In [3]:
```
import numpy as np
```

# Linear Regression Mathematical Intution (Cost Function And Gradient Descent):

In [10]:
```
# Define the height and weight data
X = np.array([152, 157, 163, 157, 160, 168, 173, 175, 178, 183,190])
y = np.array([48, 53, 58, 56, 61, 66, 70, 73, 79, 86,88])


# Initialize parameters
m = 0  # initial guess for slope
b = 0  # initial guess for y-intercept

# Hyperparameters
learning_rate = 0.00006
num_epochs = 1000000

# Number of data points
n = len(X)

# Gradient Descent
for epoch in range(num_epochs):
    # Calculate predictions
    y_pred = m * X + b

    # Calculate the loss (MSE)
    cost = (1 / (2 * n)) * np.sum((y_pred - y) ** 2)

    # Calculate gradients
    gradient_m = (1 / n) * np.sum((y_pred - y) * X)
    gradient_b = (1 / n) * np.sum(y_pred - y)

    # Update parameters using gradients and learning rate
    m -= learning_rate * gradient_m
    b -= learning_rate * gradient_b
```

```
    # Print loss for monitoring
    if epoch % 100000 == 0:
        print(f'Epoch {epoch + 1}, Cost: {cost}, m: {m}, b: {b}')

# Final values of m and b
print(f'Final values: m = {m}, b = {b},cost: {round(cost)}')
```

```
Epoch 1, Cost: 2331.818181818182, m: 0.6879054545454546, b: 0.004025454545454546
Epoch 100001, Cost: 32.210299910541345, m: 0.4197990113169063, b: -3.2123053998074793
Epoch 200001, Cost: 30.58136494242403, m: 0.4382404457498476, b: -6.338426111093785
Epoch 300001, Cost: 29.040902014813945, m: 0.45617408590491026, b: -9.378467498403353
Epoch 400001, Cost: 27.584105962308577, m: 0.4736139141558405, b: -12.334799799708684
Epoch 500001, Cost: 26.206432601556397, m: 0.490573527864607, b: -15.209727987273448
Epoch 600001, Cost: 24.903584556586527, m: 0.507066149982906, b: -18.005493564778465
Epoch 700001, Cost: 23.67149785400631, m: 0.5231046393617279, b: -20.724276314959777
Epoch 800001, Cost: 22.50632924625037, m: 0.5387015007770635, b: -23.368195999127547
Epoch 900001, Cost: 21.40444422333999, m: 0.5538688946795373, b: -25.939314009886267
Final values: m = 0.5686185012276397, b = -28.439610322573525,cost: 20
```

# Logistic Regression Mathematical Intution (Cost Function And Gradient Descent):

In [9]:

```
# Define the features and labels
X = np.array([2.5, 1.5, 3.5, 2.0, 4.0, 3.0, 2.5, 3.0, 2.0, 1.0])
y = np.array([1, 0, 1, 0, 1, 1, 0, 1, 0, 0])

# Initialize parameters
theta = 0  # Initial guess for weights (one weight for each feature)
bias = 0  # Initial guess for bias term

# Hyperparameters
learning_rate = 0.0005
num_iterations = 100000

# Number of data points
m = len(X)

# Gradient Descent
for iteration in range(num_iterations):
    # Calculate predictions (logits)
```

```
    z = X*theta + bias
    y_pred = 1 / (1 + np.exp(-z))  # Using NumPy's exp function directly

    # Calculate the log loss (cross-entropy loss)
    cost = (-1/m) * np.sum(y * np.log(y_pred) + (1 - y) * np.log(1 - y_pred))

    # Calculate gradients
    gradient_theta = (1/m) * np.sum(X*(y_pred - y))
    gradient_bias = (1/m) * np.sum(y_pred - y)

    # Update parameters using gradients and learning rate
    theta -= learning_rate * gradient_theta
    bias -= learning_rate * gradient_bias

    # Print loss for monitoring
    if iteration % 10000 == 0:
        print(f'Iteration {iteration + 1}, Cost: {cost},Theta = {theta}, Bias = {bias}')

# Final values of theta and bias
print(f'Final values: Theta = {theta}, Bias = {bias}, cost: {round(cost, 4)}')
```

```
Iteration 1, Cost: 0.6931471805599454,Theta = 0.00017500000000000003, Bias = 0.0
Iteration 10001, Cost: 0.601583150171683,Theta = 0.3548276024415794, Bias = -0.47187303096081995
Iteration 20001, Cost: 0.5505307863424463,Theta = 0.5303751793133015, Bias = -0.9453515927352263
Iteration 30001, Cost: 0.5093853845889779,Theta = 0.6890681139380744, Bias = -1.3700292906165874
Iteration 40001, Cost: 0.475833643314449,Theta = 0.8331822185532951, Bias = -1.75324098541214
Iteration 50001, Cost: 0.4481126575134082,Theta = 0.964777206650959, Bias = -2.101361628448987
Iteration 60001, Cost: 0.4249037714979318,Theta = 1.0856400168797495, Bias = -2.419739418645759
Iteration 70001, Cost: 0.4052250952011625,Theta = 1.1972807506522725, Bias = -2.7127869514681846
Iteration 80001, Cost: 0.38834277015284324,Theta = 1.3009598028729108, Bias = -2.98412258374647
Iteration 90001, Cost: 0.3737037276216073,Theta = 1.3977247347381418, Bias = -3.2367135384644166
Final values: Theta = 1.488437989761718, Bias = -3.4729790288848634, cost: 0.3609
```

In [ ]: