

## SVM Kernels Indepth Intuition And Practical Explanation

```
In [1]: 1 import numpy as np
        2 import matplotlib.pyplot as plt
        3
        4 x = np.linspace(-5.0, 5.0, 100) # ye line space -5 se 5 k bichme 100 point create krdega
        5 y = np.sqrt(10**2 - x**2)
```

```
In [2]: 1 x
```

```
Out[2]: array([-5.          , -4.8989899 , -4.7979798 , -4.6969697 , -4.5959596 ,
        -4.49494949, -4.39393939, -4.29292929, -4.19191919, -4.09090909,
        -3.98989899, -3.88888889, -3.78787879, -3.68686869, -3.58585859,
        -3.48484848, -3.38383838, -3.28282828, -3.18181818, -3.08080808,
        -2.97979798, -2.87878788, -2.77777778, -2.67676768, -2.57575758,
        -2.47474747, -2.37373737, -2.27272727, -2.17171717, -2.07070707,
        -1.96969697, -1.86868687, -1.76767677, -1.66666667, -1.56565657,
        -1.46464646, -1.36363636, -1.26262626, -1.16161616, -1.06060606,
        -0.95959596, -0.85858586, -0.75757576, -0.65656566, -0.55555556,
        -0.45454545, -0.35353535, -0.25252525, -0.15151515, -0.05050505,
         0.05050505,  0.15151515,  0.25252525,  0.35353535,  0.45454545,
         0.55555556,  0.65656566,  0.75757576,  0.85858586,  0.95959596,
         1.06060606,  1.16161616,  1.26262626,  1.36363636,  1.46464646,
         1.56565657,  1.66666667,  1.76767677,  1.86868687,  1.96969697,
         2.07070707,  2.17171717,  2.27272727,  2.37373737,  2.47474747,
         2.57575758,  2.67676768,  2.77777778,  2.87878788,  2.97979798,
         3.08080808,  3.18181818,  3.28282828,  3.38383838,  3.48484848,
         3.58585859,  3.68686869,  3.78787879,  3.88888889,  3.98989899,
         4.09090909,  4.19191919,  4.29292929,  4.39393939,  4.49494949,
         4.5959596 ,  4.6969697 ,  4.7979798 ,  4.8989899 ,  5.          ])
```

In [3]: 1 [x,-x]

```

Out[3]: [array([-5.      , -4.8989899 , -4.7979798 , -4.6969697 , -4.5959596 ,
        -4.49494949, -4.39393939, -4.29292929, -4.19191919, -4.09090909,
        -3.98989899, -3.88888889, -3.78787879, -3.68686869, -3.58585859,
        -3.48484848, -3.38383838, -3.28282828, -3.18181818, -3.08080808,
        -2.97979798, -2.87878788, -2.77777778, -2.67676768, -2.57575758,
        -2.47474747, -2.37373737, -2.27272727, -2.17171717, -2.07070707,
        -1.96969697, -1.86868687, -1.76767677, -1.66666667, -1.56565657,
        -1.46464646, -1.36363636, -1.26262626, -1.16161616, -1.06060606,
        -0.95959596, -0.85858586, -0.75757576, -0.65656566, -0.55555556,
        -0.45454545, -0.35353535, -0.25252525, -0.15151515, -0.05050505,
         0.05050505,  0.15151515,  0.25252525,  0.35353535,  0.45454545,
         0.55555556,  0.65656566,  0.75757576,  0.85858586,  0.95959596,
         1.06060606,  1.16161616,  1.26262626,  1.36363636,  1.46464646,
         1.56565657,  1.66666667,  1.76767677,  1.86868687,  1.96969697,
         2.07070707,  2.17171717,  2.27272727,  2.37373737,  2.47474747,
         2.57575758,  2.67676768,  2.77777778,  2.87878788,  2.97979798,
         3.08080808,  3.18181818,  3.28282828,  3.38383838,  3.48484848,
         3.58585859,  3.68686869,  3.78787879,  3.88888889,  3.98989899,
         4.09090909,  4.19191919,  4.29292929,  4.39393939,  4.49494949,
         4.5959596 ,  4.6969697 ,  4.7979798 ,  4.8989899 ,  5.      ]),
array([ 5.      ,  4.8989899 ,  4.7979798 ,  4.6969697 ,  4.5959596 ,
         4.49494949,  4.39393939,  4.29292929,  4.19191919,  4.09090909,
         3.98989899,  3.88888889,  3.78787879,  3.68686869,  3.58585859,
         3.48484848,  3.38383838,  3.28282828,  3.18181818,  3.08080808,
         2.97979798,  2.87878788,  2.77777778,  2.67676768,  2.57575758,
         2.47474747,  2.37373737,  2.27272727,  2.17171717,  2.07070707,
         1.96969697,  1.86868687,  1.76767677,  1.66666667,  1.56565657,
         1.46464646,  1.36363636,  1.26262626,  1.16161616,  1.06060606,
         0.95959596,  0.85858586,  0.75757576,  0.65656566,  0.55555556,
         0.45454545,  0.35353535,  0.25252525,  0.15151515,  0.05050505,
        -0.05050505, -0.15151515, -0.25252525, -0.35353535, -0.45454545,
        -0.55555556, -0.65656566, -0.75757576, -0.85858586, -0.95959596,
        -1.06060606, -1.16161616, -1.26262626, -1.36363636, -1.46464646,
        -1.56565657, -1.66666667, -1.76767677, -1.86868687, -1.96969697,
        -2.07070707, -2.17171717, -2.27272727, -2.37373737, -2.47474747,
        -2.57575758, -2.67676768, -2.77777778, -2.87878788, -2.97979798,
        -3.08080808, -3.18181818, -3.28282828, -3.38383838, -3.48484848,
        -3.58585859, -3.68686869, -3.78787879, -3.88888889, -3.98989899,
        -4.09090909, -4.19191919, -4.29292929, -4.39393939, -4.49494949,
        -4.5959596 , -4.6969697 , -4.7979798 , -4.8989899 , -5.      ])]

