|  |  |  |  |
| --- | --- | --- | --- |
| **Topic: K-Means & K-Nearest Neighbor (KNN)** | **Theory** | **Mathematics** | **Numerical** |
|  |  |  |

#### Theory questions 40. What is meant by K Nearest Neighbor algorithm?

* K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on

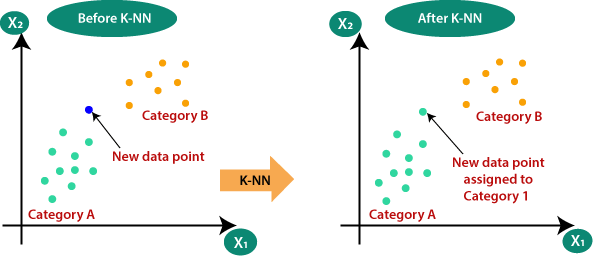
Supervised Learning technique.

* K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
* K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
* K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
* K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
* It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
* KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
* **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.



**41. Why do we need a K-NN Algorithm?**

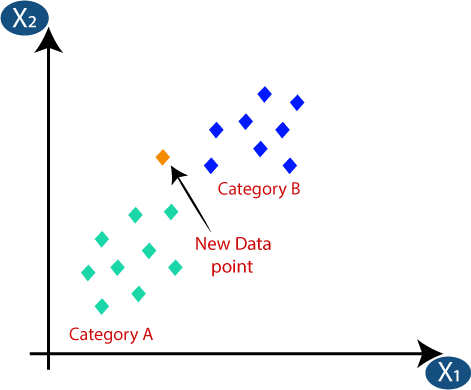
Suppose there are two categories, i.e., Category A and Category B, and we have a new data

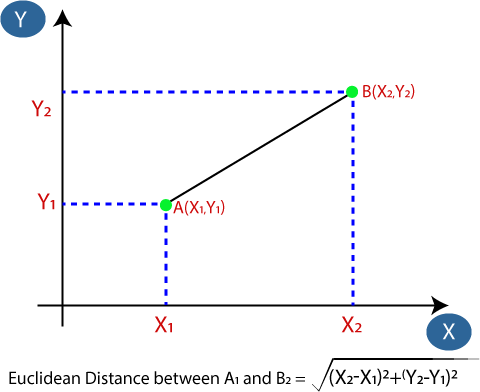
point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:

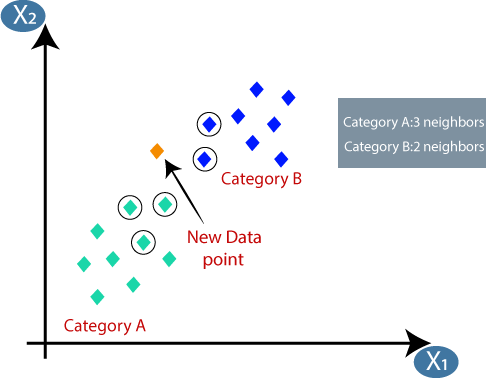
**42. How does K-NN work?**

The K-NN working can be explained on the basis of the below algorithm:

* **Step-1:** Select the number K of the neighbors
* **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
* **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
* **Step-4:** Among these k neighbors, count the number of the data points in each category.
* **Step-5:** Assign the new data points to that category for which the number of the neighbour is maximum.
* **Step-6:** Our model is ready.

Suppose we have a new data point and we need to put it in the required category. Consider the below image:

* Firstly, we will choose the number of neighbors, so we will choose the k=5.
* Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:
* By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the below image:



* As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.

