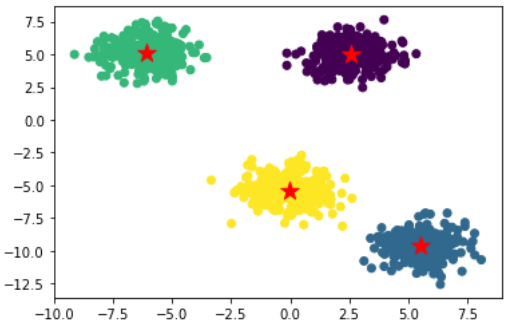
DBSCAN

In [4]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels as sm
```

In [5]:
```
from sklearn.datasets import make_blobs

X,_=make_blobs(n_samples=1000,n_features=2,
               centers=4,random_state=10)
X
```

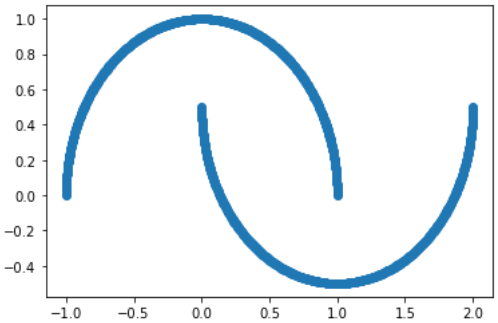
Out[5]:
```
array([[ 4.4016599 , -9.42456185],
       [-6.12795872,  5.01025395],
       [-6.25332117,  5.01582549],
       ...,
       [-5.29208972,  4.54111279],
       [ 5.77276444, -8.64259652],
       [ 1.54076696,  4.06513159]])
```

In [9]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, random_state=10)
kmeans.fit(X)

# plt.figure(figsize=(10,8))
plt.scatter(X[:,0],X[:,1],c=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0],
            kmeans.cluster_centers_[:,1],
            c='r', marker='*',s=200);
```

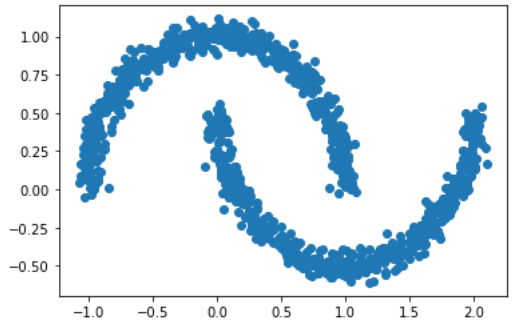


Try with another dataset

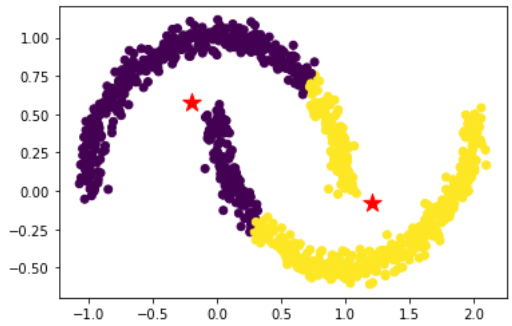
In [10]:
```
from sklearn.datasets import make_moons
X,_=make_moons(n_samples=1000, random_state=10)
plt.scatter(X[:,0],X[:,1]);
```



In [12]:
```
X,_=make_moons(n_samples=1000, random_state=10,noise=0.05)
plt.scatter(X[:,0],X[:,1]);
```



In [13]:
```
# Building the K Means model

k_moons=KMeans(n_clusters=2, random_state=10)
k_moons.fit(X)
plt.scatter(X[:,0],X[:,1],c=k_moons.labels_)
plt.scatter(k_moons.cluster_centers_[:,0],
            k_moons.cluster_centers_[:,1],
            c='r',marker='*',s=200);
```

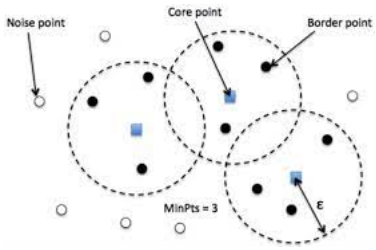


Drawback of K means

- Only Spherical clustering
- Need to know K in advance
- Unduly influenced by outliers

DBSCAN

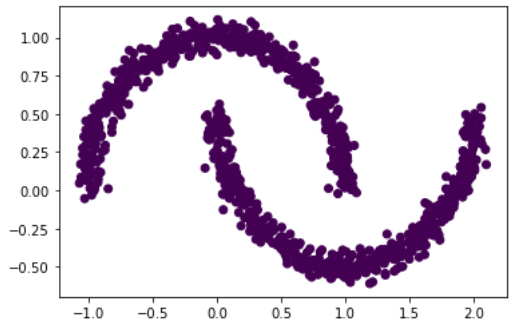
Density Based Spatial Clustering of Applications with Noise



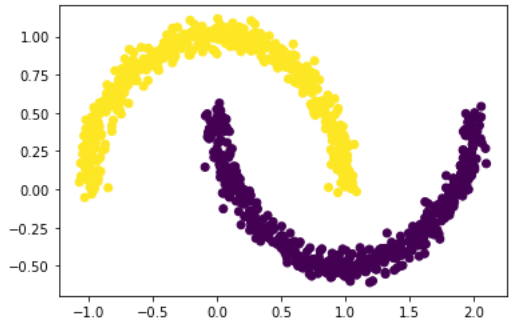
In [14]:
```
from sklearn.cluster import DBSCAN

db=DBSCAN(eps=0.5, min_samples=5)
db.fit(X)
```

Out[14]:
```
DBSCAN()
```

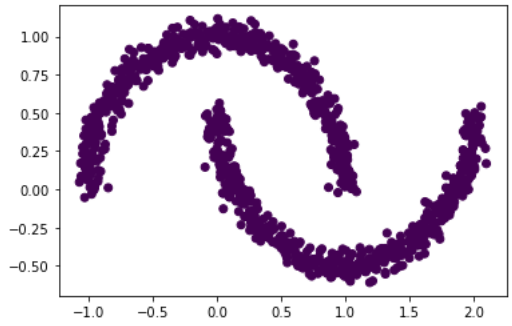
In [15]:
```
plt.scatter(X[:,0],X[:,1],c=db.labels_);
```



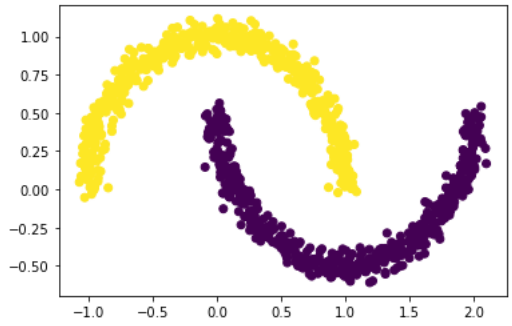
In [16]:
```
db=DBSCAN(eps=0.2, min_samples=5)
db.fit(X)
plt.scatter(X[:,0],X[:,1],c=db.labels_);
```



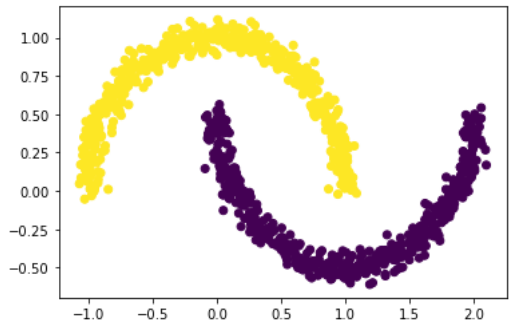
In [17]:
```
db=DBSCAN(eps=0.4, min_samples=5)
db.fit(X)
plt.scatter(X[:,0],X[:,1],c=db.labels_);
```



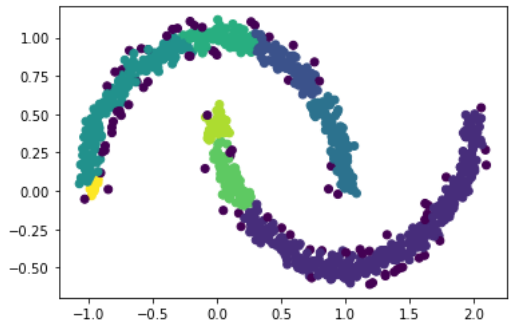
In [18]:
```
db=DBSCAN(eps=0.3, min_samples=5)
db.fit(X)
plt.scatter(X[:,0],X[:,1],c=db.labels_);
```



In [19]:
```
db=DBSCAN(eps=0.1, min_samples=5)
db.fit(X)
plt.scatter(X[:,0],X[:,1],c=db.labels_);
```



In [20]:
```
db=DBSCAN(eps=0.05, min_samples=5)
db.fit(X)
plt.scatter(X[:,0],X[:,1],c=db.labels_);
```



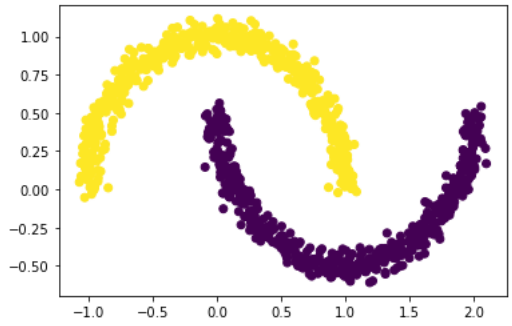
In [21]:
```
db.labels_
```

Out[21]:
```
array([ 0,  0, -1,  1,  0,  0,  1,  4,  2,  1,  0,  3,  1,  0,  3,  3,  0,
        0,  0,  3,  0,  4,  1,  4,  3,  2,  4,  0,  1,  0,  3,  5,  3,  5,
        3,  3,  3,  0,  3,  0,  3,  0,  0,  5,  3,  0,  0,  0,  6, -1,  2,
        2,  3,  3,  3,  1,  3,  3,  5,  2,  0,  0,  0,  0,  0,  0,  5,  0,
        0,  3,  3,  3,  3,  1,  0,  4,  0,  3,  3,  5,  0,  3,  5,  1,  3,
        3,  0,  0,  0,  3,  5,  6,  2,  1,  0,  1,  2,  0,  3,  5,  1,  0,
        0,  0,  0,  0,  3,  3,  3,  2,  2,  3,  2,  0,  0,  3, -1,  0, -1,
        2,  0,  0,  3,  1, -1,  4,  2,  0,  0,  0,  1,  4, -1,  0,  1,  0,
        0,  0, -1,  2,  5,  0,  0,  1,  0,  0,  5,  2,  1, -1,  5,  3,  0,
        0,  3,  0,  0,  0,  3, -1,  4,  0,  1,  3,  0,  3,  6,  0,  0,  0,
        0,  2,  4,  2,  1,  3,  0,  1,  6,  3,  0,  0, -1,  3,  3,  4, -1,
        0, -1, -1,  0,  0,  3,  5,  4,  4,  4,  3,  3,  6, -1,  4,  4,  5,
        3,  4,  5,  6,  0,  2, -1,  0,  0,  0,  0, -1,  0, -1,  4,  5,  5,
        0,  2,  0,  1,  5,  0,  1,  0,  1,  0,  0,  0,  1, -1,  3,  4,  0,
        3,  5,  5,  6,  2,  3,  2,  0,  4,  5,  3,  1,  1,  0,  0,  4,  5,
        4,  1,  0,  3,  0,  1,  6,  0,  0,  0,  0,  3,  0,  1,  3,  2,  3,
        3,  2,  2,  6,  4,  2,  0,  2,  0,  5,  4,  0,  4,  0, -1,  0, -1,
        0,  0,  0,  0,  5,  2,  0,  0,  3,  3,  4,  0,  0,  5,  2,  2, -1,
        2,  4,  0,  4,  0,  1,  0,  0,  1,  3,  7,  3,  0,  1,  3,  3,  0,
       -1,  7,  0, -1,  2,  1,  6,  0,  1,  0,  2,  3,  0,  2,  3,  2,  1,
        0,  2,  0,  0,  1,  3,  0,  0,  0,  4,  3,  0,  3,  0,  2,  4,  3,
        0, -1,  3,  1,  2, -1,  0,  4,  1,  0,  0,  0, -1,  3,  0,  3,  3,
       -1,  1,  5, -1,  1,  0,  2,  0,  0,  2,  0,  2,  0,  4,  3, -1,  6,
        3,  6,  2, -1,  1,  2,  0, -1,  2,  0,  1,  0,  0,  7,  3,  2,  0,
        4,  0,  4,  5,  3,  7,  3,  1,  0,  0,  6,  5,  3,  1,  3,  2,  1,
        0,  0, -1,  0,  2,  0,  4,  0,  0,  0,  2,  2,  0, -1,  0,  2,  0,
        4,  0,  0,  4,  2,  0,  5,  0,  3,  6,  0,  3,  5,  2,  3,  0,  5,
        5,  7,  3,  4,  0,  0, -1,  5,  3,  4,  1,  0,  4,  4,  4,  3,  1,
        3,  3, -1,  0,  0,  0,  3,  0,  6,  4,  0,  3,  6,  0,  0,  0,  0,
        3,  0,  5,  5,  0,  0,  0,  7,  4,  2,  4,  0, -1,  2,  5,  0,  5,
        0,  3,  3,  0, -1,  3,  5,  0,  5,  0,  1,  3,  1,  0,  0,  0,  0,
        1,  0,  2,  4, -1,  2,  5, -1,  0,  2,  5,  3,  2,  1,  3,  1,  2,
        3,  2, -1,  0,  0,  1,  4, -1,  0,  0,  4,  0,  1,  3,  6,  2, -1,
        5,  3,  1,  0,  0,  5, -1,  0,  0,  2,  4, -1,  3,  1,  3,  0,  3,
        2,  0,  0,  0,  0,  0,  2,  0,  0,  0,  2,  3,  2,  0,  5,  0,  0,
        0,  3,  0,  3,  5,  2, -1,  7, -1,  4,  3, -1,  3,  1,  7,  3,  4,
        1,  4,  1,  0,  0,  5,  5,  3,  3,  0,  0,  2,  0,  1,  0,  3,  0,
        0,  0,  0,  3,  5,  5,  0,  0,  0,  2,  6,  0,  4,  0,  3,  2,  2,
        4,  0,  3,  0, -1,  0,  6,  4,  3,  1, -1,  0,  1,  2,  0, -1,  5,
       -1,  3,  0,  0,  0,  6,  3,  1,  4,  1,  0,  0,  4,  0,  0,  4,  0,
        0,  4,  6,  2, -1,  4,  2,  3,  5,  4,  2,  0,  0,  3,  0,  0,  0,
        1,  4,  2,  0,  5,  0,  0,  1,  3,  4, -1,  0, -1,  3,  3,  0,  5,
        3,  0,  0,  1,  5,  0,  0,  3,  3,  3,  0,  5,  3,  3,  3,  0,  0,
        2,  3,  1,  3,  4,  3,  0, -1,  2,  2,  0,  0,  0,  3,  0,  5,  0,
        0,  3,  3,  5,  0,  3,  2, -1,  0,  1,  2, -1,  6,  2,  1,  0,  0,
        4,  0, -1, -1,  0,  0,  0,  0,  4,  1,  3,  3,  3,  1,  0,  1,  6,
        0,  2,  1,  4, -1,  2,  3,  5, -1,  3,  0,  6,  0,  0,  4,  3,  1,
        0,  0,  5,  4,  3, -1,  0,  0,  5,  3, -1,  3,  0,  0,  0,  0,  0,
        0,  5,  0,  3,  3,  0, -1,  0,  3,  2,  0,  6,  5,  0,  0,  2, -1,
        2,  3,  2,  0,  0,  5,  0,  3,  3,  0,  1,  3,  0,  1,  1,  3,  4,
       -1,  3,  4,  2,  2,  0,  0,  3,  3,  0,  4,  3,  2,  0,  0,  4,  2,
        2,  6,  0,  3,  3,  0,  0,  6,  6,  0,  0,  0,  5,  0,  0,  0,  3,
        0, -1,  4,  0,  5,  1,  0,  2,  3,  0,  0, -1,  6,  0,  0,  5, -1,
        1,  0,  0,  3,  0,  3,  3,  1,  3,  0,  0,  0,  0,  2,  0,  2,  3,
        0,  0,  3, -1, -1,  0,  1, -1,  0,  0,  0,  2,  0,  3,  0,  2,  3,
        4,  2,  0,  0,  0,  5,  0,  3,  0,  2,  5, -1,  0, -1,  5,  3,  0,
        2,  0,  0,  3,  0,  0,  3,  3, -1,  2, -1,  2,  0,  0,  0,  0,  1,
        5,  5,  1,  3,  3,  2, -1,  4, -1,  6, -1,  3,  6,  2,  3,  4,  2,
        2,  0,  0, -1,  0,  0,  1,  1,  3, -1,  5,  0,  1,  4])
```

In [24]:
```
np.unique(db.labels_)
```

Out[24]:
```
array([-1,  0,  1,  2,  3,  4,  5,  6,  7])
```

In [25]:
```
# Final

db=DBSCAN(eps=0.1, min_samples=4)
db.fit(X)
plt.scatter(X[:,0],X[:,1],c=db.labels_);
```



In [26]:
```
np.unique(db.labels_)
```

Out[26]:
```
array([0, 1])
```

In [27]:
```
db.components_
```

Out[27]:
```
array([[ 1.58003725, -0.2637535 ],
       [ 1.92032444,  0.25263852],
       [ 0.1045824 ,  0.22931663],
       ...,
       [ 2.0380175 ,  0.45642281],
       [ 0.74283487,  0.62906381],
       [-0.20898842,  0.99377143]])
```

In [28]:
```
db.core_sample_indices_
```

Out[28]:
```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259,
       260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
       273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285,
       286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298,
       299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311,
       312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324,
       325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337,
       338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350,
       351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363,
       364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376,
       378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390,
       391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403,
       404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416,
       417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429,
       430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
       443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455,
       456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468,
       469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481,
       482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494,
       495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507,
       508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520,
       521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533,
       534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546,
       547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559,
       560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 573,
       574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586,
       587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599,
       600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612,
       613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625,
       626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638,
       639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651,
       652, 653, 654, 655, 657, 658, 659, 660, 661, 662, 663, 664, 665,
       666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678,
       679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691,
       692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704,
       705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717,
       718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730,
       731, 732, 733, 734, 735, 736, 737, 739, 740, 741, 742, 743, 744,
       745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757,
       758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770,
       771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783,
       784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796,
       797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809,
       810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822,
       823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835,
       836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848,
       849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861,
       862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874,
       875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887,
       888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900,
       901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913,
       914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926,
       927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939,
       940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952,
       953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965,
       966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978,
       979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991,
       992, 993, 994, 995, 996, 997, 998, 999])
```

In [29]:
```
len(db.components_)
```

Out[29]:
```
996
```

In [30]:
```
len(db.core_sample_indices_)
```

Out[30]:
```
996
```

In [32]:
```
X[0]
```

Out[32]:
```
array([ 1.58003725, -0.2637535 ])
```

Prediction

In [33]:
```
db.predict(X)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [33], in <cell line: 1>()
----> 1 db.predict(X)

AttributeError: 'DBSCAN' object has no attribute 'predict'
```

X= components, taken as feature set

y= labels of components, taken as target

Formulate a classification problem by splitting train-test etc.

Prediction using KNeighborsClassifier

In [35]:
```
X_new= db.components_
y=db.labels_[db.core_sample_indices_]
```

In [37]:
```
X_new
```

Out[37]:
```
array([[ 1.58003725, -0.2637535 ],
       [ 1.92032444,  0.25263852],
       [ 0.1045824 ,  0.22931663],
       ...,
       [ 2.0380175 ,  0.45642281],
       [ 0.74283487,  0.62906381],
       [-0.20898842,  0.99377143]])
```

In [38]:
```
y
```

Out[38]:
```
array([0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 1, 1])
```

In [39]:
```
from sklearn.model_selection import train_test_split

X_train, X_test,y_train,y_test=train_test_split(X_new,y,test_size=0.2, random_state=10)

X_train.shape, X_test.shape,y_train.shape,y_test.shape
```

Out[39]:
```
((796, 2), (200, 2), (796,), (200,))
```

In [40]:
```
from sklearn.neighbors import KNeighborsClassifier
kn=KNeighborsClassifier(n_neighbors=20)
kn.fit(X_train,y_train)
```

Out[40]:
```
KNeighborsClassifier(n_neighbors=20)
```

In [41]:
```
from sklearn.metrics import classification_report

report=classification_report(y_test,kn.predict(X_test))
print(report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        94
           1       1.00      1.00      1.00       106

    accuracy                           1.00       200
   macro avg       1.00      1.00      1.00       200
weighted avg       1.00      1.00      1.00       200
```

In [42]:
```
new_X=np.array([[1.456, 0.2334]])
```

In [43]:
```
kn.predict(new_X)
```

Out[43]:
```
array([0])
```

In [ ]:
```

```