# PRACTICO N°1

## Sis-420 Com-300

Velasquez Guerra Itzel Emily

Importamos la librería **numpy** para trabajar con vectores, matrices, generar numeros aleatorio, etc. Y **pandas** para manipular y analizar datos

01

# Section

```
[ ]  import numpy as np
     import pandas as pd
```

```
[ ]  # Generador de  estaturas
     #np.random.seed(55)  # Fijar una semilla
     ValEstaturas = np.random.uniform(1.4, 2.2, 100)  #(min,max, cantidad)
     ValPesos = []  #Almacen de pesos
```

```
[ ]  # Generador controlado de pesos
     for estatura in ValEstaturas:
         # Calcular  peso mínimo y máximo, IMC saludable (18.5 a 24.9)
         peso_min = 18.5 * (estatura ** 2)
         peso_max = 24.9 * (estatura ** 2)

         # Generar un peso aleatorio entre el peso mínimo y máximo calculado
         peso = np.random.uniform(peso_min, peso_max)
         ValPesos.append(peso)  # Añadir el peso a la lista de pesos
```

Un generador con random de la variable independiente “estatura”, asignar valores min, max y la cantidad.
Y declarar otra variable de tipo vector para guardar las variables dependientes “pesos”

Con el ciclo for, haremos un generador de pesos, donde como base se utiliza el IMC saludable, para obtener valores mínimos y máximos que debe tener la persona de acuerdo a su estatura, posteriormente generar de manera aleatoria los pesos teniendo limitaciones. Y guardar estos datos en nuestro dataset generado

02

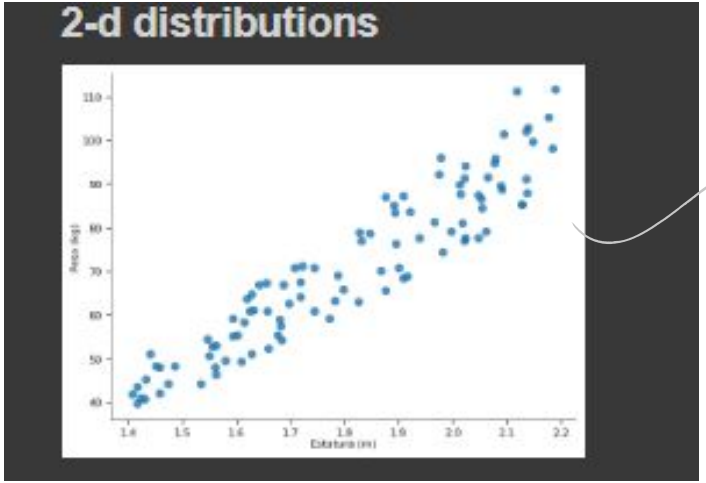
# Section

Todos los datos dependientes e independientes generados aleatoriamente, los guardamos en nuestra data, con dos columnas

```
[35] # Crear un DataFrame con los datos de estatura y peso
     data = pd.DataFrame({
         'Estatura (m)': ValEstaturas,
         'Peso (kg)': ValPesos
     })
```

```
[36] data
     #Var independiente sera estatura
     #Var dependiente peso
```

| | Estatura (m) | Peso (kg) |
|---|---|---|
| 0 | 1.474487 | 44.143554 |
| 1 | 2.177325 | 105.193841 |
| 2 | 1.787088 | 69.118956 |
| 3 | 1.594018 | 59.284394 |
| 4 | 1.824899 | 63.062981 |
| ... | ... | ... |
| 95 | 2.138826 | 102.850322 |
| 96 | 1.771881 | 59.239358 |
| 97 | 2.088309 | 89.721300 |




Con las opciones de google colab, podemos previsualizar cómo será nuestra gráfica de puntos, de acuerdo a los datos generados

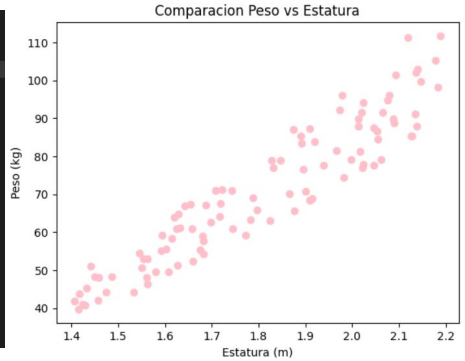
Para revisar los datos generados, tanto en coherencia y cantidad de datos pedidos, imprimos el data con el nombre en que guardamos la información.

03

# Section

Importamos la librería **matplotlib** que funciona como MATLAB, y en nuestro código nos ayudará a graficar, generar etiquetas “títulos”, etc. y mostrar la relación entre la estatura y el peso.

```
0s  import matplotlib.pyplot as mplt

0s  [38] # Ver datos generados de PUNTOS
         mplt.xlabel('Estatura (m)')  # Eje X
         mplt.ylabel('Peso (kg)')  # Eje Y PREDECIR

         mplt.scatter(data['Estatura (m)'], data['Peso (kg)'], color='pink')
         mplt.title('Comparacion Peso vs Estatura')


         mplt.show()  # Mostrar el gráfico
```




La gráfica generada por codigo, se asemeja a la gráfica que ya nos mostro google colab

En la primera línea establecemos la etiqueta del eje X, "Estatura (m)", en la segunda establecemos la etiqueta del eje Y, "Peso (kg)", y en las demás etiquetas asignamos color, títulos y etiquetas que ayudará a entender la gráfica

04

# Section

Insertamos la librería **sklearn** que tiene funciones para realizar regresión lineal y otros modelos lineales.

