K-Nearest Neighbor(KNN) Algorithm

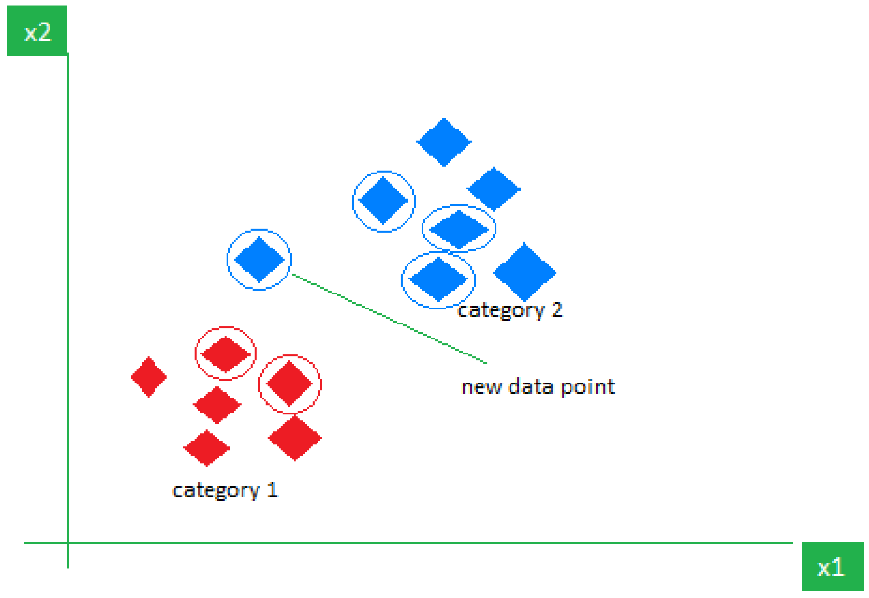
The K-Nearest Neighbors (KNN) algorithm is a robust and intuitive machine learning method employed to tackle classification and regression problems. By capitalizing on the concept of similarity, KNN predicts the label or value of a new data point by considering its K closest neighbours in the training dataset. In this article, we will learn about a supervised learning algorithm (KNN) or the k – Nearest Neighbours, highlighting it's user-friendly nature.

What is the K-Nearest Neighbors Algorithm?

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:



KNN Algorithm working visualization

Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set. Note that the unclassified points are marked as 'White'.

Intuition Behind KNN Algorithm

If we plot these points on a graph, we may be able to locate some clusters or groups. Now, given an unclassified point, we can assign it to a group by close to a cluster of points classified as 'Red' has a higher probability of getting classified as 'Red'.

Intuitively, we can see that the first point (2.5, 7) should be classified as 'Green' and the second point (5.5, 4.5) should be classified as 'Red'.

Why do we need a KNN algorithm?

(K-NN) algorithm is a versatile and widely used machine learning algorithm that is primarily used for its simplicity and ease of implementation. It does not require any assumptions about the underlying data distribution. It can also handle both numerical and categorical data, making it a flexible choice for various types of datasets in classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of data points in a given dataset. K-NN is less sensitive to outliers compared to other algorithms.

The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d}(x_j - X_{i_j})^2}]$$

Manhattan Distance

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

$$d\left(x,y\right)=\sum_{i=1}^{n}\left|x_i-y_i\right|$$

Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

$$d\left(x,y\right)=\left(\sum_{i=1}^{n}\left(x_i-y_i\right)^p\right)^{\frac{1}{p}}$$

From the formula above we can say that when p = 2 then it is the same as the formula for the Euclidean distance and when p = 1 then we obtain the formula for the Manhattan distance.

The above-discussed metrics are most common while dealing with a Machine Learning problem but there are other distance metrics as well like Hamming Distance which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be boolean as well as string values.

How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. Cross-validation methods can help in selecting the best k value for the given dataset.

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.

To make predictions, the algorithm calculates the distance between each new data point in the test dataset and all the data points in the training dataset. The Euclidean distance is a commonly used distance metric in K-NN, but other distance metrics, such as Manhattan distance or Minkowski distance, can also be used depending on the problem and data. Once the distances between the new data point and all the data points in the training dataset are calculated, the algorithm proceeds to find the K nearest neighbors based on these distances. The specific method for selecting the nearest neighbors can vary, but a common approach is to sort the distances in ascending order and choose the K data points with the shortest distances.