```

In [4]:

```
1 y=np.hstack([y,-y])  
2 x=np.hstack([x,-x])
```

In [5]: 1 x

```

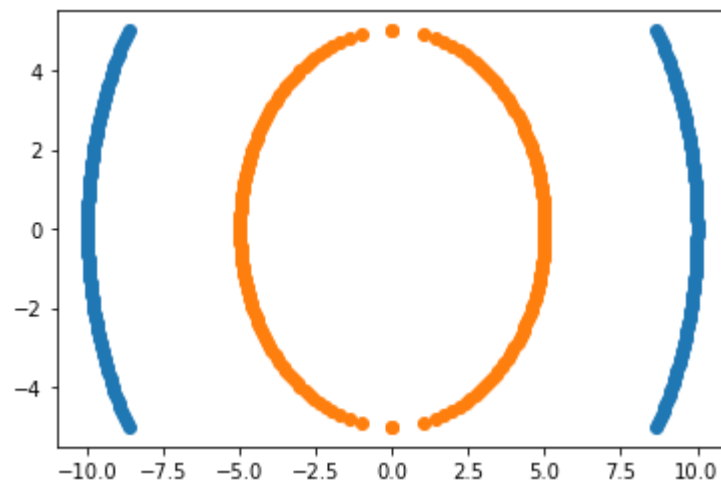
Out[5]: array([-5.          , -4.8989899 , -4.7979798 , -4.6969697 , -4.5959596 ,
        -4.49494949, -4.39393939, -4.29292929, -4.19191919, -4.09090909,
        -3.98989899, -3.88888889, -3.78787879, -3.68686869, -3.58585859,
        -3.48484848, -3.38383838, -3.28282828, -3.18181818, -3.08080808,
        -2.97979798, -2.87878788, -2.77777778, -2.67676768, -2.57575758,
        -2.47474747, -2.37373737, -2.27272727, -2.17171717, -2.07070707,
        -1.96969697, -1.86868687, -1.76767677, -1.66666667, -1.56565657,
        -1.46464646, -1.36363636, -1.26262626, -1.16161616, -1.06060606,
        -0.95959596, -0.85858586, -0.75757576, -0.65656566, -0.55555556,
        -0.45454545, -0.35353535, -0.25252525, -0.15151515, -0.05050505,
         0.05050505,  0.15151515,  0.25252525,  0.35353535,  0.45454545,
         0.55555556,  0.65656566,  0.75757576,  0.85858586,  0.95959596,
         1.06060606,  1.16161616,  1.26262626,  1.36363636,  1.46464646,
         1.56565657,  1.66666667,  1.76767677,  1.86868687,  1.96969697,
         2.07070707,  2.17171717,  2.27272727,  2.37373737,  2.47474747,
         2.57575758,  2.67676768,  2.77777778,  2.87878788,  2.97979798,
         3.08080808,  3.18181818,  3.28282828,  3.38383838,  3.48484848,
         3.58585859,  3.68686869,  3.78787879,  3.88888889,  3.98989899,
         4.09090909,  4.19191919,  4.29292929,  4.39393939,  4.49494949,
         4.5959596 ,  4.6969697 ,  4.7979798 ,  4.8989899 ,  5.          ,
         5.          ,  4.8989899 ,  4.7979798 ,  4.6969697 ,  4.5959596 ,
         4.49494949,  4.39393939,  4.29292929,  4.19191919,  4.09090909,
         3.98989899,  3.88888889,  3.78787879,  3.68686869,  3.58585859,
         3.48484848,  3.38383838,  3.28282828,  3.18181818,  3.08080808,
         2.97979798,  2.87878788,  2.77777778,  2.67676768,  2.57575758,
         2.47474747,  2.37373737,  2.27272727,  2.17171717,  2.07070707,
         1.96969697,  1.86868687,  1.76767677,  1.66666667,  1.56565657,
         1.46464646,  1.36363636,  1.26262626,  1.16161616,  1.06060606,
         0.95959596,  0.85858586,  0.75757576,  0.65656566,  0.55555556,
         0.45454545,  0.35353535,  0.25252525,  0.15151515,  0.05050505,
        -0.05050505, -0.15151515, -0.25252525, -0.35353535, -0.45454545,
        -0.55555556, -0.65656566, -0.75757576, -0.85858586, -0.95959596,
        -1.06060606, -1.16161616, -1.26262626, -1.36363636, -1.46464646,
        -1.56565657, -1.66666667, -1.76767677, -1.86868687, -1.96969697,
        -2.07070707, -2.17171717, -2.27272727, -2.37373737, -2.47474747,
        -2.57575758, -2.67676768, -2.77777778, -2.87878788, -2.97979798,
        -3.08080808, -3.18181818, -3.28282828, -3.38383838, -3.48484848,
        -3.58585859, -3.68686869, -3.78787879, -3.88888889, -3.98989899,
        -4.09090909, -4.19191919, -4.29292929, -4.39393939, -4.49494949,
        -4.5959596 , -4.6969697 , -4.7979798 , -4.8989899 , -5.          ])