**43. What is the difference between KNN and K means?**

* K-NN is a **Supervised** while K-means is an **unsupervised** Learning.
* K-NN is a **classification** or **regression** machine learning algorithm while K-means is a

**clustering** machine learning algorithm.

* K-NN is a **lazy learner** while K-Means is an **eager learner**. An eager learner has a model fitting that means a training step but a lazy learner does not have a training phase.
* K-NN performs much better if all of the data have the same scale but this is not true for K-means.
* [K-means](http://en.wikipedia.org/wiki/K_means) is a clustering algorithm that tries to partition a set of points into K sets (clusters) such that the points in each cluster tend to be near each other. It is unsupervised because the points have no external classification.
* [K-nearest neighbors](http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm) is a classification (or regression) algorithm that in order to determine the classification of a point, combines the classification of the K nearest points. It is supervised because you are trying to classify a point based on the known classification of other points.

**44. What is K-means used for?**

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering

problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k- means clustering.

K-Means Clustering is an [Unsupervised Learning algorithm,](https://www.javatpoint.com/unsupervised-machine-learning) which groups the unlabeled dataset into different clusters. Here K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.

It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

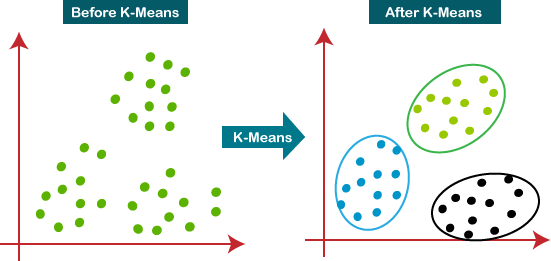
It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means [clustering](https://www.javatpoint.com/clustering-in-machine-learning) algorithm mainly performs two tasks:

* Determines the best value for K center points or centroids by an iterative process.
* Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

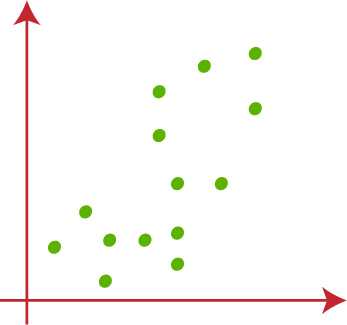
Hence each cluster has datapoints with some commonalities, and it is away from other clusters. The below diagram explains the working of the K-means Clustering Algorithm:

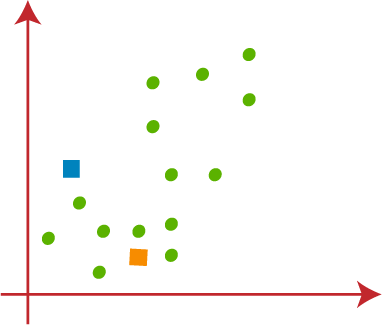


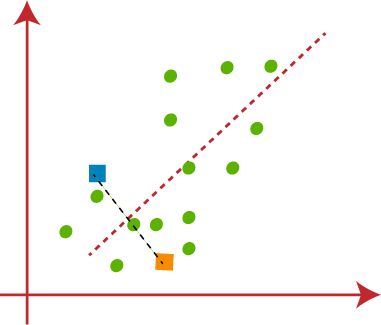
#### 45. How does K-means work?

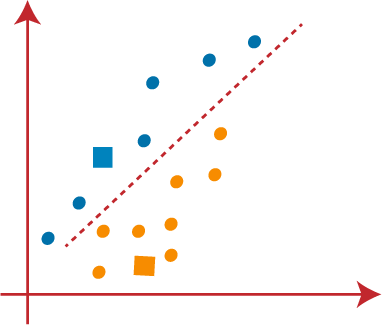
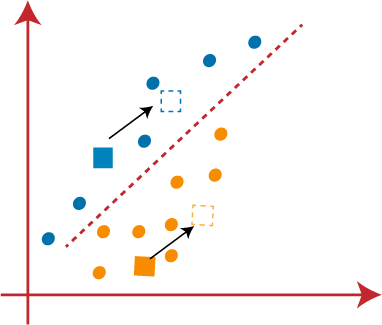
The working of the K-Means algorithm is explained in the below steps:

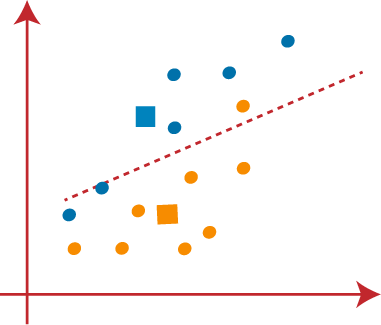
* **Step-1:** Select the number K to decide the number of clusters.
* **Step-2:** Select random K points or centroids. (It can be other from the input dataset).
* **Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.
* **Step-4:** Calculate the variance and place a new centroid of each cluster.
* **Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
* **Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.
* **Step-7**: The model is ready.

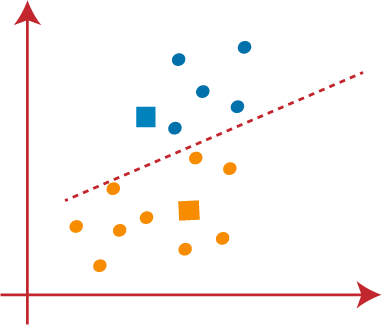
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:

* Let's take number k of clusters, K=2, to identify dataset and to put them into different clusters. It means here we will try to group these datasets into 2 different clusters.
* We need to choose some random k points or centroid to form the cluster. These points can be either the points from the dataset or any other point. So, here we are selecting the below two points as k points, which are not the part of our dataset. Consider the below image:
* Now we will assign each data point of the scatter plot to its closest K-point or centroid. We will compute it by applying some mathematics that we have studied to calculate the distance between two points. So, we will draw a median between both the centroids. Consider the below image:



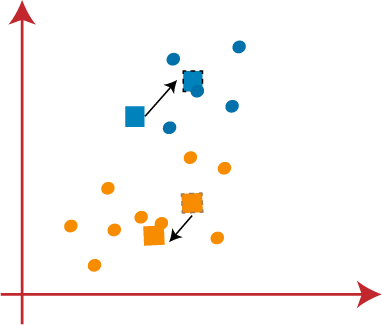
* From the above image, it is clear that points left side of the line is near to the K1 or blue centroid, and points to the right of the line are close to the yellow centroid. Let's color them as blue and yellow for clear visualization.
* As we need to find the closest cluster, so we will repeat the process by choosing **a new centroid**. To choose the new centroids, we will compute the center of gravity of these centroids, and will find new centroids as below:
* Next, we will reassign each datapoint to the new centroid. For this, we will repeat the same process of finding a median line. The median will be like below image:\

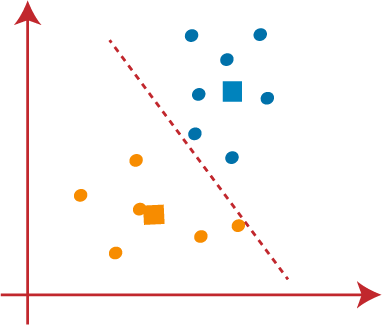


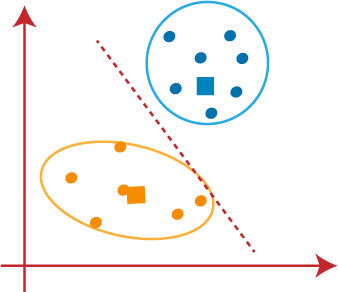
From the above image, we can see, one yellow point is on the left side of the line, and two blue points are right to the line. So, these three points will be assigned to new centroids.

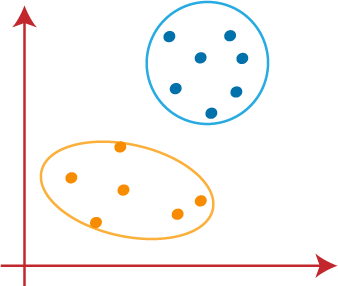
As reassignment has taken place, so we will again go to the step-4, which is finding new centroids or K-points.

