos = len(data.query(query_str + " & ENFERMO == 1"))

    # Probabilidad condicional ENFERMO|S
    if conteo_poblacion > 0:
        prob_enfermo_dado_s = conteo_enfermos / conteo_poblacion
    else:
        prob_enfermo_dado_s = 0

    # Crear diccionario con resultados
    resultado = {col: comb[i] for i, col in enumerate(dfNdata["Sintoma"])}
    resultado['P(D|S)'] = round(prob_enfermo_dado_s, 4)  # redondeado a 4␣
 ↪decimales
    resultados.append(resultado)

# Crear DataFrame final
tabla_resultados = pd.DataFrame(resultados)

print(tabla_resultados)
```

```
    AM  NEUMONIA  EPOC  CARDIOVASCULAR  RENAL_CRONICA  P(D|S)
0    0         0     0               0              0  0.9277
1    0         0     0               0              1  0.8987
2    0         0     0               1              0  0.9138
```

3	0	0	0	1	1	0.8810
4	0	0	1	0	0	0.9188
5	0	0	1	0	1	0.8000
6	0	0	1	1	0	0.8750
7	0	0	1	1	1	0.9375
8	0	1	0	0	0	0.9362
9	0	1	0	0	1	0.8975
10	0	1	0	1	0	0.9495
11	0	1	0	1	1	0.8734
12	0	1	1	0	0	0.9447
13	0	1	1	0	1	0.9474
14	0	1	1	1	0	0.9787
15	0	1	1	1	1	1.0000
16	1	0	0	0	0	0.9046
17	1	0	0	0	1	0.9246
18	1	0	0	1	0	0.9094
19	1	0	0	1	1	0.9577
20	1	0	1	0	0	0.9236
21	1	0	1	0	1	0.8861
22	1	0	1	1	0	0.9156
23	1	0	1	1	1	0.9744
24	1	1	0	0	0	0.9146
25	1	1	0	0	1	0.9270
26	1	1	0	1	0	0.9129
27	1	1	0	1	1	0.9176
28	1	1	1	0	0	0.9204
29	1	1	1	0	1	0.9111
30	1	1	1	1	0	0.9399
31	1	1	1	1	1	0.9508

P(D|¬S) = 0.9277 algo alta pero, los 5 sintomas solo representan el 39% de la población y al ser basados parea mortandad, pues los niveles de enfremedades de los otros sintomas son altos, pero no tienen un impacto tan alto en las muertes por malos diagnosticos, además el calculo se basa en el dataset que tenemos, el cual se puede ver sesgado ya que solo tenemos personas que van al medico y que seguramente presentan algun tipo de molestia, pero en en este caso creemos que si sirve porque ya que una persona que no presenta molestias no se presentaria en el medico, el resto son es P(D|S(i))

8 Ahora calcularemos P(D)

Dados **5 síntomas** $(S_1, S_2, S_3, S_4, S_5)$ y un estado $e = \{s_1, s_2, s_3, s_4, s_5\}$, la expresión queda como:

$$P(D) = \sum_{e_1=0}^{1} \sum_{e_2=0}^{1} \sum_{e_3=0}^{1} \sum_{e_4=0}^{1} \sum_{e_5=0}^{1} P(D \mid s_1 = e_1, s_2 = e_2, s_3 = e_3, s_4 = e_4, s_5 = e_5) \prod_{i=1}^{5} P(s_i = e_i)$$

Todas las combinaciones posibles de síntomas (32 combinaciones), se multiplica la probabilidad condicional $P(D \mid S)$ por las probabilidades de cada síntoma, y luego se suman.

[43]:
```
def p_d(pDdadoS,pS):
    Valor_pd = 0
    for i in range(len(pS)):
        Valor_pd += pDdadoS[i]*pS[i]
    return Valor_pd

P_d = round((p_d(tabla_resultados['P(D|S)'],␣
 ↪df_resultados["Probabilidad_S"])),4)
P_Nod = round(1 - (p_d(tabla_resultados['P(D|S)'],␣
 ↪df_resultados["Probabilidad_S"])),4)

print(f"P(D) = {P_d} y P(¬D): {P_Nod}")
```

```
P(D) = 0.9239 y P(¬D): 0.0761
```

Por ende P(D) = 92.06% y y P(¬D) = 7.94%

8.0.1 Calculo P(D|M) y P(¬D|M)

[44]:
```
# Casos con D y M
casos_DM = len(data[(data["ENFERMO"] == 1) & (data["FECHA_DEF"] == 1)])

# Calculo de probabilidades
P_D_dado_M = casos_DM / muertos
P_notD_dado_M = 1 - P_D_dado_M

print(f"P(D|M) = {P_D_dado_M}")
print(f"P(¬D|M) = {P_notD_dado_M}")
```

```
P(D|M) = 0.8981132075471698
P(¬D|M) = 0.10188679245283017
```

Falsos negativos 10.18%

9 Calculo P(M | ¬D)

[45]:
```
P_M_dado_notD = (P_notD_dado_M*Pct_muertos) / P_Nod
print(f"P(M|¬D) = {P_M_dado_notD}")
```

```
P(M|¬D) = 0.04990804904075804
```

Calculo de error de diagnostico, muertos con mal diagnostico 4.99%

10 Calculo P(D ¬M) falsos positivos (no muertos)

[46]:
```
P_D_dado_notM = (P_d-(P_D_dado_M*Pct_muertos))/(1-Pct_muertos)
print(f"P(D|¬M) = {P_D_dado_notM}")
```

```
P(D|¬M) = 0.9248984658094682
```

Este cálculo nos da la cantidad de personas que recibieron un diagnóstico y no murieron

10.1 Referencias

1. Russell, S., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed., pp. 510–519). Pearson.
2. Huang, B. (2015, mayo 25). *Bayesian Networks* [Video]. YouTube. https://www.youtube.com/watch?v=TuGDMj43ehw
3. Secretaría de Salud. (2025). Datos Abiertos Dirección General de Epidemiología. Dirección General de Epidemiología. https://www.gob.mx/salud/documentos/datos-abiertos-152127