After identifying the K nearest neighbors, the algorithm makes predictions based on the labels or values associated with these neighbors. For classification tasks, the majority class among the K neighbors is assigned as the predicted label for the new data point. For regression tasks, the average or weighted average of the values of the K neighbors is assigned as the predicted value.

Let X be the training dataset with n data points, where each data point is represented by a d-dimensional feature vector X_i and Y be the corresponding labels or values for each data point in X.Given a new data point x, the algorithm calculates the distance between x and each data point X_i in X using a distance metric, such as Euclidean distance:

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i_j})^2}]$$

The algorithm selects the K data points from X that have the shortest distances to x. For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x. For regression tasks, the algorithm calculates the average or weighted average of the values y of the K

Applications of the KNN Algorithm

- **Data Preprocessing** – While dealing with any Machine Learning problem we first perform the EDA part in which if we find that the data contains missing values then there are multiple imputation methods are available as well. One of such method is KNN Imputer which is quite effective ad generally used for sophisticated imputation methodologies.
- **Pattern Recognition** – KNN algorithms work very well if you have trained a KNN algorithm using the MNIST dataset and then performed the evaluation process then you must have come across the fact that the accuracy is too high.
- **Recommendation Engines** – The main task which is performed by a KNN algorithm is to assign a new query point to a pre-existed group that has been created using a huge corpus of datasets. This is exactly what is required in the recommender systems to assign each user to a particular group and then provide them recommendations based on that group's preferences.

Advantages of the KNN Algorithm

- **Easy to implement** as the complexity of the algorithm is not that high.
- **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

Disadvantages of the KNN Algorithm

- **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
- **Curse of Dimensionality** – There is a term known as the peaking

dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.

- **Prone to Overfitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

Example Program:

Assume 0 and 1 as the two classifiers (groups).

C++

```
// C++ program to find groups of unknown
// Points using K nearest neighbour algorithm.
#include <bits/stdc++.h>
using namespace std;

struct Point
{
    int val;     // Group of point
    double x, y;     // Co-ordinate of point
    double distance; // Distance from test point
};

// Used to sort an array of points by increasing
// order of distance
bool comparison(Point a, Point b)
{
    return (a.distance < b.distance);
}

// This function finds classification of point p using
// k nearest neighbour algorithm. It assumes only two
// groups and returns 0 if p belongs to group 0, else
// 1 (belongs to group 1).
int classifyAPoint(Point arr[], int n, int k, Point p)
{
    // Fill distances of all points from p
    for (int i = 0; i < n; i++)
        arr[i].distance =
            sqrt((arr[i].x - p.x) * (arr[i].x - p.x) +
                 (arr[i].y - p.y) * (arr[i].y - p.y));
    sort(arr, arr+n, comparison);

    // Now consider the first k elements and only
    // two groups
    int freq1 = 0;     // Frequency of group 0
    int freq2 = 0;     // Frequency of group 1
    for (int i = 0; i < k; i++)
    {
        if (arr[i].val == 0)
            freq1++;
        else if (arr[i].val == 1)
            freq2++;
    }

    return (freq1 > freq2 ? 0 : 1);
}

// Driver code
int main()
{
    int n = 17; // Number of data points
    Point arr[n];

    arr[0].x = 1;
    arr[0].y = 12;
    arr[0].val = 0;

    arr[1].x = 2;
    arr[1].y = 5;
    arr[1].val = 0;

    arr[2].x = 5;
    arr[2].y = 3;
    arr[2].val = 1;

    arr[3].x = 3;
    arr[3].y = 2;
    arr[3].val = 1;

    arr[4].x = 3;
    arr[4].y = 6;
    arr[4].val = 0;

    arr[5].x = 1.5;
    arr[5].y = 9;
    arr[5].val = 1;
    arr[6].val = 1;

    arr[7].x = 6;
    arr[7].y = 1;
    arr[7].val = 1;

    arr[8].x = 3.8;
    arr[8].y = 3;
    arr[8].val = 1;

    arr[9].x = 3;
    arr[9].y = 10;
    arr[9].val = 0;

    arr[10].x = 5.6;
    arr[10].y = 4;
    arr[10].val = 1;

    arr[11].x = 4;
    arr[11].y = 2;
    arr[11].val = 1;

    arr[12].x = 3.5;
    arr[12].y = 8;
    arr[12].val = 0;

    arr[13].x = 2;
    arr[13].y = 11;
    arr[13].val = 0;

    arr[14].x = 2;
    arr[14].y = 5;
    arr[14].val = 1;

    arr[15].x = 2;
    arr[15].y = 9;
    arr[15].val = 0;

    arr[16].x = 1;
    arr[16].y = 7;
    arr[16].val = 0;

    /*Testing Point*/
    Point p;
    p.x = 2.5;
    p.y = 7;
```

```
    printf ("The value classified to unknown point"
            " is %d.\n", classifyAPoint(arr, n, k, p));
    return 0;
}
```