* We will repeat the process by finding the center of gravity of centroids, so the new centroids will be as shown in the below image:



* As we got the new centroids so again will draw the median line and reassign the data points. So, the image will be:
* We can see in the above image; there are no dissimilar data points on either side of the line, which means our model is formed. Consider the below image:



As our model is ready, so we can now remove the assumed centroids, and the two final clusters will be as shown in the below image:

**46. Is K nearest neighbor supervised or unsupervised?**

KNN is a simple supervised learning algorithm.

KNN works on a basic assumption that data points of similar classes are closer to each other. Now suppose you have a classification problem to identify whether a given data point is of class A or class B and your aim is to classify test datapoints into these classes for which you have a training dataset of alreday classified data points.

KNN assignes 1/k probability to ‗k‘ nearest pre classified training data point from our test data point and 0 probability to rest of data points, where k may be any number (but try to put it odd to avoid tie cases). After that we count the number of each classes (i.e A and B) out of those K points and our test data point will be classified as that class whose count is greater.

For example if we are using k=5 then our algorithm looks for 5 nearest point from our test dataset point and count the number of each class out of those 5 point.Suppose our counting results in A=3 and B=2 then KNN will assign class A to that test dataset point.

KNN is categorized under supervised ML techniques. It works assuming that similar classes data points are near one another. Take a case of data points classification, where they fit? Class A or B? For this you have a classified data points training set. KNN assumes a probability of 1/k for 'k' closest data point, then assumes zero probability for other data points.

Now k could be any digit, let‘s count the classes (A, B) out of the k points, the test data points get classified as the class with higher count. If k = 6, then the algorithm searches for the nearest point in the dataset, and count instances of every class out of the 6 point. Now assume counting turns out to be A = 2, B = 1, then KNN will allot A to the data set point.

In this scenario, Data points classification is done considering their proximity with known Data points classes, thus KNN is a supervised ML algorithm.

**47. What are advantages and limitations of KNN and K means?**

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm

that can be used to solve both classification and regression problems. It‘s easy to implement and understand, but has a major drawback of becoming significantly slows as the size of that data in use grows. KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression). In the case of classification and regression, we saw that choosing the right K for our data is done by trying several Ks and picking the one that works best.

#### Advantages

* The algorithm is simple and easy to implement.
* There‘s no need to build a model, tune several parameters, or make additional assumptions.
* The algorithm is versatile. It can be used for classification, regression, and search (as we will see in the next section).

#### Disadvantages

* The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

**Mathematics based questions 48. How to choose right value for K in KNN?** To select the K that‘s right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while

maintaining the algorithm‘s ability to accurately make predictions when it‘s given data it hasn‘t seen before. Here are some things to keep in mind:

* As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I‘m thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
* Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
* In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

**49. How to choose the value of "K number of clusters" in K-means Clustering?** The performance of the K-means clustering algorithm depends upon highly efficient clusters that it forms. But choosing the optimal number of clusters is a big task. There are some different ways to find the optimal number of clusters, but here we are discussing the most appropriate method to find the number of clusters or value of K. The method is given below: **Elbow Method**

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of**

**Squares**, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

WCSS= ∑Pi in Cluster1 distance(Pi C1)2 +∑Pi in Cluster2distance(Pi C2)2+∑Pi in CLuster3 distance(Pi C3)2 In the above formula of WCSS,

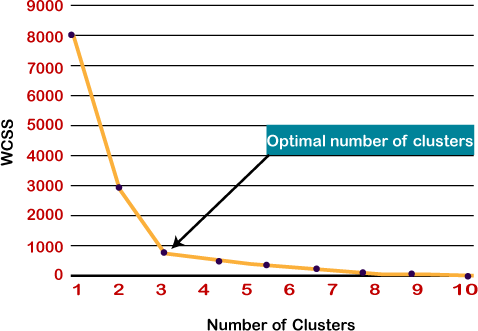
∑Pi in Cluster1 distance(Pi C1)2: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

To measure the distance between data points and centroid, we can use any method such as Euclidean distance or Manhattan distance.