```

```
In [6]: 1 x1 = np.linspace(-5.0, 5.0, 100)
        2 y1 = np.sqrt(5**2 - x1**2)
        3 y1 = np.hstack([y1, -y1])
        4 x1 = np.hstack([x1, -x1])
```

```
In [7]: 1 plt.scatter(y, x) #blue point
        2 plt.scatter(y1, x1) #orange point inner one
```

Out[7]: <matplotlib.collections.PathCollection at 0x13fd9f9a550>



In [8]: 1 np.vstack([y,x]).T

Out[8]: array([[ 8.66025404, -5. ],  
 [ 8.71779204, -4.8989899 ],  
 [ 8.77378994, -4.7979798 ],  
 [ 8.82827705, -4.6969697 ],  
 [ 8.88128118, -4.5959596 ],  
 [ 8.93282873, -4.49494949],  
 [ 8.98294476, -4.39393939],  
 [ 9.03165312, -4.29292929],  
 [ 9.07897646, -4.19191919],  
 [ 9.12493632, -4.09090909],  
 [ 9.16955321, -3.98989899],  
 [ 9.21284664, -3.88888889],  
 [ 9.25483518, -3.78787879],  
 [ 9.29553652, -3.68686869],  
 [ 9.3349675 , -3.58585859],  
 [ 9.37314414, -3.48484848],  
 [ 9.41008171, -3.38383838],  
 [ 9.44579475, -3.28282828],  
 [ 9.4802971 , -3.18181818],  
 [ 9.51388408, -3.08080808],  
 [ 9.54664408, -2.98080808],  
 [ 9.57856408, -2.88080808],  
 [ 9.60964408, -2.78080808],  
 [ 9.63988408, -2.68080808],  
 [ 9.66928408, -2.58080808],  
 [ 9.69784408, -2.48080808],  
 [ 9.72556408, -2.38080808],  
 [ 9.75244408, -2.28080808],  
 [ 9.77848408, -2.18080808],  
 [ 9.80368408, -2.08080808],  
 [ 9.82804408, -1.98080808],  
 [ 9.85156408, -1.88080808],  
 [ 9.87424408, -1.78080808],  
 [ 9.89608408, -1.68080808],  
 [ 9.91708408, -1.58080808],  
 [ 9.93724408, -1.48080808],  
 [ 9.95656408, -1.38080808],  
 [ 9.97504408, -1.28080808],  
 [ 9.99268408, -1.18080808],  
 [10.00948408, -1.08080808],  
 [10.02544408, -0.98080808],  
 [10.04056408, -0.88080808],  
 [10.05484408, -0.78080808],  
 [10.06828408, -0.68080808],  
 [10.08088408, -0.58080808],  
 [10.09264408, -0.48080808],  
 [10.10356408, -0.38080808],  
 [10.11364408, -0.28080808],  
 [10.12288408, -0.18080808],  
 [10.13128408, -0.08080808],  
 [10.13884408, 0.02080808],  
 [10.14556408, 0.12080808],  
 [10.15144408, 0.22080808],  
 [10.15648408, 0.32080808],  
 [10.16068408, 0.42080808],  
 [10.16404408, 0.52080808],  
 [10.16656408, 0.62080808],  
 [10.16824408, 0.72080808],  
 [10.16908408, 0.82080808],  
 [10.16908408, 0.92080808],  
 [10.16824408, 1.02080808],  
 [10.16656408, 1.12080808],  
 [10.16404408, 1.22080808],  
 [10.16068408, 1.32080808],  
 [10.15648408, 1.42080808],  
 [10.15144408, 1.52080808],  
 [10.14556408, 1.62080808],  
 [10.13884408, 1.72080808],  
 [10.13128408, 1.82080808],  
 [10.12288408, 1.92080808],  
 [10.11364408, 2.02080808],  
 [10.10356408, 2.12080808],  
 [10.09264408, 2.22080808],  
 [10.08088408, 2.32080808],  
 [10.06828408, 2.42080808],  
 [10.05484408, 2.52080808],  
 [10.04056408, 2.62080808],  
 [10.02544408, 2.72080808],  
 [10.00948408, 2.82080808],  
 [9.99268408, 2.92080808],  
 [9.97504408, 3.02080808],  
 [9.95656408, 3.12080808],  
 [9.93724408, 3.22080808],  
 [9.91708408, 3.32080808],  
 [9.89608408, 3.42080808],  
 [9.87424408, 3.52080808],  
 [9.85156408, 3.62080808],  
 [9.82804408, 3.72080808],  
 [9.80368408, 3.82080808],  
 [9.77848408, 3.92080808],  
 [9.75244408, 4.02080808],  
 [9.72556408, 4.12080808],  
 [9.69784408, 4.22080808],  
 [9.66928408, 4.32080808],  
