```
In [1]: import matplotlib.pyplot as plt
          import numpy as np
          %matplotlib inline
          x=np.arange(-6,6,0.01)
          y=1/(1+np.exp(-x))
          plt.plot(x,y)
          plt.title('Sigmoid/Logistic Activation Funciton')
          plt.xlabel('Input')
          plt.ylabel('Output')
 Out[1]: Text(0, 0.5, 'Output')
                       Sigmoid/Logistic Activation Funciton
            1.0
            0.8
            0.6
           Ontbut
0.4
            0.2
             0.0
                              -2
                                   Input
          Activation Function - Sigmoid/Logistic's Derivative
 In [2]: y=1/(1+np.exp(-x))
          dy=y^{*}(1-y)
          plt.plot(x,dy)
          plt.title('Sigmoid/Logistic Activation Funciton Derivative')
          plt.xlabel('Input')
          plt.ylabel('Output')
 Out[2]: Text(0, 0.5, 'Output')
                    Sigmoid/Logistic Activation Funciton Derivative
             0.25
             0.20
           o.15
            0.05
             0.00
                                    Input
          Activation Function - Tanh/Hyperbolic Tangent
 In [3]: x=np.arange(-6,6,0.01)
         y=(2 / (1+ np.exp(-2*x)))-1
          plt.plot(x,y)
          plt.title('Tanh/Hyperbolic Tangent Activation Funciton')
          plt.xlabel('Input')
          plt.ylabel('Output')
 Out[3]: Text(0, 0.5, 'Output')
                     Tanh/Hyperbolic Tangent Activation Funciton
              1.00
              0.75
              0.50
              0.25
              0.00
             -0.25
             -0.50
             -0.75
             -1.00
                                     Input
          Activation Function - Tanh/Hyperbolic Tangent Derivative
 In [4]: dy=1-y**2
          plt.plot(x,dy)
          plt.title('Tanh/Hyperbolic Tangent Activation Funciton Derivative')
          plt.xlabel('Input')
          plt.ylabel('Output')
 Out[4]: Text(0, 0.5, 'Output')
                Tanh/Hyperbolic Tangent Activation Funciton Derivative
            1.0
            0.8
             0.6
            0.2
          Activation Function - ReLu(Rectified Linear Unit)
 In [5]: x=np.arange(-6,6,0.01)
          z=np.zeros(len(x))
          y=np.maximum(z,x)
          plt.plot(x,y)
          plt.title('Rectified Linear Unit Activation Funciton')
          plt.xlabel('Input')
          plt.ylabel('Output')
 Out[5]: Text(0, 0.5, 'Output')
                     Rectified Linear Unit Activation Funciton
                -6
                                  Input
          Activation Function - Leaky-ReLu
In [10]: x=np.arange(-6,6,0.01)
          y = np.where(x > 0, x, x * 0.01)
          plt.plot(x,y)
          plt.title('Leaky ReLu Activation Funciton')
          plt.xlabel('Input')
          plt.ylabel('Output')
Out[10]: Text(0, 0.5, 'Output')
                        Leaky ReLu Activation Funciton
                             -2
                                  Input
          Activation Function - Softplus
 In [7]: x=np.arange(-6,6,0.01)
          y=np.log(1+ np.exp(x))
          plt.plot(x,y)
          plt.title('Softplus Activation Funciton')
plt.xlabel('Input')
          plt.ylabel('Output')
 Out[7]: Text(0, 0.5, 'Output')
                         Softplus Activation Funciton
                             -2
          Activation Function - Softmax
 In [8]: x=np.arange(-6,6,0.01)
          y=np.exp(x - np.max(x))
          z=y / y.sum()
          plt.plot(x,z)
          plt.title('Softmax Activation Funciton')
          plt.xlabel('Input')
          plt.ylabel('Output')
 Out[8]: Text(0, 0.5, 'Output')
                            Softmax Activation Funciton
            0.010
            0.008
            0.006
         o.004 o
            0.002
             0.000
                               -2
                                     Input
          Activation Function - ELU(Exponential Linear Units)
 In [9]: def elu(arr, alpha):
              x=np.arange(-6,6,0.01)
              y=np.linspace(0, len(x), len(x))
              a = []
              for x in arr:
                   if x >= 0:
                       a.append(x)
                   else:
                       a.append(alpha * (np.exp(x)-1))
              return a
          y = elu(x, 1.0)
          plt.plot(x, y)
          plt.title('Exponential Linear Units Activation Funciton')
          plt.xlabel('Input')
          plt.ylabel('Output')
 Out[9]: Text(0, 0.5, 'Output')
                    Exponential Linear Units Activation Funciton
```

Activation Function - Sigmoid/Logistic