Tarea 2

En esta tarea pondremos en práctica los conceptos de probabilidad vistos en clase. En la primera parte, resolveremos ejercicios a mano. En la segunda parte, resolveremos ejercicios en Python. En cada parte doy una breve introducción a los conceptos necesarios para resolver los ejercicios.

Parte 1

En esta parte de la tarea, resolveremos ejercicios a mano.

Sean X, Y variables aleatorias continuas con función de densidad

$$f_{xy}(x, y) = \frac{1}{2}x + \frac{3}{2}y \quad \text{para } 0 \leq x \leq 1, 0 \leq y \leq 1$$

La función de distribución acumulada, o cdf, está dada por

$$F(x, y) = \int_0^x \int_0^y \left(\frac{1}{2}x + \frac{3}{2}y\right) \mathrm{d}y\mathrm{d}x$$

La función de densidad condicional está dada por

$$f(x|y) = \frac{f(x, y)}{f(y)}$$

1. Obtener $F(x, y)$. *Tip:* integrar primero respecto de una variable (e.g., con respecto a y) dejando la otra constante. Luego integrar con respecto a la otra.
2. Obtener las funciones marginales $f(x)$ y $f(y)$.
3. Encontrar la condicional $f(x|y)$.

Parte 2

En esta parte de la tarea, resolveremos ejercicios en Python.

A

La pmf de una distribución de probabilidad discreta de Poisson permite calcular la probabilidad un número dado de eventos en un intervalo, dada una tasa de ocurrencia λ.

$$p(x \mid \lambda) = \frac{e^{-\lambda}\lambda^x}{x!} \tag{11}$$

¿Cómo obtener las siguientes probabilidades?

4. $p(x = 3 \mid \lambda = 3/4) = ?$
5. $P(0 \leq x \leq 10 \mid \lambda = 3) = ?$ *Tip:* notar que es una probabilidad acumulada, con $x = 0$ hasta $x = 10$.

Nota: puedes crear tu propia función en Python para obtenerlas. Recuerda que la función factorial está disponible en la librería `math`, y la función exponencial en la librería `numpy`.

B

La integración numérica es conjunto de algoritmos que aproximan el área bajo la curva de una función en una o múltiples dimensiones. Uno de esos algoritmos, la regla del trapecio, el área bajo la curva de $f(x)$ en el intervalo $[a, b]$ es la suma de diversas subáreas formadas por n subintervalos. En la regla de Simpson, el área total es las áreas de subintervalos en donde cada subintervalo es aproximado con una función cuadrática. La exactitud de la aproximación depende de la cantidad de subintervalos n:

En Python se puede realizar integración numérica con la librería `scipy`. Por ejemplo, para resolver

$$f(x) = \int_0^2 x^2 \mathrm{d}x$$

Podemos usar la función `quad` de la siguiente manera

```
from scipy.integrate import quad

def f(x):
  return x**2

# f es la función a integrar
# a es el límite inferior de integración
# b es el límite superior de integración
```

```
quad(f, a=0, b=2)
# (2.666666666666667, 2.960594732333751e-14)
```

Que retorna una tupla con el valor de la integral (2.66...) y el error de la aproximación ($2.96e^{-14}$). Este resultado se puede verificar analíticamente: x^2 es $x^3/3$ y evaluando en $x = 2$ y $x = 0$ se obtiene $8/3$.

Para una función en dos dimensiones, la función `nquad` permite integrar en dos o más dimensiones. Por ejemplo, para resolver

$$f(x, y) = \int_0^1 \int_0^1 \left(\frac{1}{2}x + \frac{3}{2}y\right) \mathrm{d}x\mathrm{d}y$$

Podemos usar la función `nquad` de la siguiente manera

```
from scipy.integrate import nquad

def f(x, y):
  return 0.5*x + 1.5*y

nquad(f, [[0, 1], [0, 1]])
# (1.0, 1.9392741548933183e-14)
```

O si se quiere integrar con límite superior en x en ∞

$$f(x, y) = \int_0^\infty \int_0^1 \left(\frac{1}{2}x + \frac{3}{2}y\right) \mathrm{d}x\mathrm{d}y$$

```
ax = 0
bx = np.inf
ay = 0
by = 1
nquad(f, [[ax, bx], [ay, by]])
```

Un resultado similar debería obtenerse con `dblquad`.