Java

```
// Java program to find groups of unknown
// Points using K nearest neighbour algorithm.
import java.io.*;
import java.util.*;

class GFG {
    static class Point {
        int val; // Group of point
        double x, y; // Co-ordinate of point
        double distance; // Distance from test point
    }

    // Used to sort an array of points by increasing
    // order of distance
    static class comparison implements Comparator<Point> {

        public int compare(Point a, Point b)
        {
            if (a.distance < b.distance)
                return -1;
            else if (a.distance > b.distance)
                return 1;
            return 0;
        }
    }

    // This function finds classification of point p using
    // k nearest neighbour algorithm. It assumes only two
    // groups and returns 0 if p belongs to group 0, else
    // 1 (belongs to group 1).
    static int classifyAPoint(Point arr[], int n, int k,
                              Point p)
    {
        // Fill distances of all points from p
        for (int i = 0; i < n; i++)
            arr[i].distance = Math.sqrt(
                (arr[i].x - p.x) * (arr[i].x - p.x)
                + (arr[i].y - p.y) * (arr[i].y - p.y));
        // Now consider the first k elements and only
        // two groups
        int freq1 = 0; // Frequency of group 0
        int freq2 = 0; // Frequency of group 1
        for (int i = 0; i < k; i++) {
            if (arr[i].val == 0)
                freq1++;
            else if (arr[i].val == 1)
                freq2++;
        }

        return (freq1 > freq2 ? 0 : 1);
    }

    // Driver code
    public static void main(String[] args)
    {
        int n = 17; // Number of data points
        Point[] arr = new Point[n];
        for (int i = 0; i < 17; i++) {
            arr[i] = new Point();
        }
        arr[0].x = 1;
        arr[0].y = 12;
        arr[0].val = 0;

        arr[1].x = 2;
        arr[1].y = 5;
        arr[1].val = 0;

        arr[2].x = 5;
        arr[2].y = 3;
        arr[2].val = 1;

        arr[3].x = 3;
        arr[3].y = 2;
        arr[3].val = 1;

        arr[4].x = 3;
        arr[4].y = 6;
        arr[4].val = 0;

        arr[5].x = 1.5;
        arr[5].y = 9;
        arr[5].val = 1;
        arr[6].val = 1;

        arr[7].x = 6;
        arr[7].y = 1;
        arr[7].val = 1;

        arr[8].x = 3.8;
        arr[8].y = 3;
        arr[8].val = 1;

        arr[9].x = 3;
        arr[9].y = 10;
        arr[9].val = 0;

        arr[10].x = 5.6;
        arr[10].y = 4;
        arr[10].val = 1;

        arr[11].x = 4;
        arr[11].y = 2;
        arr[11].val = 1;

        arr[12].x = 3.5;
        arr[12].y = 8;
        arr[12].val = 0;

        arr[13].x = 2;
        arr[13].y = 11;
        arr[13].val = 0;

        arr[14].x = 2;
        arr[14].y = 5;
        arr[14].val = 1;

        arr[15].x = 2;
        arr[15].y = 9;
        arr[15].val = 0;

        arr[16].x = 1;
        arr[16].y = 7;
        arr[16].val = 0;

        /*Testing Point*/
        Point p = new Point();
        p.x = 2.5;
        p.y = 7;
        System.out.println(
            "The value classified to unknown point is "
            + classifyAPoint(arr, n, k, p));
    }
}

// This code is contributed by Karandeep1234
```