Creamos un objeto llamado *Regresión* que representa un modelo de regresión lineal.

```
[39] from sklearn import linear_model

[40] Regresion = linear_model.LinearRegression()

[41] #Vector por cada valor independiente
     Estaturas = data['Estatura (m)'].values.reshape(-1,1)
     modelo = Regresion.fit(Estaturas, data['Peso (kg)'])

     #saber los valores de formula
     print("Interseccion b", modelo.intercept_)
     print("Pendiente m", modelo.coef_)

     Interseccion b -63.881068054727024
     Pendiente m [74.57559529]
```

Estas líneas de código son las encargadas de ajustar el modelo de regresión lineal y obtener los parámetros del modelo.

Primera línea extraemos los valores de la columna "Estatura (m)" del DataFrame Y *.reshape(-1, 1)* transforma este array en una matriz de una sola columna.

Segunda línea: *Regresion.fit()* es el método que ajusta el modelo de regresión lineal a tus datos. La variable *Estaturas* es la matriz de características (variable independiente).. Y al final imprimimos los datos.

Importamos la librería **mean_squared_error** para ayudar a esta función se utiliza para calcular el MSE (error cuadratico medio).

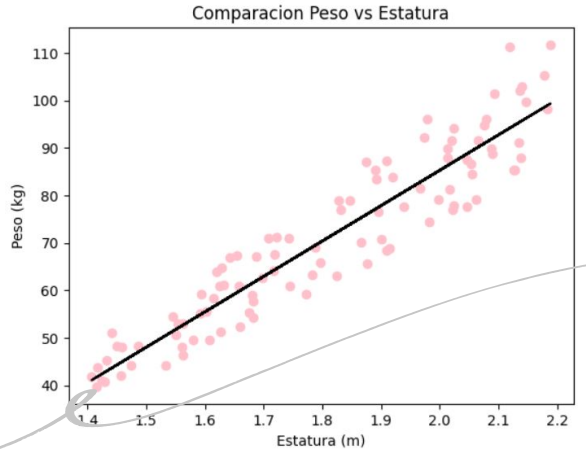
05

# Section

```
#Encontrar el erro cuadratico
from sklearn.metrics import mean_squared_error

Interseccion = modelo.intercept_
Pendiente = modelo.coef_

y = data['Peso (kg)']
y_pred = Interseccion + (Pendiente * data['Estatura (m)'])
mse = mean_squared_error(y, y_pred)
print(f"Error cuadrático medio (MSE): {mse:.2f}")
```

```
Error cuadrático medio (MSE): 38.91
```

```
[43] # Grafico mas Linea
mplt.xlabel('Estatura (m)')
mplt.ylabel('Peso (kg)')

mplt.scatter(data['Estatura (m)'], data['Peso (kg)'], color='pink')
mplt.title('Comparacion Peso vs Estatura')

#Linea
mplt.plot(Estaturas, modelo.predict(Estaturas), color='black')

mplt.show()
```




En la línea 43 copiamos el código que utilizamos para graficar, pero añadimos la penúltima línea, para que se grafique la línea en base a los cálculos realizados

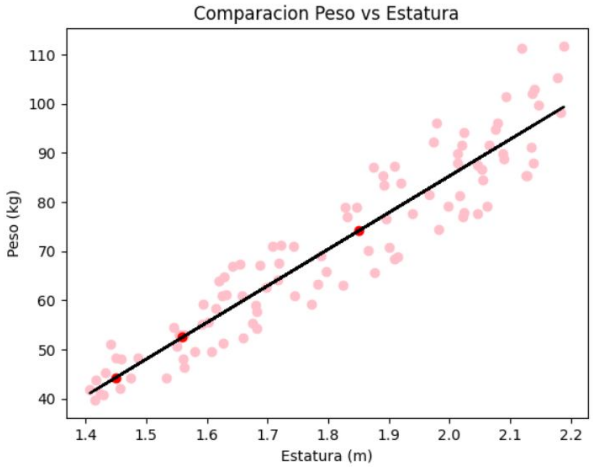
Guardamos el valor de la intersección del modelo en la variable Intersección y Pendiente.
*y = data['Peso (kg)']:* Asigna los valores reales de peso (variable objetivo) a la variable y.
*y_pred = Interseccion + (Pendiente * data['Estatura (m)']):* Calcula las predicciones del modelo para cada valor de estatura. Y se utiliza la ecuación de la recta obtenida del modelo (intersección + pendiente * estatura).

Creamos la variable **entrada** para poder ingresar datos de nuestra variable independiente “estatura” y en base al entrenamiento realizado mediante código, pueda descifrar un valor aproximado o el más adecuado

06

Section

```
[44] #insertar datos y probar
     entrada = [[1.85], [1.45] ,[1.56]]
     modelo.predict(entrada)

     array([74.08378324, 44.25354512, 52.4568606 ])

[45] #Grafica mas datos ingresados
     mplt.xlabel('Estatura (m)')
     mplt.ylabel('Peso (kg)')

     mplt.scatter(data['Estatura (m)'], data['Peso (kg)'], color='pink')
     mplt.title('Comparacion Peso vs Estatura')
     mplt.plot(Estaturas, modelo.predict(Estaturas), color='black')

     #datos ingresados
     mplt.scatter(entrada, modelo.predict(entrada), color='red')


     mplt.show()
```




Para visualizar nuestros resultados finales, volvemos a copiar el codigo para imprimir el grafico, pero ahora incluyendo los datos de entrada que se pusieron a prueba y estos se marque de color rojo para identificarlos mejor.

# Thanks!

Sis-420  Com-300

Velasquez Guerra Itzel Emily - DAD.