**C++ program:**

What I Observed in the Given C++ Program are explained in detail below with all the 5 stages of KNN Algorithm implementation.

**Data:**

|  |  |  |
| --- | --- | --- |
| **X** | **Y** | **Val** |
| 1 | 12 | 0 |
| 2 | 5 | 0 |
| 5 | 3 | 1 |
| 3 | 2 | 1 |
| 3 | 6 | 0 |
| 1.5 | 9 | 1 |
| 7 | 2 | 1 |
| 6 | 1 | 1 |
| 3.8 | 3 | 1 |
| 3 | 10 | 0 |
| 5.6 | 4 | 1 |
| 4 | 2 | 1 |
| 3.5 | 8 | 0 |
| 2 | 11 | 0 |
| 2 | 5 | 1 |
| 2 | 9 | 0 |
| 1 | 7 | 0 |

**Testing point**

**P:** X = 2.5 and Y = 7

1. **Determine parameter K = number of nearest neighbors.**

Finding ‘k’ value is not easy. Usually k is chosen to be odd number if the number of classes is 2. Another simplest approach to select k is finding the sqrt (total number of items). Smaller k gives more noisy data and larger k value makes it take more time to compute.

This k value is a value that considers the nearest distances form the point to the other training points. which is used to predict the class that new point should be in. In both cases the k is doing same work.

This is what the K = 3 is in the program. Since, there are two classes 0 and 1.

1. **Calculate the distance between the query-instance and all the training samples.**

Which is sqrt((training point x – new point x) \* (training point x – new point x) + (training point y – new point y) \* (training point y – new point y))

|  |  |  |
| --- | --- | --- |
| **X** | **Y** | **Distance** |
| 1 | 12 | Sqrt ((1 – 2.5)2 + (12 - 7)2) =  5.2201532544552753 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 |
| 5 | 3 | Sqrt ((5 – 2.5)2 + (3 - 7)2) =  4.7169905660283016 |
| 3 | 2 | Sqrt ((3 – 2.5)2 + (2 - 7)2) =  5.0249378105604450 |
| 3 | 6 | Sqrt ((3 – 2.5)2 + (6 - 7)2) =  1.1180339887498949 |
| 1.5 | 9 | Sqrt ((1.5 – 2.5)2 + (9 - 7)2) =  2.2360679774997898 |
| 7 | 2 | Sqrt ((7 – 2.5)2 + (2 - 7)2) =  6.7268120235368549 |
| 6 | 1 | Sqrt ((6 – 2.5)2 + (1 - 7)2) =  6.9462219947249020 |
| 3.8 | 3 | Sqrt ((3.8 – 2.5)2 + (3 - 7)2) =  4.2059481689626175 |
| 3 | 10 | Sqrt ((3 – 2.5)2 + (10 - 7)2) =  3.0413812651491097 |
| 5.6 | 4 | Sqrt ((5.6 – 2.5)2 + (4 - 7)2) =  4.3139309220245980 |
| 4 | 2 | Sqrt ((4 – 2.5)2 + (2 - 7)2) =  5.2201532544552753 |
| 3.5 | 8 | Sqrt ((3.5 – 2.5)2 + (8 - 7)2) =  1.4142135623730951 |
| 2 | 11 | Sqrt ((2 – 2.5)2 + (11 - 7)2) =  4.0311288741492746 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 |
| 2 | 9 | Sqrt ((2 – 2.5)2 + (9 - 7)2) =  2.0615528128088303 |
| 1 | 7 | Sqrt ((1 – 2.5)2 + (7 - 7)2) =  1.5000000000000000 |

1. **Sort the distance and determine nearest neighbors based on the K-th minimum distance**