To find the optimal value of clusters, the elbow method follows the below steps:

* It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
* For each value of K, calculates the WCSS value.
* Plots a curve between calculated WCSS values and the number of clusters K.
* The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method. The graph for the elbow method looks like the below image:



*Note: We can choose the number of clusters equal to the given data points. If we choose the number of clusters equal to the data points, then the value of WCSS becomes zero, and that will be the endpoint of the plot.*

The K-Nearest Neighbors (K-NN) algorithm is a popular Machine Learning algorithm used mostly for solving classification problems.

In this article, you'll learn how the K-NN algorithm works with practical examples.

We'll use diagrams, as well sample data to show how you can classify data using the K-NN algorithm. We'll also discuss the advantages and disadvantages of using the algorithm.

The k-nearest neighbors (KNN) algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. It is one of the popular and simplest classification and regression classifiers used in machine learning today.

While the KNN algorithm can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered “plurality voting”, the term, “majority vote” is more commonly used in literature. The distinction between these terminologies is that “majority voting” technically requires a majority of greater than 50%, which primarily works when there are only two categories. When you have multiple classes—e.g. four categories, you don’t necessarily need 50% of the vote to make a conclusion about a class; you could assign a class label with a vote of greater than 25%. The University of Wisconsin-Madison summarizes this well with an example [here](https://sebastianraschka.com/pdf/lecture-notes/stat479fs18/02_knn_notes.pdf) (link resides outside ibm.com).

## 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](https://www.geeksforgeeks.org/calculate-the-euclidean-distance-using-numpy) 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 an object.

distance(x,Xi)=∑j=1d(xj–Xij)2]distance(*x*,*Xi*​)=∑*j*=1*d*​(*xj*​–*Xij*​​)2​]

### Manhattan Distance

[Manhattan Distance](https://www.geeksforgeeks.org/how-to-calculate-manhattan-distance-in-r) 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(x,y)=∑i=1n∣xi−yi∣*d*(*x*,*y*)=∑*i*=1*n*​∣*xi*​−*yi*​∣

### Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the [Minkowski distance](https://www.geeksforgeeks.org/minkowski-distance-python).

d(x,y)=(∑i=1n(xi−yi)p)1p*d*(*x*,*y*)=(∑*i*=1*n*​(*xi*​−*yi*​)*p*)*p*1​

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](https://www.geeksforgeeks.org/machine-learning) problem but there are other distance metrics as well like [Hamming Distance](https://www.geeksforgeeks.org/hamming-distance-two-strings) 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 Does the K-Nearest Neighbors Algorithm Work?**

The K-NN algorithm compares a new data entry to the values in a given data set (with different classes or categories).

Based on its closeness or similarities in a given range (**K**) of neighbors, the algorithm assigns the new data to a class or category in the data set (training data).

Let's break that down into steps:

**Step #1 -** Assign a value to **K**.

**Step #2 -** Calculate the distance between the new data entry and all other existing data entries (you'll learn how to do this shortly). Arrange them in ascending order.

**Step #3 -** Find the **K** nearest neighbors to the new entry based on the calculated distances.

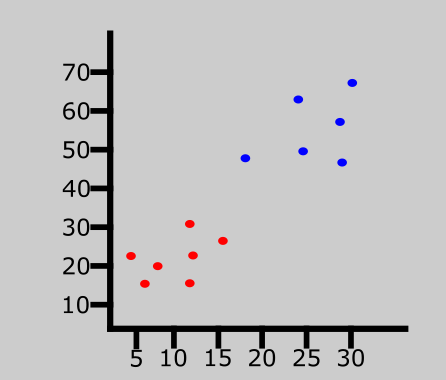
**Step #4 -** Assign the new data entry to the majority class in the nearest neighbors.