Python3

```
import math

def classifyAPoint(points,p,k=3):
    '''
     This function finds the classification of p using
     k nearest neighbor algorithm. It assumes only two
     groups and returns 0 if p belongs to group 0, else
      1 (belongs to group 1).

      Parameters -
          points: Dictionary of training points having two keys - 0 and 1
                   Each key have a list of training data points belong to that

          p : A tuple, test data point of the form (x,y)

          k : number of nearest neighbour to consider, default is 3
    '''

    distance=[]
    for group in points:
        for feature in points[group]:

            #calculate the euclidean distance of p from training points
            euclidean_distance = math.sqrt((feature[0]-p[0])**2 +(feature[1]-p[1]

            # Add a tuple of form (distance,group) in the distance list
            distance.append((euclidean_distance,group))

    # sort the distance list in ascending order
    # and select first k distances
    distance = sorted(distance)[:k]

    freq1 = 0 #frequency of group 0
    freq2 = 0 #frequency og group 1
```

```
        elif d[1] == 1:
            freq2 += 1

    return 0 if freq1>freq2 else 1

# driver function
def main():

    # Dictionary of training points having two keys - 0 and 1
    # key 0 have points belong to class 0
    # key 1 have points belong to class 1

    points = {0:[(1,12),(2,5),(3,6),(3,10),(3.5,8),(2,11),(2,9),(1,7)],
              1:[(5,3),(3,2),(1.5,9),(7,2),(6,1),(3.8,1),(5.6,4),(4,2),(2,5)]}

    # testing point p(x,y)
    p = (2.5,7)

    # Number of neighbours
    k = 3

    print("The value classified to unknown point is: {}".\
          format(classifyAPoint(points,p,k)))

if __name__ == '__main__':
    main()
```

C#

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;

// C# program to find groups of unknown
// Points using K nearest neighbour algorithm.
class Point {
  public int val; // Group of point
  public double x, y; // Co-ordinate of point
  public int distance; // Distance from test point
}

class HelloWorld {
  // groups and returns 0 if p belongs to group 0, else
  // 1 (belongs to group 1).
  public static int classifyAPoint(List<Point> arr, int n, int k, Point p)
  {
    // Fill distances of all points from p
    for (int i = 0; i < n; i++)
      arr[i].distance = (int)Math.Sqrt((arr[i].x - p.x) * (arr[i].x - p.x) + (arr[

    // Sort the Points by distance from p
    arr.Sort(delegate(Point x, Point y) {
      return x.distance.CompareTo(y.distance);
    });

    // Now consider the first k elements and only
    // two groups
    int freq1 = 0; // Frequency of group 0
    int freq2 = 0; // Frequency of group 1
    for (int i = 0; i < k; i++) {
      if (arr[i].val == 0)
        freq1++;
      else if (arr[i].val == 1)
        freq2++;
    }

    return (freq1 > freq2 ? 0 : 1);
  }

  static void Main() {
    int n = 17; // Number of data points
    List<Point> arr = new List<Point>();
    for(int i = 0; i < n; i++){
      arr.Add(new Point());
    }

    arr[0].x = 1;
    arr[0].y = 12;
    arr[0].val = 0;

    arr[1].x = 2;
    arr[1].y = 5;
    arr[1].val = 0;

    arr[2].x = 5;
    arr[2].y = 3;
    arr[2].val = 1;

    arr[3].x = 3;
    arr[4].x = 3;
    arr[4].y = 6;
    arr[4].val = 0;

    arr[5].x = 1.5;
    arr[5].y = 9;
    arr[5].val = 1;

    arr[6].x = 7;
    arr[6].y = 2;
    arr[6].val = 1;

    arr[7].x = 6;
    arr[7].y = 1;
    arr[7].val = 1;

    arr[8].x = 3.8;
    arr[8].y = 3;
    arr[8].val = 1;

    arr[9].x = 3;
    arr[9].y = 10;
    arr[9].val = 0;

    arr[10].x = 5.6;
    arr[10].y = 4;
    arr[10].val = 1;

    arr[11].x = 4;
    arr[11].y = 2;
    arr[11].val = 1;

    arr[12].x = 3.5;
    arr[12].y = 8;
    arr[12].val = 0;

    arr[13].x = 2;
    arr[13].y = 11;
    arr[13].val = 0;

    arr[14].x = 2;
    arr[14].y = 5;
    arr[14].val = 1;

    arr[15].x = 2;
    arr[15].y = 9;
```

```
        arr[16].x = 1;
        arr[16].y = 7;
        arr[16].val = 0;