 [9.63988408, 4.42080808],  
 [9.60964408, 4.52080808],  
 [9.57856408, 4.62080808],  
 [9.54664408, 4.72080808],  
 [9.51388408, 4.82080808],  
 [9.4802971 , 4.92080808],  
 [9.44579475, 5.02080808],  
 [9.41008171, 5.12080808],  
 [9.37314414, 5.22080808],  
 [9.3349675 , 5.32080808],  
 [9.29553652, 5.42080808],  
 [9.25483518, 5.52080808],  
 [9.21284664, 5.62080808],  
 [9.16955321, 5.72080808],  
 [9.12493632, 5.82080808],  
 [9.07897646, 5.92080808],  
 [9.03165312, 6.02080808],  
 [8.98294476, 6.12080808],  
 [8.93282873, 6.22080808],  
 [8.88128118, 6.32080808],  
 [8.82827705, 6.42080808],  
 [8.77378994, 6.52080808],  
 [8.71779204, 6.62080808],  
 [8.66025404, 6.72080808],  
 [8.60125404, 6.82080808],  
 [8.54080404, 6.92080808],  
 [8.47890404, 7.02080808],  
 [8.41555404, 7.12080808],  
 [8.35075404, 7.22080808],  
 [8.28450404, 7.32080808],  
 [8.21680404, 7.42080808],  
 [8.14765404, 7.52080808],  
 [8.07705404, 7.62080808],  
 [8.00500404, 7.72080808],  
 [7.93150404, 7.82080808],  
 [7.85655404, 7.92080808],  
 [7.78015404, 8.02080808],  
 [7.70230404, 8.12080808],  
 [7.62300404, 8.22080808],  
 [7.54225404, 8.32080808],  
 [7.46005404, 8.42080808],  
 [7.37640404, 8.52080808],  
 [7.29130404, 8.62080808],  
 [7.20475404, 8.72080808],  
 [7.11675404, 8.82080808],  
 [7.02730404, 8.92080808],  
 [6.93640404, 9.02080808],  
 [6.84405404, 9.12080808],  
 [6.75025404, 9.22080808],  
 [6.65500404, 9.32080808],  
 [6.55830404, 9.42080808],  
 [6.46015404, 9.52080808],  
 [6.36055404, 9.62080808],  
 [6.25950404, 9.72080808],  
 [6.15700404, 9.82080808],  
 [6.05305404, 9.92080808],  
 [5.94765404, 10.02080808],  
 [5.84080404, 10.12080808],  
 [5.73250404, 10.22080808],  
 [5.62275404, 10.32080808],  
 [5.51155404, 10.42080808],  
 [5.39890404, 10.52080808],  
 [5.28480404, 10.62080808],  
 [5.16925404, 10.72080808],  
 [5.05225404, 10.82080808],  
 [4.93380404, 10.92080808],  
 [4.81390404, 11.02080808],  
 [4.69255404, 11.12080808],  
 [4.56975404, 11.22080808],  
 [4.44550404, 11.32080808],  
 [4.31980404, 11.42080808],  
 [4.19265404, 11.52080808],  
 [4.06405404, 11.62080808],  
 [3.93400404, 11.72080808],  
 [3.80250404, 11.82080808],  
 [3.66955404, 11.92080808],  
 [3.53515404, 12.02080808],  
 [3.39930404, 12.12080808],  
 [3.26200404, 12.22080808],  
 [3.12325404, 12.32080808],  
 [2.98305404, 12.42080808],  
 [2.84140404, 12.52080808],  
 [2.69830404, 12.62080808],  
 [2.55375404, 12.72080808],  
 [2.40775404, 12.82080808],  
 [2.26030404, 12.92080808],  
 [2.11140404, 13.02080808],  
 [1.96105404, 13.12080808],  
 [1.80925404, 13.22080808],  
 [1.65600404, 13.32080808],  
 [1.50130404, 13.42080808],  
 [1.34515404, 13.52080808],  
 [1.18755404, 13.62080808],  
 [1.02850404, 13.72080808],  
 [0.86800404, 13.82080808],  
 [0.70605404, 13.92080808],  
 [0.54265404, 14.02080808],  
 [0.37780404, 14.12080808],  
 [0.21150404, 14.22080808],  
 [0.04375404, 14.32080808],  
 [-0.12544404, 14.42080808],  
 [-0.29619404, 14.52080808],  
 [-0.46850404, 14.62080808],  
 [-0.64237404, 14.72080808],  
 [-0.81780404, 14.82080808],  
 [-0.99480404, 14.92080808],  
 [-1.17337404, 15.02080808],  
 [-1.35352404, 15.12080808],  
 [-1.53525404, 15.22080808],  