Don't worry if the steps above seem confusing at the moment. The examples in the sections that follow will help you understand better.

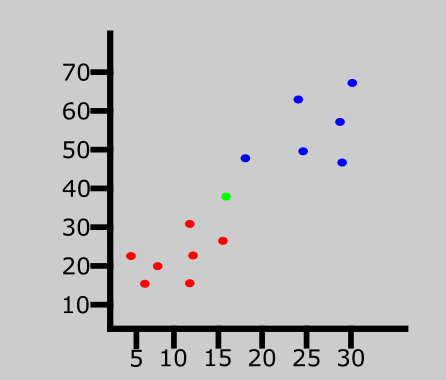
**K-Nearest Neighbors Classifiers and Model Example With Diagrams**

With the aid of diagrams, this section will help you understand the steps listed in the previous section.

Consider the diagram below:

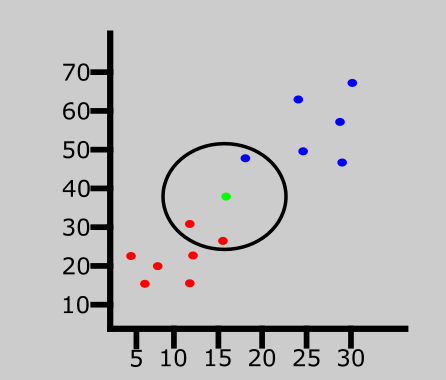


The graph above represents a data set consisting of two classes — red and blue.



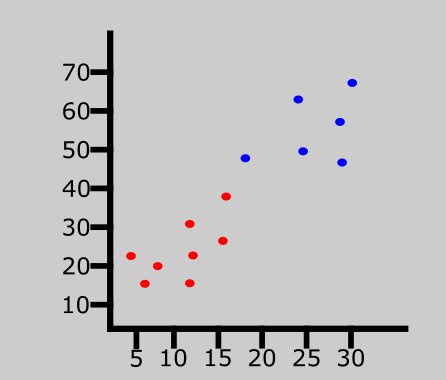
A new data entry has been introduced to the data set. This is represented by the green point in the graph above.

We'll then assign a value to **K** which denotes the number of neighbors to consider before classifying the new data entry. Let's assume the value of **K** is 3.



Since the value of **K** is 3, the algorithm will only consider the 3 nearest neighbors to the green point (new entry). This is represented in the graph above.

Out of the 3 nearest neighbors in the diagram above, the majority class is red so the new entry will be assigned to that class.



The last data entry has been classified as red.

**K-Nearest Neighbors Classifiers and Model Example With Data Set**

In the last section, we saw an example the K-NN algorithm using diagrams. But we didn't discuss how to know the distance between the new entry and other values in the data set.

In this section, we'll dive a bit deeper. Along with the steps followed in the last section, you'll learn how to calculate the distance between a new entry and other existing values using the Euclidean distance formula.

Note that you can also calculate the distance using the Manhattan and Minkowski distance formulas.

Let's get started!

| Brightness | Saturation | Class |
| --- | --- | --- |
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |

The table above represents our data set. We have two columns — **Brightness** and **Saturation**. Each row in the table has a class of either **Red** or **Blue**.

Before we introduce a new data entry, let's assume the value of **K** is 5.

**How to Calculate Euclidean Distance in the K-Nearest Neighbors Algorithm**

Here's the new data entry:

| Brightness | Saturation | Class |
| --- | --- | --- |
| 20 | 35 | ? |

We have a new entry but it doesn't have a class yet. To know its class, we have to calculate the distance from the new entry to other entries in the data set using the Euclidean distance formula.

Here's the formula: √(X₂-X₁)²+(Y₂-Y₁)²

Where:

* X₂ = New entry's brightness (20).
* X₁= Existing entry's brightness.
* Y₂ = New entry's saturation (35).
* Y₁ = Existing entry's saturation.

