K Nearest Neighbors

A man is known by the company he keeps

K Nearest Neighbors

- KNN Theory

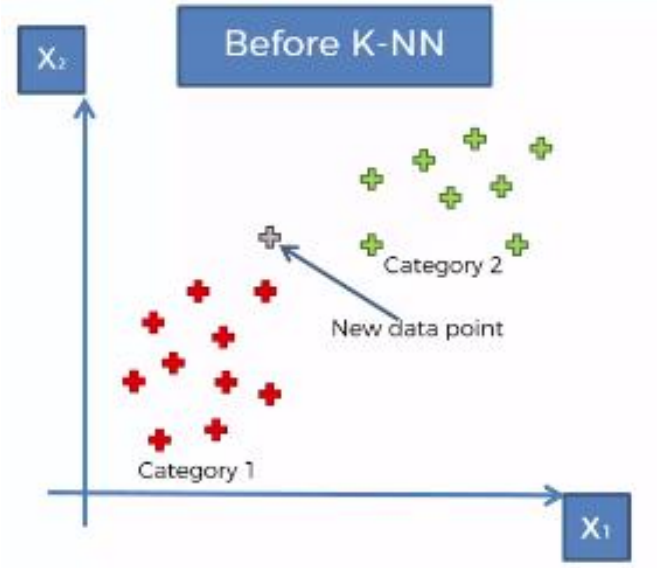
KNN Theory

KNN - Introduction

K Nearest Neighbors is a **classification** (and regression) algorithm that operates on a very simple principle of distance of some nearest data points.

It is suggested for binary as well as multiclass classification.

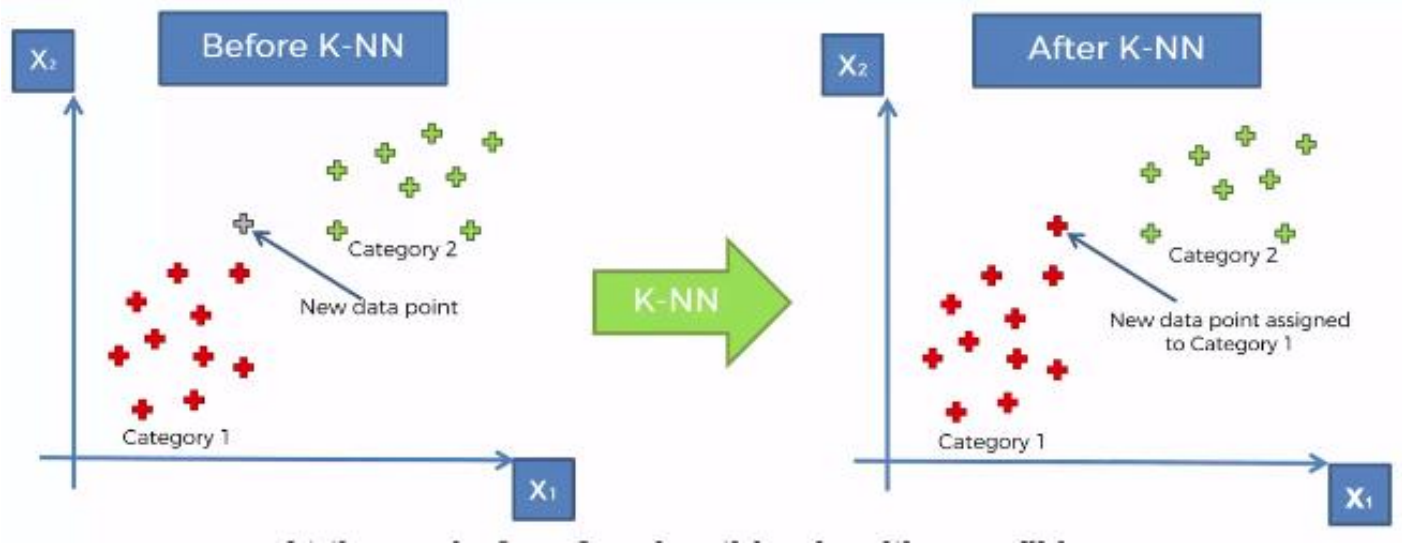
Lets see an example:
Imagine that we have a scenario where we have two categories present in the dataset, and lets say we add a new data point.