        /*Testing Point*/
        Point p = new Point();
        p.x = 2.5;
        p.y = 7;

        // Parameter to decide group of the testing point
        int k = 3;
        Console.WriteLine("The value classified to unknown point is " + classifyAPoint
      }
    }

    // The code is contributed by Nidhi goel.
```

Javascript

```
class Point {
  constructor(val, x, y, distance) {
    this.val = val; // Group of point
    this.x = x; // X-coordinate of point
    this.y = y; // Y-coordinate of point
    this.distance = distance; // Distance from test point
  }
}

// Used to sort an array of points by increasing order of distance
class Comparison {
  compare(a, b) {
    if (a.distance < b.distance) {
      return -1;
    } else if (a.distance > b.distance) {
      return 1;
    }
    return 0;
  }
}

// This function finds classification of point p using
// k nearest neighbour algorithm. It assumes only two
// groups and returns 0 if p belongs to group 0, else
// 1 (belongs to group 1).
```

```
      arr[i].distance = Math.sqrt((arr[i].x - p.x) * (arr[i].x - p.x) + (arr[i].y -
    }

    // Sort the Points by distance from p
    arr.sort(new Comparison());

    // Now consider the first k elements and only two groups
    let freq1 = 0; // Frequency of group 0
    let freq2 = 0; // Frequency of group 1
    for (let i = 0; i < k; i++) {
      if (arr[i].val === 0) {
        freq1++;
      } else if (arr[i].val === 1) {
        freq2++;
      }
    }

    return freq1 > freq2 ? 0 : 1;
  }

  // Driver code
  const n = 17; // Number of data points
  const arr = new Array(n);
  for (let i = 0; i < 17; i++) {
    arr[i] = new Point();
  }
  arr[0].x = 1;
  arr[0].y = 12;
  arr[0].val = 0;

  arr[1].x = 2;
  arr[1].y = 5;
  arr[1].val = 0;

  arr[2].x = 5;
  arr[2].y = 3;
  arr[2].val = 1;

  arr[3].x = 3;
  arr[3].y = 2;
  arr[3].val = 1;

  arr[4].x = 3;
  arr[4].y = 6;
  arr[4].val = 0;

  arr[5].x = 1.5;
```

```
        arr[6].x = 7;
        arr[6].y = 2;
        arr[6].val = 1;

        arr[7].x = 6;
        arr[7].y = 1;
        arr[7].val = 1;

        arr[8].x = 3.8;
        arr[8].y = 3;
        arr[8].val = 1;

        arr[9].x = 3;
        arr[9].y = 10;
        arr[9].val = 0;

        arr[10].x = 5.6;
        arr[10].y = 4;
        arr[10].val = 1;

        arr[11].x = 4
        arr[11].y = 2;
        arr[11].val = 1;

        arr[12].x = 3.5;
        arr[12].y = 8;
        arr[12].val = 0;

        arr[13].x = 2;
        arr[13].y = 11;
        arr[13].val = 0;

        arr[14].x = 2;
        arr[14].y = 5;
        arr[14].val = 1;

        arr[15].x = 2;
        arr[15].y = 9;
        arr[15].val = 0;

        arr[16].x = 1;
        arr[16].y = 7;
        arr[16].val = 0;

        // Testing Point
        let p = {
```

```
    val: -1, // uninitialized
  };

  // Parameter to decide group of the testing point
  let k = 3;

  console.log(
    "The value classified to unknown point is " +
      classifyAPoint(arr, n, k, p)
  );

  function classifyAPoint(arr, n, k, p) {
    // Fill distances of all points from p
    for (let i = 0; i < n; i++) {
      arr[i].distance = Math.sqrt(
        (arr[i].x - p.x) * (arr[i].x - p.x) + (arr[i].y - p.y) * (arr[i].y - p.y)
      );
    }

    // Sort the Points by distance from p
    arr.sort(function (a, b) {
      if (a.distance < b.distance) return -1;
      else if (a.distance > b.distance) return 1;
      return 0;
    });

    // Now consider the first k elements and only two groups
    let freq1 = 0; // Frequency of group 0
    let freq2 = 0; // Frequency of group 1
    for (let i = 0; i < k; i++) {
      if (arr[i].val == 0) freq1++;
      else if (arr[i].val == 1) freq2++;
    }

    return freq1 > freq2 ? 0 : 1;
  }
```

Output:

```
The value classified as an unknown point is 0.
```

Time Complexity: O(N * logN)

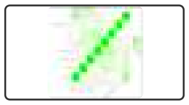
Auxiliary Space: O(1)

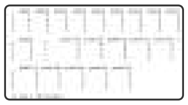
Last Updated : 09 Nov, 2023

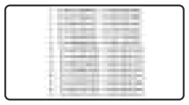
39

Similar Reads

k-nearest neighbor algorithm in Python

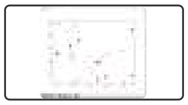

How To Predict Diabetes using K-Nearest Neighbor in R

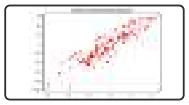

ML | T-distributed Stochastic Neighbor Embedding (t-SNE) Algorithm

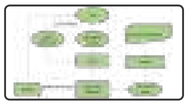

ML | Implementation of KNN classifier using Sklearn

ML | Kaggle Breast Cancer Wisconsin Diagnosis using KNN and Cross Validation

IBM HR Analytics Employee Attrition & Performance using KNN

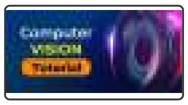

Implementation of KNN using OpenCV


KNN Model Complexity

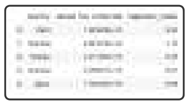

Introductory guide to Information Retrieval using KNN and KDTree

Implementation of K Nearest Neighbors

Related Tutorials



Computer Vision Tutorial


Pandas AI: The Generative AI Python Library

Top Computer Vision Projects (2023)

Deep Learning Tutorial

Top 100+ Machine Learning Projects for 2023 [with

Previous

Implementation of Ridge Regression from Scratch using Python

Next

Implementation of Elastic Net Regression From Scratch

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy Normal Medium Hard Expert

Improved By : NIRBHAYKUMAR1, snk_pan, gurukiranx, simmytarika5, pankajsharmagfg, vinayedula, prachisoda1234, amartyaghoshgfg, karandeep1234, classroompxico, abhishekm482g, abhasonkar01, harleenkaugw0j

Article Tags : Directi , Machine Learning , Machine Learning

Practice Tags : Directi, Machine Learning, Machine Learning

Improve Article Report Issue

Basic DSA Coding Problems

DSA Roadmap by Sandeep Jain

DSA with JavaScript

Top 100 DSA Interview Problems

All Cheat Sheets

CSS

JavaScript

Bootstrap

ReactJS

AngularJS

NodeJS

Express.js

Lodash

Computer Science

GATE CS Notes

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Python

Python Programming Examples

Django Tutorial

Python Projects

Python Tkinter

OpenCV Python Tutorial

Python Interview Question

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Deep Learning Tutorial

DevOps

Git

AWS

Docker

Kubernetes

Azure

GCP

Competitive Programming

Top DSA for CP

Top 50 Tree Problems

Top 50 Graph Problems

Top 50 Array Problems

System Design

What is System Design

Monolithic and Distributed SD

Scalability in SD

Databases in SD

Top 15 Websites for CP

Crack System Design Round

System Design Interview Questions

Interview Corner

Company Wise Preparation

Preparation for SDE

Experienced Interviews

Internship Interviews

Competitive Programming

Aptitude Preparation

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

Commerce

Accountancy

Business Studies

Economics

Human Resource Management (HRM)

Management

Income Tax

Finance

Statistics for Economics

UPSC

Polity Notes

Geography Notes

History Notes

Science and Technology Notes

Economics Notes

Important Topics in Ethics

UPSC Previous Year Papers

SSC/ BANKING

SSC CGL Syllabus

SBI PO Syllabus

SBI Clerk Syllabus

IBPS PO Syllabus

IBPS Clerk Syllabus

Aptitude Questions

SSC CGL Practice Papers

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Share your Experiences

Internships