A logo of a university

Description automatically generated

**كلية العلوم**

**قــســــــــــم الانـــظــــمــــة الــــطـبـيـة الـــذكــــــيـــة**

**Intelligent Medical Systems Department**

**محاضرة نظري عملي لحساب قيمة الاحتمالية والاحتمالية الشرطية وكذلك اضافة الاحتماليات وتقاطعها**

**Class: Second**

**Lecturer:** **Prof. Dr. Mehdi Ebady Manaa**

In medical systems, probability is a powerful tool for handling uncertainty—particularly in areas such as test results, disease detection, and patient response to treatments. This lecture explores how probability helps answer questions like:  
- What is the chance a patient tests positive?  
- What are all possible outcomes for a group of patients?  
- How do we calculate the probability of specific sequences of test results?  
  
We will consider a scenario in which we analyze the outcome of three independent diagnostic tests. Each test can return either a Positive (P) or Negative (N) result.

# 1. Sample Space (S)

Each of the three tests can result in either a Positive (P) or Negative (N), so the total number of outcomes is 2^3 = 8. Assuming independence between tests, the full sample space is:  
  
S = {  
 (P, P, P), (P, P, N), (P, N, P), (P, N, N),  
 (N, P, P), (N, P, N), (N, N, P), (N, N, N)  
}

# 2. Probability of Each Outcome

Assume:  
- P(P) = 0.7 (Probability of a Positive result)  
- P(N) = 0.3 (Probability of a Negative result)  
  
The probabilities of each sequence can be calculated as the product of the individual probabilities:  
- P(P, P, P) = 0.7 \* 0.7 \* 0.7 = 0.343  
- P(P, P, N) = 0.7 \* 0.7 \* 0.3 = 0.147  
- P(P, N, P) = 0.7 \* 0.3 \* 0.7 = 0.147  
- P(P, N, N) = 0.7 \* 0.3 \* 0.3 = 0.063  
- P(N, P, P) = 0.3 \* 0.7 \* 0.7 = 0.147  
- P(N, P, N) = 0.3 \* 0.7 \* 0.3 = 0.063  
- P(N, N, P) = 0.3 \* 0.3 \* 0.7 = 0.063  
- P(N, N, N) = 0.3 \* 0.3 \* 0.3 = 0.027

# 3. Addition Rule of Probability

To find the probability that a patient has either exactly one or exactly two positive results:  
  
Define:  
- Event A: Exactly 2 Positives = (P, P, N), (P, N, P), (N, P, P)  
- Event B: Exactly 1 Positive = (P, N, N), (N, P, N), (N, N, P)  
  
P(A) = 0.147 + 0.147 + 0.147 = 0.441  
P(B) = 0.063 + 0.063 + 0.063 = 0.189  
Since A and B are mutually exclusive:  
P(A ∪ B) = P(A) + P(B) = 0.441 + 0.189 = 0.63

# 4. Intersection of Events

Let:  
- Event C: First test is positive  
- Event D: Last test is positive  
  
Matching outcomes:  
- C: (P, P, P), (P, P, N), (P, N, P), (P, N, N) → P(C) = 0.7  
- D: (P, P, P), (P, N, P), (N, P, P), (N, N, P) → P(D) = 0.7  
- C ∩ D: (P, P, P), (P, N, P) → P(C ∩ D) = 0.343 + 0.147 = 0.49

# 5. Python Code for Probability Calculations

from itertools import product

# Define base probabilities

P\_P = 0.7 # Probability of Positive

P\_N = 0.3 # Probability of Negative

# Generate all possible test outcomes (3 outcomes, each P or N)

outcomes = list(product(['P', 'N'], repeat=3))

# Calculate probability of each outcome

def get\_prob(outcome):

prob = 1

for test in outcome:

prob \*= P\_P if test == 'P' else P\_N

return prob

# Define events

# Event A: Exactly two positives

event\_A = [o for o in outcomes if o.count('P') == 2]

# Event B: Exactly one positive

event\_B = [o for o in outcomes if o.count('P') == 1]

# Event C: First test is Positive

event\_C = [o for o in outcomes if o[0] == 'P']

# Event D: Last test is Positive

event\_D = [o for o in outcomes if o[2] == 'P']

# Event C ∩ D: First and last are Positive

event\_C\_and\_D = [o for o in outcomes if o[0] == 'P' and o[2] == 'P']

# Addition Rule: P(A ∪ B)

P\_A = sum(get\_prob(o) for o in event\_A)

P\_B = sum(get\_prob(o) for o in event\_B)

P\_A\_or\_B = P\_A + P\_B # Mutually exclusive

# Intersection Rule by listing

P\_C\_and\_D = sum(get\_prob(o) for o in event\_C\_and\_D)

# Multiplication Rule (if C and D were independent)

P\_C = sum(get\_prob(o) for o in event\_C)

P\_D = sum(get\_prob(o) for o in event\_D)

P\_C\_times\_D = P\_C \* P\_D # Only valid if C and D are independent

# Print results

print("P(A):", round(P\_A, 3))

print("P(B):", round(P\_B, 3))

print("P(A ∪ B) [Addition]:", round(P\_A\_or\_B, 3))

print("P(C):", round(P\_C, 3))

print("P(D):", round(P\_D, 3))

print("P(C ∩ D) [By listing]:", round(P\_C\_and\_D, 3))

print("P(C ∩ D) [Multiplication]:", round(P\_C\_times\_D, 3))