KNN - Introduction

Example:
Imagine that we have a scenario where we have two categories present in the dataset, and lets say we add a new data point.
Question is: Should it fall into the red category or the blue category ?
How do we classify this new data point ?

This is where KNN will help us, by the time we apply KNN to this problem, we will get results something like this:



KNN – How it works ?

Lets see how the KNN algorithm works

STEP 1: Choose the number of 'K' neighbors

Note that there is no standard way of defining the optimal value of 'K'.
We do that using certain workarounds. We'll see that while modelling.
Most common value for 'K' is 5.

STEP 2: Find the distance of each point from our new observation point.

Note: Usually 'Euclidean distance' is used which is default in sklearn.
However, you can use other distances such as Minkowski distance, Manhattan distance, Hamming distance etc.

STEP 3: Find the nearest 'K' points to the observation point and count the categories for each point among 'K'

e.g. Find the nearest 5 points and count how many are 'red' and how many are 'green' among those 5 points.

STEP 4: Assign the new data point to the category where you counted the most neighbors

KNN – How it works ?

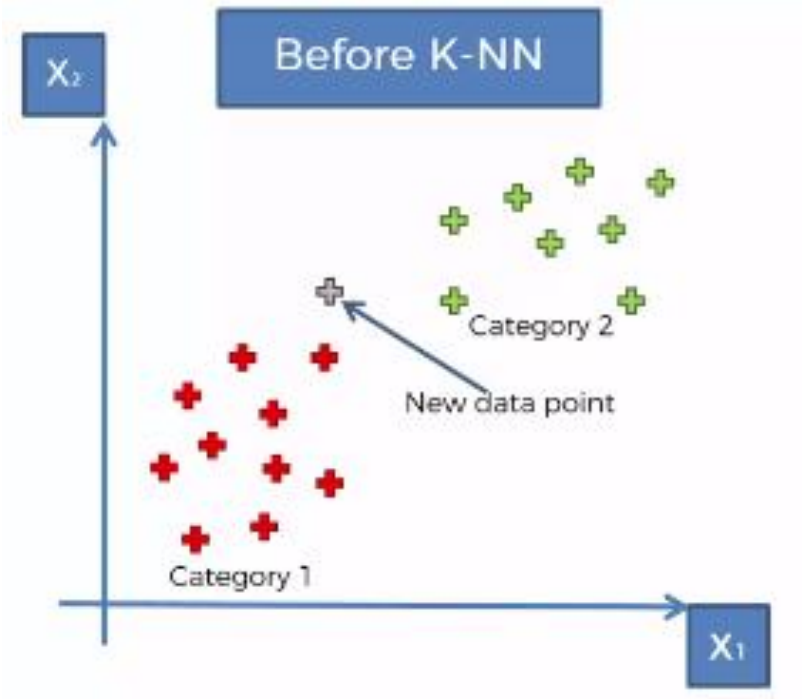
Example:

So here is our previous example. We have ‘red’ and ‘green’ categories.

A new data point is added to our model.

Lets see how to classify this manually.

We will use Euclidean Distance for this classification.



KNN – How it works ?

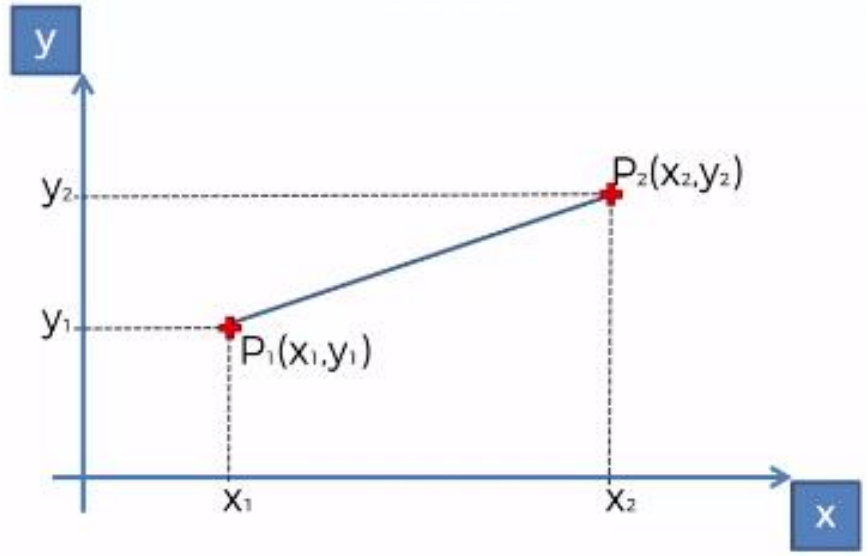
Example:

So here is our previous example. We have ‘red’ and ‘green’ categories.