Sorting the distances in ascending order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **Distance** | **Rank Minimum Distance** | **Does it include in 3- nearest neighbors?** |
| 1 | 12 | Sqrt ((1 – 2.5)2 + (12 - 7)2) =  5.2201532544552753 | 14 | No |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 6 | No |
| 5 | 3 | Sqrt ((5 – 2.5)2 + (3 - 7)2) =  4.7169905660283016 | 12 | No |
| 3 | 2 | Sqrt ((3 – 2.5)2 + (2 - 7)2) =  5.0249378105604450 | 13 | No |
| 3 | 6 | Sqrt ((3 – 2.5)2 + (6 - 7)2) =  1.1180339887498949 | 1 | Yes |
| 1.5 | 9 | Sqrt ((1.5 – 2.5)2 + (9 - 7)2) =  2.2360679774997898 | 7 | No |
| 7 | 2 | Sqrt ((7 – 2.5)2 + (2 - 7)2) =  6.7268120235368549 | 16 | No |
| 6 | 1 | Sqrt ((6 – 2.5)2 + (1 - 7)2) =  6.9462219947249020 | 17 | No |
| 3.8 | 3 | Sqrt ((3.8 – 2.5)2 + (3 - 7)2) =  4.2059481689626175 | 10 | No |
| 3 | 10 | Sqrt ((3 – 2.5)2 + (10 - 7)2) =  3.0413812651491097 | 8 | No |
| 5.6 | 4 | Sqrt ((5.6 – 2.5)2 + (4 - 7)2) =  4.3139309220245980 | 11 | No |
| 4 | 2 | Sqrt ((4 – 2.5)2 + (2 - 7)2) =  5.2201532544552753 | 15 | No |
| 3.5 | 8 | Sqrt ((3.5 – 2.5)2 + (8 - 7)2) =  1.4142135623730951 | 2 | Yes |
| 2 | 11 | Sqrt ((2 – 2.5)2 + (11 - 7)2) =  4.0311288741492746 | 9 | No |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 5 | No |
| 2 | 9 | Sqrt ((2 – 2.5)2 + (9 - 7)2) =  2.0615528128088303 | 4 | No |
| 1 | 7 | Sqrt ((1 – 2.5)2 + (7 - 7)2) =  1.5000000000000000 | 3 | Yes |

1. **Gather the category Val of the nearest neighbors**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Y** | **Distance** | **Rank Minimum Distance** | **Does it include in 3- nearest neighbors?** | **Category of nearest neighbor**  **Val** |
| 1 | 12 | Sqrt ((1 – 2.5)2 + (12 - 7)2) =  5.2201532544552753 | 14 | No | 0 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 6 | No | 0 |
| 5 | 3 | Sqrt ((5 – 2.5)2 + (3 - 7)2) =  4.7169905660283016 | 12 | No | 1 |
| 3 | 2 | Sqrt ((3 – 2.5)2 + (2 - 7)2) =  5.0249378105604450 | 13 | No | 1 |
| 3 | 6 | Sqrt ((3 – 2.5)2 + (6 - 7)2) =  1.1180339887498949 | 1 | Yes | 0 |
| 1.5 | 9 | Sqrt ((1.5 – 2.5)2 + (9 - 7)2) =  2.2360679774997898 | 7 | No | 1 |
| 7 | 2 | Sqrt ((7 – 2.5)2 + (2 - 7)2) =  6.7268120235368549 | 16 | No | 1 |
| 6 | 1 | Sqrt ((6 – 2.5)2 + (1 - 7)2) =  6.9462219947249020 | 17 | No | 1 |
| 3.8 | 3 | Sqrt ((3.8 – 2.5)2 + (3 - 7)2) =  4.2059481689626175 | 10 | No | 1 |
| 3 | 10 | Sqrt ((3 – 2.5)2 + (10 - 7)2) =  3.0413812651491097 | 8 | No | 0 |
| 5.6 | 4 | Sqrt ((5.6 – 2.5)2 + (4 - 7)2) =  4.3139309220245980 | 11 | No | 1 |
| 4 | 2 | Sqrt ((4 – 2.5)2 + (2 - 7)2) =  5.2201532544552753 | 15 | No | 1 |
| 3.5 | 8 | Sqrt ((3.5 – 2.5)2 + (8 - 7)2) =  1.4142135623730951 | 2 | Yes | 0 |
| 2 | 11 | Sqrt ((2 – 2.5)2 + (11 - 7)2) =  4.0311288741492746 | 9 | No | 0 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 5 | No | 1 |
| 2 | 9 | Sqrt ((2 – 2.5)2 + (9 - 7)2) =  2.0615528128088303 | 4 | No | 0 |
| 1 | 7 | Sqrt ((1 – 2.5)2 + (7 - 7)2) =  1.5000000000000000 | 3 | Yes | 0 |

