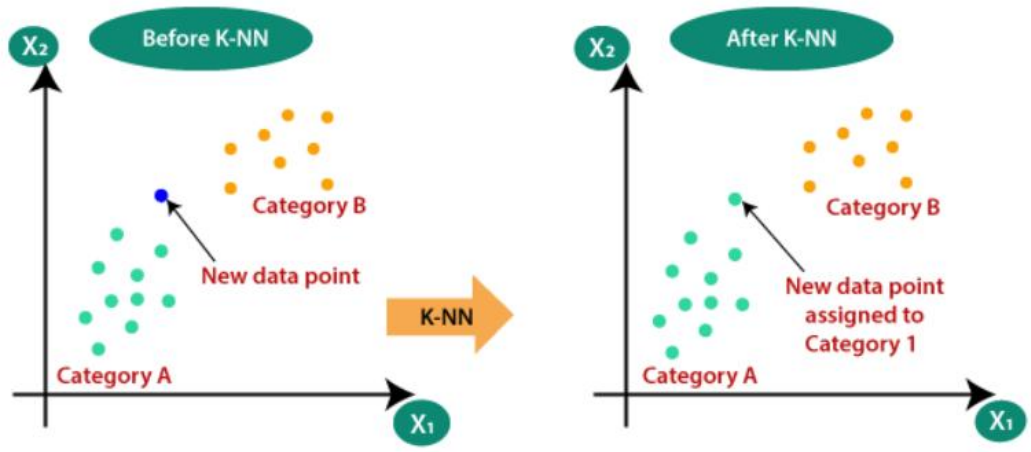
K-Nearest Neighbors (KNN) Classifier

K-Nearest Neighbors (**KNN**) is a **simple and powerful** machine learning algorithm used for **classification and regression**.

How KNN Works?

1. **Choose a value of k** (number of neighbors).
2. **Find the k nearest data points** based on distance (e.g., Euclidean distance).
3. **Majority voting** (for classification) → The most common class among the k neighbors is assigned to the new data point.
4. **Averaging** (for regression) → The output is the average of k nearest values.



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 neighbor is maximum.

- **Step-6:** Our model is ready.

Implementing KNN in Python

```
from sklearn.neighbors import KNeighborsClassifier


# Create KNN model with k=5

knn = KNeighborsClassifier(n_neighbors=5)


# Train the model

knn.fit(trainData, trainLabel)


# Make predictions

predictions = knn.predict(testData)


# Evaluate the model

from sklearn.metrics import classification_report

print(classification_report(testLabel, predictions))
```

Choosing the Best k Value

- **Small k (e.g., 1-3)** → Can lead to **overfitting** (too sensitive to noise).
- **Large k (e.g., 10-20)** → Leads to **smoother decision boundaries**, reducing variance.
- **Common choice:** k = sqrt(n), where n is the number of training samples.

Advantages of KNN

✅ **Simple & Easy to implement**
✅ **Non-parametric** (doesn't assume a specific data distribution)
✅ **Works well for small datasets**

Disadvantages of KNN

❌ **Computationally expensive** (slow on large datasets)
❌ **Sensitive to irrelevant features** (feature scaling is necessary)
❌ **Doesn't work well with high-dimensional data**

Ref: https://www.tpointtech.com/k-nearest-neighbor-algorithm-for-machine-learning