A new data point is added to our model.

Lets see how to classify this manually.

We will use Euclidean Distance for this classification.



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

KNN – How it works ?

Example:

So here is our previous example. We have ‘red’ and ‘green’ categories.

A new data point is added to our model.

Lets see how to classify this manually.

We will use Euclidean Distance for this classification.

We identify the 5 closest points using distance along with their classification.

3 are ‘red’ and 2 are ‘green’

KNN – How it works ?

Example:

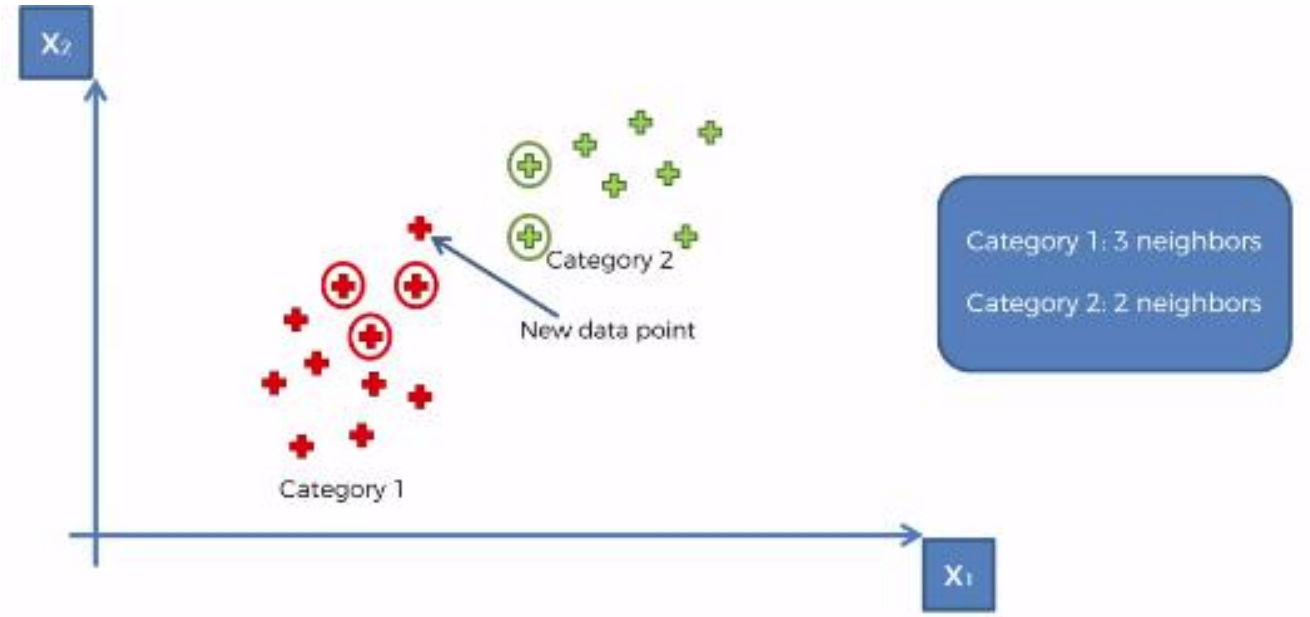
So here is our previous example. We have ‘red’ and ‘green’ categories.

A new data point is added to our model.

Lets see how to classify this manually.

We will use Euclidean Distance for this classification.

Hence the point is classified as ‘red’ point



KNN – Pros & Cons

Pros:

- Very simple
- Training is trivial. You just sort your data.
- Multi class classification. It works with any number of classes
- Easy to add more data
- It has very few parameters to tune
  - K (number of neighbors)
  - Distance metric (type of distance)

Cons:

- **High Prediction Cost: I**t is worse for large datasets because it has to sort the dataset.
- **Not good with high dimensional data:** As features increase and you get higher dimensions in your data, it becomes really difficult to measure distances in various dimensions.
- Categorical Features do not work well with KNN