Let's do the calculation together. I'll calculate the first three.

**Distance #1**

For the first row, d1:

| Brightness | Saturation | Class |
| --- | --- | --- |
| 40 | 20 | Red |

d1 = √(20 - 40)² + (35 - 20)²  
= √400 + 225  
= √625  
= 25

We now know the distance from the new data entry to the first entry in the table. Let's update the table.

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | ? |
| 60 | 90 | Blue | ? |
| 10 | 25 | Red | ? |
| 70 | 70 | Blue | ? |
| 60 | 10 | Red | ? |
| 25 | 80 | Blue | ? |

**Distance #2**

For the second row, d2:

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 50 | 50 | Blue | ? |

d2 = √(20 - 50)² + (35 - 50)²  
= √900 + 225  
= √1125  
= 33.54

Here's the table with the updated distance:

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 60 | 90 | Blue | ? |
| 10 | 25 | Red | ? |
| 70 | 70 | Blue | ? |
| 60 | 10 | Red | ? |
| 25 | 80 | Blue | ? |

**Distance #3**

For the third row, d3:

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 60 | 90 | Blue | ? |

d2 = √(20 - 60)² + (35 - 90)²  
= √1600 + 3025  
= √4625  
= 68.01

Updated table:

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 60 | 90 | Blue | 68.01 |
| 10 | 25 | Red | ? |
| 70 | 70 | Blue | ? |
| 60 | 10 | Red | ? |
| 25 | 80 | Blue | ? |

At this point, you should understand how the calculation works. Attempt to calculate the distance for the last four rows.

Here's what the table will look like after all the distances have been calculated:

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 60 | 90 | Blue | 68.01 |
| 10 | 25 | Red | 10 |
| 70 | 70 | Blue | 61.03 |
| 60 | 10 | Red | 47.17 |
| 25 | 80 | Blue | 45 |

Let's rearrange the distances in ascending order:

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |
| 70 | 70 | Blue | 61.03 |
| 60 | 90 | Blue | 68.01 |

Since we chose 5 as the value of **K**, we'll only consider the first five rows. That is:

| Brightness | Saturation | Class | Distance |
| --- | --- | --- | --- |
| 10 | 25 | Red | 10 |
| 40 | 20 | Red | 25 |
| 50 | 50 | Blue | 33.54 |
| 25 | 80 | Blue | 45 |
| 60 | 10 | Red | 47.17 |

As you can see above, the majority class within the 5 nearest neighbors to the new entry is **Red**. Therefore, we'll classify the new entry as **Red**.

Here's the updated table:

| Brightness | Saturation | Class |
| --- | --- | --- |
| 40 | 20 | Red |
| 50 | 50 | Blue |
| 60 | 90 | Blue |
| 10 | 25 | Red |
| 70 | 70 | Blue |
| 60 | 10 | Red |
| 25 | 80 | Blue |
| 20 | 35 | Red |

**How to Choose the Value of K in the K-NN Algorithm**

There is no particular way of choosing the value **K**, but here are some common conventions to keep in mind:

* Choosing a very low value will most likely lead to inaccurate predictions.
* The commonly used value of **K** is 5.
* Always use an odd number as the value of **K**.

**Advantages of K-NN Algorithm**

* It is simple to implement.
* No training is required before classification.

**Disadvantages of K-NN Algorithm**

* Can be cost-intensive when working with a large data set.
* A lot of memory is required for processing large data sets.
* Choosing the right value of **K** can be tricky.

**Summary**

In this article, we talked about the K-Nearest Neighbors algorithm. It is often used for classification problems.

We saw an example using diagrams to explain how the algorithms works.

We also saw an example using sample data to see the steps involved in classifying a new data entry.

Lastly, we discussed the advantages and disadvantages of the algorithm, and how you can choose the value of **K**.