La integración simbólica se distingue de la numérica en que la simbólica encuentra la solución analítica de la integral, mientras que la numérica encuentra una aproximación. La integración simbólica es posible con la librería `sympy`. Por ejemplo, para resolver:

$$f(x) = \int_0^2 x^2 \mathrm{d}x$$

Podemos usar la función `integrate` de la siguiente manera

```
from sympy import integrate, symbols

x = symbols('x')
f = x**2

integrate(f, (x, 0, 2))
# 8/3
```

Dada la función de densidad de probabilidad conjunta

$$f(x,y) = \int_0^1 \int_0^1 \left(\frac{1}{2}x + \frac{3}{2}y\right) \mathrm{d}x\mathrm{d}y$$

Si queremos obtener la marginal de x con `sympy`, podemos usar la función `integrate` de la siguiente manera

```
x, y = symbols('x y')
f = 0.5*x + 1.5*y
# marginal de x; notar que integramos con respecto a y, y el límite inferior y superior de
fx = integrate(f, (y, 0, 1))
fx
# x/2 + 3/4
```

Si quisiéramos obtener la probabilidad de que x se encuentre en el intervalo entre 0 y 0.7 $(0 < x < 0.7)$, integramos la marginal de x en ese intervalo:

```
# probabilidad de que x>2
integrate(fx, (x, 0, 0.7))
# 0.64
```

Considerando lo anterior, resuelve los siguientes ejercicios:

Un sistema electrónico tiene uno de cada dos tipos diferentes de componentes de operación en operación conjunta. Denote con X y Y las duraciones aleatorias de los componentes del tipo I y tipo II respectivamente. La función de densidad de probabilidad conjunta está dada por

$$f_{xy}(x,y) = \frac{1}{8}x \exp\left[-(x+y)/2\right], \quad \text{para } x, y > 0$$

Nota: las mediciones son en cientos de horas.

6. Encontrar $P(X > 1, Y > 1)$. *Tip:* el límite inferior de integración es 1 para ambas variables.
7. Encontrar la probabilidad de que el componente tipo II tenga una vida útil de más de 200 horas (ver la Nota). Hacerlo de dos maneras:
    - 7.1. Usando integración numérica con `nquad` o `dblquad`.
    - 7.2. Usando la función de densidad marginal $f(y)$ que puedes obtener con `Sympy`. *Tip:* el infinito en sympy se escribe como `oo`, hay que importarlo con `from sympy import oo`.

```
from sympy import integrate, symbols, exp, oo
# ... resto del código
```

En funciones de densidad predefinidas (como la normal, uniforme, exponencial, etc.) se puede usar la función `pdf` de `scipy.stats`. Por ejemplo, para obtener la probabilidad de que una variable aleatoria normal con media μ y desviación estándar σ esté en el intervalo $[a, b]$, se puede usar la función `norm.pdf` de la siguiente manera

```
from scipy.stats import norm
from scipy.integrate import quad

mu = 0
sigma = 1
a = -1
b = 1

# probabilidad de que la variable aleatoria esté en el intervalo [a, b]
prob = quad(norm.pdf, a, b, args=(mu, sigma))
#  (0.682689492137086, 7.579375928402476e-15)
```

Como vimos en clase, la cdf (función de distribución acumulada) de una variable aleatoria X se define como el área bajo la curva de la función de densidad de probabilidad $f(x)$ hasta un valor x:

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(x)\mathrm{d}x$$

Por lo que podemos usar la función `cdf` de `scipy.stats` para obtener la probabilidad de que una variable aleatoria esté en el intervalo $[a, b]$ de la siguiente manera

```
from scipy.stats import norm

mu = 0
```

```
sigma = 1
a = -1
b = 1

# probabilidad de que la variable aleatoria esté en el intervalo [a, b]
prob = norm.cdf(b, mu, sigma) - norm.cdf(a, mu, sigma)
# 0.6826894921370859
```

El área que estaríamos encontrando es

Resolver:

8. La pdf de la distribución normal con media μ y desviación estándar σ (o varianza σ^2) es caracterizada por

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

- 8.1. Suponer que un manufacturador de un tipo de botana sabe que el peso total del paquete de botana está distribuido normalmente con una media de 80.2 gramos y una desviación estándar de 1.1 gramos. ¿Cuál es la probabilidad de que un paquete de botana pese *menos* de 78 gramos? *Tip:* puedes hacer tu propia función de python, o usar la función `norm.pdf` de `scipy.stats` e integrarla con `quad`; o usar la función `norm.cdf` de `scipy.stats`.
- 8.2. Bajo las mismas asunciones anteriores, ¿cuál es la probabilidad de que un paquete dado tenga un peso que esté entre 2 desviaciones estándar de la media? Es decir, $P(x \in [\mu - \sigma, \mu + \sigma])$.