1. **Use simple majority of the category of nearest neighbors as the prediction value of the query instance:**

We have three 0 val category. So, then we conclude that the value classified to unknown point is 0 Val category.

**Python program**

What I Observed in the Given Python Program are explained in detail below with all the 5 stages of KNN Algorithm implementation.

**DATA:**

|  |  |  |
| --- | --- | --- |
| **X** | **Y** | **Val** |
| 1 | 12 | 0 |
| 2 | 5 | 0 |
| 3 | 6 | 0 |
| 3 | 10 | 0 |
| 3.5 | 8 | 0 |
| 2 | 11 | 0 |
| 2 | 9 | 0 |
| 1 | 7 | 0 |
| 5 | 3 | 1 |
| 3 | 2 | 1 |
| 1.5 | 9 | 1 |
| 7 | 2 | 1 |
| 6 | 1 | 1 |
| 3.8 | 1 | 1 |
| 5.6 | 4 | 1 |
| 4 | 2 | 1 |
| 2 | 5 | 1 |

**Testing point**

**P:** X = 2.5 and Y = 7

1. **Determine parameter K = number of nearest neighbors.**

Finding ‘k’ value is not easy. Usually k is chosen to be odd number if the number of classes is 2. Another simplest approach to select k is finding the sqrt (total number of items). Smaller k gives more noisy data and larger k value makes it take more time to compute.

This k value is a value that considers the nearest distances form the point to the other training points. which is used to predict the class that new point should be in. In both cases the k is doing same work.

This is what the K = 3 is in the program. Since, there are two classes 0 and 1.

1. **Calculate the distance between the query-instance and all the training samples.**

Which is sqrt((training point 1st element – new point 1st element)^2 + (training point 2nd element – new point 2nd element)^2)

|  |  |  |
| --- | --- | --- |
| **X** | **Y** | **Distance** |
| 1 | 12 | Sqrt ((1 – 2.5)2 + (12 - 7)2) =  5.220153254455275 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 |
| 3 | 6 | Sqrt ((3 – 2.5)2 + (6 - 7)2) =  1.118033988749895 |
| 3 | 10 | Sqrt ((3 – 2.5)2 + (10 - 7)2) =  3.0413812651491097 |
| 3.5 | 8 | Sqrt ((3.5 – 2.5)2 + (8 - 7)2) =  1.4142135623730951 |
| 2 | 11 | Sqrt ((2 – 2.5)2 + (11 - 7)2) =  4.031128874149275 |
| 2 | 9 | Sqrt ((2 – 2.5)2 + (9 - 7)2) =  2.0615528128088303 |
| 1 | 7 | Sqrt ((1 – 2.5)2 + (7 - 7)2) =  1.5 |
| 5 | 3 | Sqrt ((5 – 2.5)2 + (3 - 7)2) =  4.716990566028302 |
| 3 | 2 | Sqrt ((3 – 2.5)2 + (2 - 7)2) =  5.024937810560445 |
| 1.5 | 9 | Sqrt ((1.5 – 2.5)2 + (9 - 7)2) =  2.23606797749979 |
| 7 | 2 | Sqrt ((7 – 2.5)2 + (2 - 7)2) =  6.726812023536855 |
| 6 | 1 | Sqrt ((6 – 2.5)2 + (1 - 7)2) =  6.946221994724902 |
| 3.8 | 1 | Sqrt ((3.8 – 2.5)2 + (1 - 7)2) =  6.139218191268331 |
| 5.6 | 4 | Sqrt ((5.6 – 2.5)2 + (4 - 7)2) =  4.313930922024598 |
| 4 | 2 | Sqrt ((4 – 2.5)2 + (2 - 7)2) =  5.220153254455275 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 |

1. **Sort the distance and determine nearest neighbors based on the K-th minimum distance**

Sorting the distances in ascending order

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **Distance** | **Rank Minimum Distance** | **Does it include in 3- nearest neighbors?** |
| 1 | 12 | Sqrt ((1 – 2.5)2 + (12 - 7)2) =  5.220153254455275 | 13 | No |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 4 | No |
| 3 | 6 | Sqrt ((3 – 2.5)2 + (6 - 7)2) =  1.118033988749895 | 1 | Yes |
| 3 | 10 | Sqrt ((3 – 2.5)2 + (10 - 7)2) =  3.0413812651491097 | 8 | No |
| 3.5 | 8 | Sqrt ((3.5 – 2.5)2 + (8 - 7)2) =  1.4142135623730951 | 2 | Yes |
| 2 | 11 | Sqrt ((2 – 2.5)2 + (11 - 7)2) =  4.031128874149275 | 9 | No |
| 2 | 9 | Sqrt ((2 – 2.5)2 + (9 - 7)2) =  2.0615528128088303 | 5 | No |
| 1 | 7 | Sqrt ((1 – 2.5)2 + (7 - 7)2) =  1.5 | 3 | Yes |
| 5 | 3 | Sqrt ((5 – 2.5)2 + (3 - 7)2) =  4.716990566028302 | 11 | No |
| 3 | 2 | Sqrt ((3 – 2.5)2 + (2 - 7)2) =  5.024937810560445 | 12 | No |
| 1.5 | 9 | Sqrt ((1.5 – 2.5)2 + (9 - 7)2) =  2.23606797749979 | 7 | No |
| 7 | 2 | Sqrt ((7 – 2.5)2 + (2 - 7)2) =  6.726812023536855 | 16 | No |
| 6 | 1 | Sqrt ((6 – 2.5)2 + (1 - 7)2) =  6.946221994724902 | 17 | No |
| 3.8 | 1 | Sqrt ((3.8 – 2.5)2 + (1 - 7)2) =  6.139218191268331 | 15 | No |
| 5.6 | 4 | Sqrt ((5.6 – 2.5)2 + (4 - 7)2) =  4.313930922024598 | 10 | No |
| 4 | 2 | Sqrt ((4 – 2.5)2 + (2 - 7)2) =  5.220153254455275 | 14 | No |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 6 | No |

1. **Gather the category Val of the nearest neighbors**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Y** | **Distance** | **Rank Minimum Distance** | **Does it include in 3- nearest neighbors?** | **Category of nearest neighbor**  **Val** |
| 1 | 12 | Sqrt ((1 – 2.5)2 + (12 - 7)2) =  5.220153254455275 | 13 | No | 0 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 4 | No | 0 |
| 3 | 6 | Sqrt ((3 – 2.5)2 + (6 - 7)2) =  1.118033988749895 | 1 | Yes | 0 |
| 3 | 10 | Sqrt ((3 – 2.5)2 + (10 - 7)2) =  3.0413812651491097 | 8 | No | 0 |
| 3.5 | 8 | Sqrt ((3.5 – 2.5)2 + (8 - 7)2) =  1.4142135623730951 | 2 | Yes | 0 |
| 2 | 11 | Sqrt ((2 – 2.5)2 + (11 - 7)2) =  4.031128874149275 | 9 | No | 0 |
| 2 | 9 | Sqrt ((2 – 2.5)2 + (9 - 7)2) =  2.0615528128088303 | 5 | No | 0 |
| 1 | 7 | Sqrt ((1 – 2.5)2 + (7 - 7)2) =  1.5 | 3 | Yes | 0 |
| 5 | 3 | Sqrt ((5 – 2.5)2 + (3 - 7)2) =  4.716990566028302 | 11 | No | 1 |
| 3 | 2 | Sqrt ((3 – 2.5)2 + (2 - 7)2) =  5.024937810560445 | 12 | No | 1 |
| 1.5 | 9 | Sqrt ((1.5 – 2.5)2 + (9 - 7)2) =  2.23606797749979 | 7 | No | 1 |
| 7 | 2 | Sqrt ((7 – 2.5)2 + (2 - 7)2) =  6.726812023536855 | 16 | No | 1 |
| 6 | 1 | Sqrt ((6 – 2.5)2 + (1 - 7)2) =  6.946221994724902 | 17 | No | 1 |
| 3.8 | 1 | Sqrt ((3.8 – 2.5)2 + (1 - 7)2) =  6.139218191268331 | 15 | No | 1 |
| 5.6 | 4 | Sqrt ((5.6 – 2.5)2 + (4 - 7)2) =  4.313930922024598 | 10 | No | 1 |
| 4 | 2 | Sqrt ((4 – 2.5)2 + (2 - 7)2) =  5.220153254455275 | 14 | No | 1 |
| 2 | 5 | Sqrt ((2 – 2.5)2 + (5 - 7)2) =  2.0615528128088303 | 6 | No | 1 |

1. **Use simple majority of the category of nearest neighbors as the prediction value of the query instance:**

We have three 0 val category. So, then we conclude that the value classified to unknown point is 0 Val category.

**Building an example of KNN Algorithm:**

There is a survey which is conducted on the face cream’s efficiency testing. To do that there is a consideration of four attributes which helps us to define the efficiency of the face cream. The attributes are absorbing capacity, time that it lasts on skin, sun protection, and Antioxidants and vitamin c, e. Here are some training samples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X1 = Absorbing capacity (Seconds) | X2 = Time it Lasts on skin (Hours) | X3 = Sun Protection (in SPF) | X4 = Antioxidant and Vitamin C, E (%) | Y = Classification |
| 5 | 8 | 100 | 10 | A |
| 7 | 6 | 50 | 7 | A |
| 10 | 4 | 15 | 5 | B |
| 8 | 6 | 30 | 5 | B |
| 6 | 7 | 50 | 8 | A |
| 5 | 6 | 50 | 12 | A |
| 3 | 4 | 15 | 5 | B |
| 5 | 8 | 100 | 15 | A |

Now the factory produces a new Face Cream that pass laboratory test with X1 = 4, X2 = 9, X3 = 50 and X4 = 6.

1. **Determine parameter K = number of nearest neighbors.**

K = 3

1. **Calculate the distance between the query- instance and all the training samples:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X1 = Absorbing capacity (Seconds) | X2 = Time it Lasts on skin (Hours) | X3 = Sun Protection (in SPF) | X4 = Antioxidant and Vitamin C, E (%) | Square Distance to the query instance (4, 9, 50, 6) |
| 5 | 8 | 100 | 10 | (5-4)2 + (8-9)2 + (100-50)2 + (10-6)2 = 50.18 |
| 7 | 6 | 50 | 7 | (7-4)2 + (6-9)2 + (50-50)2 + (7-6)2 = 4.36 |
| 10 | 4 | 15 | 5 | (10-4)2 + (4-9)2 + (15-50)2 + (5-6)2 = 35.87 |
| 8 | 6 | 30 | 5 | (8-4)2 + (6-9)2 + (30-50)2 + (5-6)2 = 20.64 |
| 6 | 7 | 50 | 8 | (6-4)2 + (7-9)2 + (50-50)2 + (8-6)2 = 3.46 |
| 5 | 6 | 50 | 12 | (5-4)2 + (6-9)2 + (50-50)2 + (12-6)2 = 6.78 |
| 3 | 4 | 15 | 5 | (3-4)2 + (4-9)2 + (15-50)2 + (5-6)2 = 35.38 |
| 5 | 8 | 100 | 15 | (5-4)2 + (8-9)2 + (100-50)2 + (15-6)2 = 50.823 |

1. **Sort the distance and determine nearest neighbors based on the k- th minimum distance:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X1 = Absorbing capacity (Seconds) | X2 = Time it Lasts on skin (Hours) | X3 = Sun Protection (in SPF) | X4 = Antioxidant and Vitamin C, E (%) | Square Distance to the query instance  (4, 9, 50, 6) | Rank minimum distance | Does it include in 3- nearest neighbors? |
| 5 | 8 | 100 | 10 | (5-4)2 + (8-9)2 + (100-50)2 + (10-6)2 = 50.18 | 7 | No |
| 7 | 6 | 50 | 7 | (7-4)2 + (6-9)2 + (50-50)2 + (7-6)2 = 4.36 | 2 | Yes |
| 10 | 4 | 15 | 5 | (10-4)2 + (4-9)2 + (15-50)2 + (5-6)2 = 35.87 | 6 | No |
| 8 | 6 | 30 | 5 | (8-4)2 + (6-9)2 + (30-50)2 + (5-6)2 = 20.64 | 4 | No |
| 6 | 7 | 50 | 8 | (6-4)2 + (7-9)2 + (50-50)2 + (8-6)2 = 3.46 | 1 | Yes |
| 5 | 6 | 50 | 12 | (5-4)2 + (6-9)2 + (50-50)2 + (12-6)2 = 6.78 | 3 | Yes |
| 3 | 4 | 15 | 5 | (3-4)2 + (4-9)2 + (15-50)2 + (5-6)2 = 35.38 | 5 | No |
| 5 | 8 | 100 | 15 | (5-4)2 + (8-9)2 + (100-50)2 + (15-6)2 = 50.82 | 8 | No |

1. **Gather the category Y of the nearest neighbors.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X1 = Absorbing capacity (Seconds) | X2 = Time it Lasts on skin (Hours) | X3 = Sun Protection (in SPF) | X4 = Antioxidant and Vitamin C, E (%) | Square Distance to the query instance (6, 7) | Rank minimum distance | Does it include in 3- nearest neighbors? | Y = Classification |
| 5 | 8 | 100 | 10 | (5-4)2 + (8-9)2 + (100-50)2 + (10-6)2 = 50.18 | 7 | No | A |
| 7 | 6 | 50 | 7 | (7-4)2 + (6-9)2 + (50-50)2 + (7-6)2 = 4.36 | 2 | Yes | A |
| 10 | 4 | 15 | 5 | (10-4)2 + (4-9)2 + (15-50)2 + (5-6)2 = 35.87 | 6 | No | B |
| 8 | 6 | 30 | 5 | (8-4)2 + (6-9)2 + (30-50)2 + (5-6)2 = 20.64 | 4 | No | B |
| 6 | 7 | 50 | 8 | (6-4)2 + (7-9)2 + (50-50)2 + (8-6)2 = 3.46 | 1 | Yes | A |
| 5 | 6 | 50 | 12 | (5-4)2 + (6-9)2 + (50-50)2 + (12-6)2 = 6.78 | 3 | Yes | A |
| 3 | 4 | 15 | 5 | (3-4)2 + (4-9)2 + (15-50)2 + (5-6)2 = 35.38 | 5 | No | B |
| 5 | 8 | 100 | 15 | (5-4)2 + (8-9)2 + (100-50)2 + (15-6)2 = 50.82 | 8 | No | A |

1. **Use simple majority of the category of nearest neighbors as the prediction value of the query instance:**

We have 3 A Classifications. So, then we conclude that a new face cream that pass laboratory test with X1 = 4, X2 = 9, X3 = 50 and X4 = 6 is included in **A** category.