 [-1.71856404, 15.32080808],  
 [-1.90345404, 15.42080808],  
 [-2.08992404, 15.52080808],  
 [-2.27797404, 15.62080808],  
 [-2.46760404, 15.72080808],  
 [-2.65881404, 15.82080808],  
 [-2.85160404, 15.92080808],  
 [-3.04597404, 16.02080808],  
 [-3.24192404, 16.12080808],  
 [-3.43945404, 16.22080808],  
 [-3.63856404, 16.32080808],  
 [-3.83925404, 16.42080808],  
 [-4.04152404, 16.52080808],  
 [-4.24537404, 16.62080808],  
 [-4.45080404, 16.72080808],  
 [-4.65781404, 16.82080808],  
 [-4.86640404, 16.92080808],  
 [-5.07657404, 17.02080808],  
 [-5.28832404, 17.12080808],  
 [-5.50165404, 17.22080808],  
 [-5.71656404, 17.32080808],  
 [-5.93305404, 17.42080808],  
 [-6.15112404, 17.52080808],  
 [-6.37077404, 17.62080808],  
 [-6.59200404, 17.72080808],  
 [-6.81481404, 17.82080808],  
 [-7.03920404, 17.92080808],  
 [-7.26517404, 18.02080808],  
 [-7.49272404, 18.12080808],  
 [-7.72185404, 18.22080808],  
 [-7.95256404, 18.32080808],  
 [-8.18485404, 18.42080808],  
 [-8.41872404, 18.52080808],  
 [-8.65417404, 18.62080808],  
 [-8.89120404, 18.72080808],  
 [-9.12981404, 18.82080808],  
 [-9.36999404, 18.92080808],  
 [-9.61174404, 19.02080808],  
 [-9.85505404, 19.12080808],  
 [-10.09992404, 19.22080808],  
 [-10.34635404, 19.32080808],  
 [-10.59435404, 19.42080808],  
 [-10.84392404, 19.52080808],  
 [-11.09505404, 19.62080808],  
 [-11.34774404, 19.72080808],  
 [-11.60200404, 19.82080808],  
 [-11.85782404, 19.92080808],  
 [-12.11520404, 20.02080808],  
 [-12.37414404, 20.12080808],  
 [-12.63464404, 20.22080808],  
 [-12.89670404, 20.32080808],  
 [-13.16032404, 20.42080808],  
 [-13.42550404, 20.52080808],  
 [-13.69224404, 20.62080808],  
 [-13.96054404, 20.72080808],  
 [-14.23040404, 20.82080808],  
 [-14.50182404, 20.92080808],  
 [-14.77480404, 21.02080808],  
 [-15.04934404, 21.12080808],  
 [-15.32544404, 21.22080808],  
 [-15.60310404, 21.32080808],  
 [-15.88232404, 21.42080808],  
 [-16.16310404, 21.52080808],  
 [-16.44544404, 21.62080808],  
 [-16.72934404, 21.72080808],  
 [-17.01480404, 21.82080808],  
 [-17.30182404, 21.92080808],  
 [-17.59040404, 22.02080808],  
 [-17.88054404, 22.12080808],  
 [-18.17224404, 22.22080808],  
 [-18.46550404, 22.32080808],  
 [-18.76032404, 22.42080808],  
 [-19.05670404, 22.52080808],  
 [-19.35464404, 22.62080808],  
 [-19.65414404, 22.72080808],  
 [-19.95520404, 22.82080808],  
 [-20.25782404, 22.92080808],  
 [-20.56200404, 23.02080808],  
 [-20.86774404, 23.12080808],  
 [-21.17504404, 23.22080808],  
 [-21.48390404, 23.32080808],  
 [-21.79432404, 23.42080808],  
 [-22.10630404, 23.52080808],  
 [-22.41984404, 23.62080808],  
 [-22.73494404, 23.72080808],  
 [-23.05160404, 23.82080808],  
 [-23.36982404, 23.92080808],  
 [-23.68960404, 24.02080808],  
 [-24.01094404, 24.12080808],  
 [-24.33384404, 24.22080808],  
 [-24.65830404, 24.32080808],  
 [-24.98432404, 24.42080808],  
 [-25.31190404, 24.52080808],  
 [-25.64104404, 24.62080808],  
 [-25.97174404, 24.72080808],  
 [-26.30400404, 24.82080808],  
 [-26.63782404, 24.92080808],  
 [-26.97320404, 25.02080808],  
 [-27.31014404, 25.12080808],  
 [-27.64864404, 25.22080808],  
 [-27.98870404, 25.32080808],  
 [-28.33032404, 25.42080808],  
 [-28.67350404, 25.52080808],  
 [-29.01824404, 25.62080808],  
 [-29.36454404, 25.72080808],  
 [-29.71240404, 25.82080808],  
 [-30.06182404, 25.92080808],  
 [-30.41280404, 26.02080808],  
 [-30.76534404, 26.12080808],  
 [-31.11944404, 26.22080808],  
 [-31.47510404, 26.32080808],  
 [-31.83232404, 26.42080808],  
 [-32.19110404, 26.52080808],  
 [-32.55144404, 26.62080808],  
 [-32.91334404, 26.72080808],  
 [-33.27680404, 26.82080808],  
 [-33.64182404, 26.92080808],  
 [-34.00840404, 27.02080808],  
 [-34.37654404, 27.12080808],  
 [-34.74624404, 27.22080808],  
 [-35.11760404, 27.32080808],  
 [-35.49062404, 27.42080808],  
 [-35.86530404, 27.52080808],  
 [-36.24164404, 27.62080808],  
 [-36.61964404, 27.72080808],  
 [-36.99930404, 27.82080808],  
 [-37.38062404, 27.92080808],  
 [-37.76360404, 28.02080808],  
 [-38.14824404, 28.12080808],  
 [-38.53454404, 28.22080808],  
 [-38.92250404, 28.32080808],  
 [-39.31212404, 28.42080808],  
 [-39.70340404, 28.52080808],  
 [-40.09634404, 28.62080808],  
 [-40.49094404, 28.72080808],  
 [-40.88720404, 28.82080808],  
 [-41.28512404, 28.92080808],  
 [-41.68470404, 29.02080808],  
 [-42.08594404, 29.12080808],  
 [-42.48882404, 29.22080808],  
 [-42.89334404, 29.32080808],  
 [-43.29950404, 29.42080808],  
 [-43.70732404, 29.52080808],  
 [-44.11680404, 29.62080808],  
 [-44.52794404, 29.72080808],  
 [-44.94074404, 29.82080808],  
 [-45.35520404, 29.92080808],  
 [-45.77132404, 30.02080808],  
 [-46.18910404, 30.12080808],  
 [-46.60854404, 30.22080808],  
 [-47.02964404, 30.32080808],  
 [-47.45240404, 30.42080808],  
 [-47.87682404, 30.52080808],  
 [-48.30290404, 30.62080808],  
 [-48.73064404, 30.72080808],  
 [-49.15994404, 30.82080808],  
 [-49.59090404, 30.92080808],  
 [-50.02352404, 31.02080808],  
 [-50.45782404, 31.12080808],  
 [-50.89384404, 31.22080808],  
 [-51.33158404, 31.32080808],  
 [-51.77094404, 31.42080808],  
 [-52.21192404, 31.52080808],  
 [-52.65452404, 31.62080808],  
 [-53.09874404, 31.72080808],  
 [-53.54458404, 31.82080808],  
 [-53.99204404, 31.92080808],  
 [-54.44112404, 32.02080808],  
 [-54.89182404, 32.12080808],  
 [-55.34414404, 32.22080808],  
 [-55.79808404, 32.32080808],  
 [-56.25364404, 32.42080808],  
 [-56.71082404, 32.52080808],  
 [-57.16962404, 32.62080808],  
 [-57.63004404, 32.72080808],  
 [-58.09208404, 32.82080808],  
 [-58.55574404, 32.92080808],  
 [-59.02102404, 33.02080808],  
 [-59.48792404, 33.12080808],  
 [-59.95644404, 33.22080808],  
 [-60.42658404, 33.32080808],  
 [-60.89834404, 33.42080808],  
 [-61.37172404, 33.52080808],  
 [-61.84672404, 33.62080808],  
 [-62.32334404, 33.72080808],  
 [-62.80158404, 33.82080808],  
 [-63.28144404, 33.92080808],  
 [-63.76292404, 34.02080808],  
 [-64.24602404, 34.12080808],  
 [-64.73074404, 34.22080808],  
 [-65.21708404, 34.32080808],  
 [-65.70504404, 34.42080808],  
 [-66.19462404, 34.52080808],  
 [-66.68582404, 34.62080808],  
 [-67.17864404, 34.72080808],  
 [-67.67308404, 34.82080808],  
 [-68.16914404, 34.92080808],  
 [-68.66682404, 35.02080808],  
 [-69.16612404, 35.12080808],  
 [-69.66704404, 35.22080808],  
 [-70.16958404, 35.32080808],  
 [-70.67376404, 35.42080808

In [10]: 1 df.tail()

Out[10]:

	X1	X2	Y
195	-1.969049	-4.59596	1
196	-1.714198	-4.69697	1
197	-1.406908	-4.79798	1
198	-0.999949	-4.89899	1
199	-0.000000	-5.00000	1

In [11]: 1 *### Independent and Dependent features*  
2 X = df.iloc[:, :2]  
3 y = df.Y

In [12]: 1 y

Out[12]: 0 0  
1 0  
2 0  
3 0  
4 0  
..  
195 1  
196 1  
197 1  
198 1  
199 1  
Name: Y, Length: 400, dtype: int64

In [13]: 1 *## Split the dataset into train and test*  
2 from sklearn.model\_selection import train\_test\_split  
3 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=0)

In [14]: 1 y\_train

Out[14]: 50 1  
63 0  
112 1  
159 0  
83 1  
..  
123 1  
192 0  
117 0  
47 0  
172 0  
Name: Y, Length: 300, dtype: int64

```
In [15]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="linear")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[15]: 0.45

```
In [16]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="poly")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[16]: 0.59

```
In [17]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="rbf")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[17]: 1.0



```
In [18]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="sigmoid")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[18]: 0.51

## accuracy before appying polynomial kernal

rbf has more accuracy , so for this circular type of data distribution use rbf  
automatically below thing will be executed by rbf

## Polynomial Kernel

$$K(x, y) = (x^T y + c)^d$$

```
In [19]: 1 # We need to find components for the Polynomial Kernel
2 #X1,X2,X1_square,X2_square,X1*X2
3 df['X1_Square']= df['X1']**2
4 df['X2_Square']= df['X2']**2
5 df['X1*X2'] = (df['X1'] *df['X2'])
6 df.head()
```

Out[19]:

	X1	X2	Y	X1_Square	X2_Square	X1*X2
0	8.660254	-5.00000	0	75.000000	25.000000	-43.301270
1	8.717792	-4.89899	0	75.999898	24.000102	-42.708375
2	8.773790	-4.79798	0	76.979390	23.020610	-42.096467
3	8.828277	-4.69697	0	77.938476	22.061524	-41.466150
4	8.881281	-4.59596	0	78.877155	21.122845	-40.818009

```
In [20]: 1 ### Independent and Dependent features
2 X = df[['X1','X2','X1_Square','X2_Square','X1*X2']]
3 y = df['Y']
```

```
In [21]: 1 y
```

```
Out[21]: 0    0
1    0
2    0
3    0
4    0
..
195   1
196   1
197   1
198   1
199   1
Name: Y, Length: 400, dtype: int64
```

```
In []: 1
```

```
In [22]: 1
2 X_train, X_test, y_train, y_test = train_test_split(X, y,
3                                           test_size = 0.25,
4                                           random_state = 0)
5
```

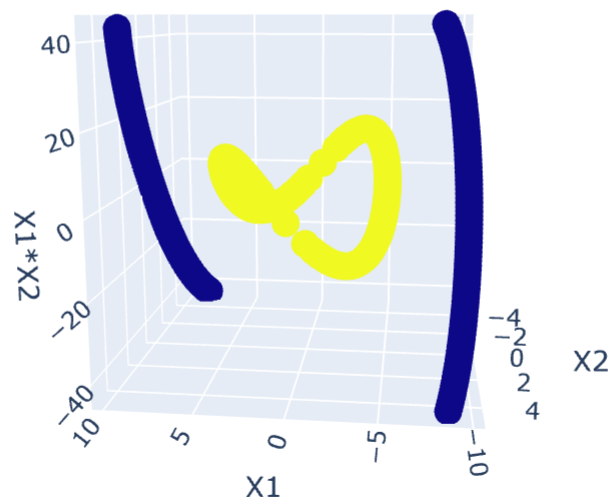
In [23]: 1 X\_train

Out[23]:

	X1	X2	X1_Square	X2_Square	X1*X2
50	4.999745	0.050505	24.997449	0.002551	0.252512
63	9.906589	1.363636	98.140496	1.859504	13.508984
112	-3.263736	3.787879	10.651974	14.348026	-12.362637
159	-9.953852	-0.959596	99.079176	0.920824	9.551676
83	3.680983	3.383838	13.549638	11.450362	12.455852
...	...	...	...	...	...
123	-4.223140	2.676768	17.834915	7.165085	-11.304366
192	-9.031653	-4.292929	81.570758	18.429242	38.772248
117	-9.445795	3.282828	89.223038	10.776962	-31.008922
47	9.996811	-0.252525	99.936231	0.063769	-2.524447
172	-9.738311	-2.272727	94.834711	5.165289	22.132526

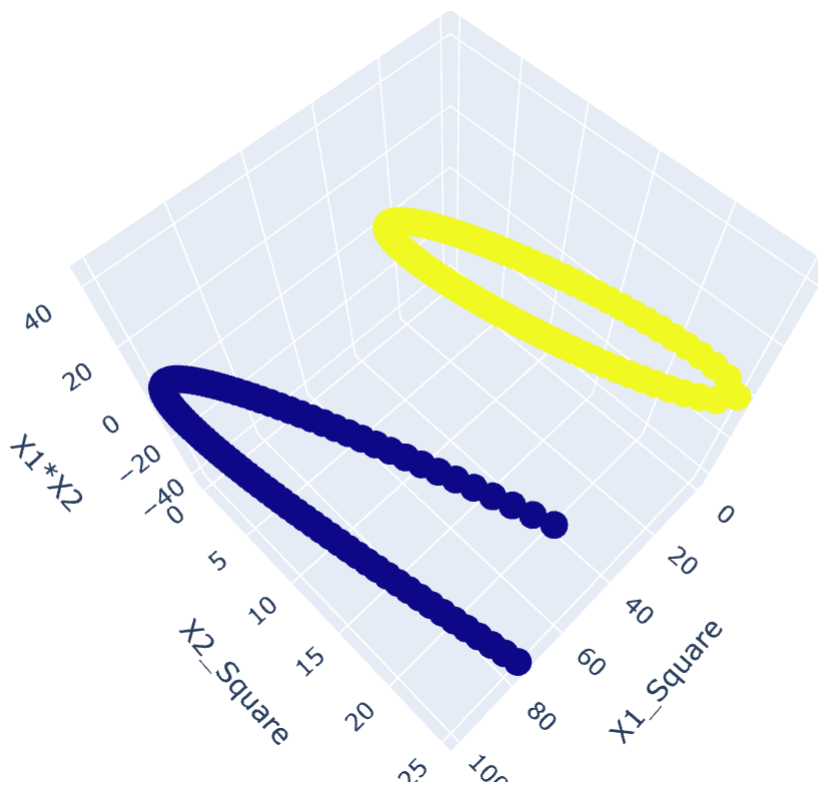
300 rows × 5 columns

```
In [24]: 1 import plotly.express as px
2
3 fig = px.scatter_3d(df, x='X1', y='X2', z='X1*X2',
4                   color='Y')
5 fig.show()
```



```
In [25]: 1 # ye upar wala graph jb apan ne normally data plot kr re hai tbka hai
2 # now isme dekh skte toh ye easily seprable nahi hai
3 # toh apan ne X1_square, X2_square nikala jissne niche wala graph bnra hai
```

```
In [26]: 1  
2 fig = px.scatter_3d(df, x='X1_Square', y='X2_Square', z='X1*X2',  
3                 color='Y')  
4 fig.show()
```



```
In [27]: 1 # data segregate kr diya pura  
2 # ab niche wali accuracy sari 1 ari hai, kyoki seprate kr skte hai
```

```
In [28]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="linear")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[28]: 1.0

```
In [29]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="poly")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[29]: 1.0

```
In [30]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="rbf")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[30]: 1.0

```
In [31]: 1 from sklearn.svm import SVC
2 from sklearn.metrics import accuracy_score
3 classifier = SVC(kernel="sigmoid")
4 classifier.fit(X_train, y_train)
5 y_pred = classifier.predict(X_test)
6 accuracy_score(y_test, y_pred)
```

Out[31]: 1.0

```
In [32]: 1 # yaha se ye nishkarsh nikalta hai ki
2 # data ko split 3d me krdege toh accuracy nikal skti hai
3 # toh uske liye apan ne polynomail kernal k formule lagaye or new data point create kr diye , jisse sari accuracy 1.0 agai
4 # or rbf kernal jo hai vo usko automatically 3D me seprate krdega , bina polynomial lagye . jese line 17 and 30 me "rbf" apply kia hai
```

