

## Assignment 02

**Title:** Implement KNN classifier for given data

**Aim:** Implement K-Nearest Neighbors (KNN) algorithm .

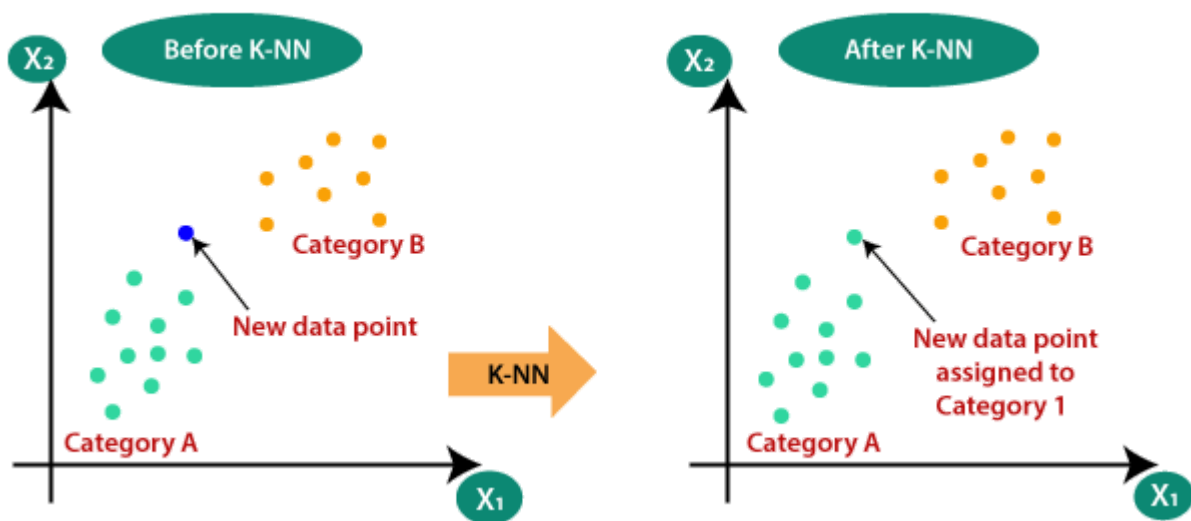
**Objectives:** To Understand the kNN classifier.

### Theory:

- 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.

### 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  $x_1$ , 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:

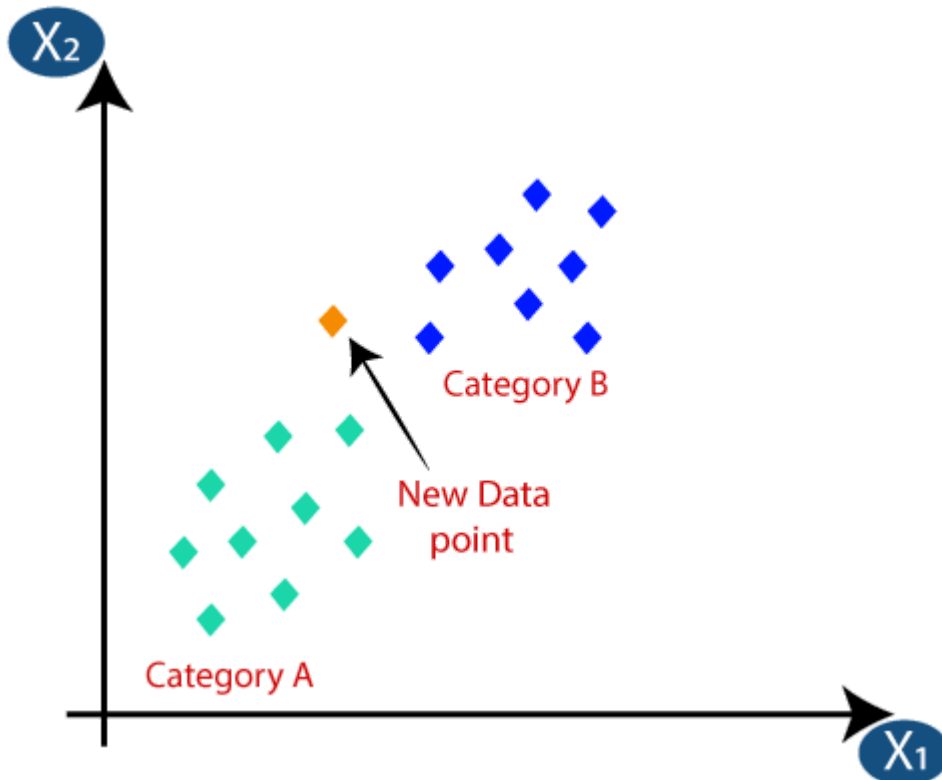


## How does K-NN work?

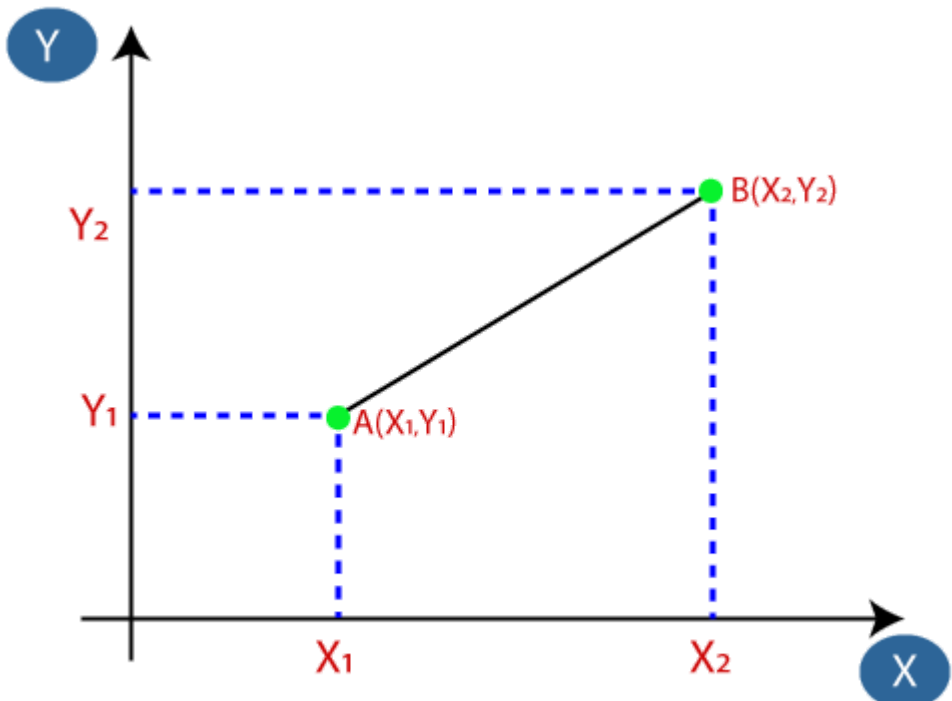
The K-NN working can be explained on the basis of the below algorithm:

- **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 neighbor is maximum.
- **Step-6:** Model is ready

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

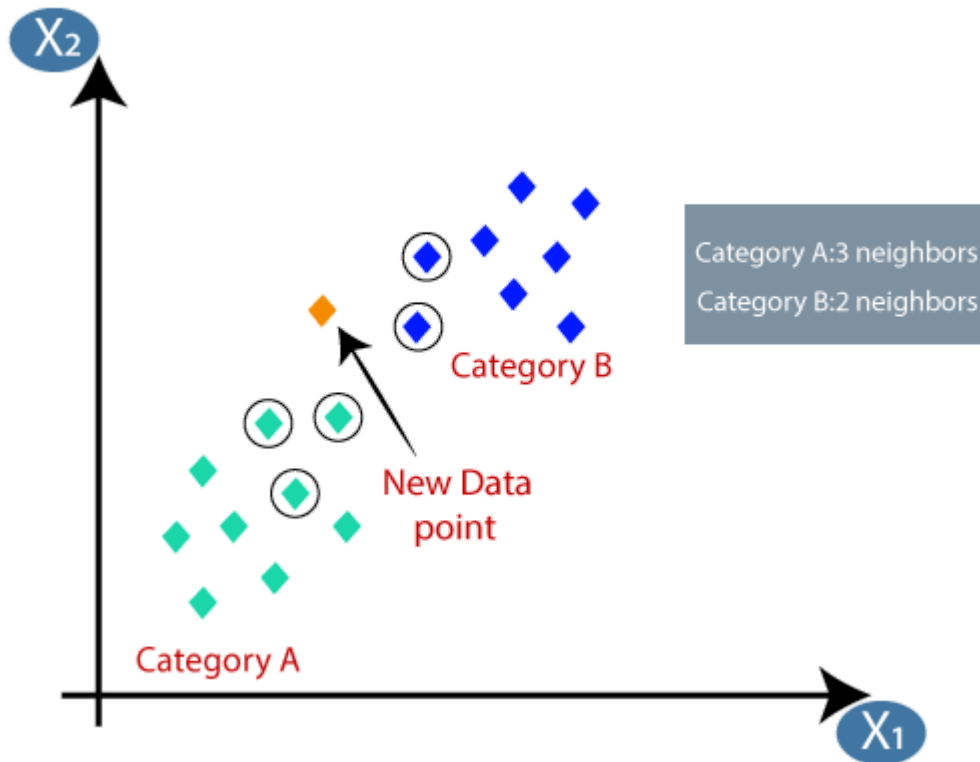


- Firstly, we will choose the number of neighbors, so we will choose the  $k=5$ .
- Next, we will calculate the **Euclidean distance** between the data points. The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:



$$\text{Euclidean Distance between } A_1 \text{ and } B_2 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

- 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.

## **How to select the value of K in the K-NN Algorithm?**

Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.
- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.
- Large values for K are good, but it may find some difficulties.

## **Advantages of KNN Algorithm:**

- It is simple to implement.
- It is robust to the noisy training data
- It can be more effective if the training data is large.

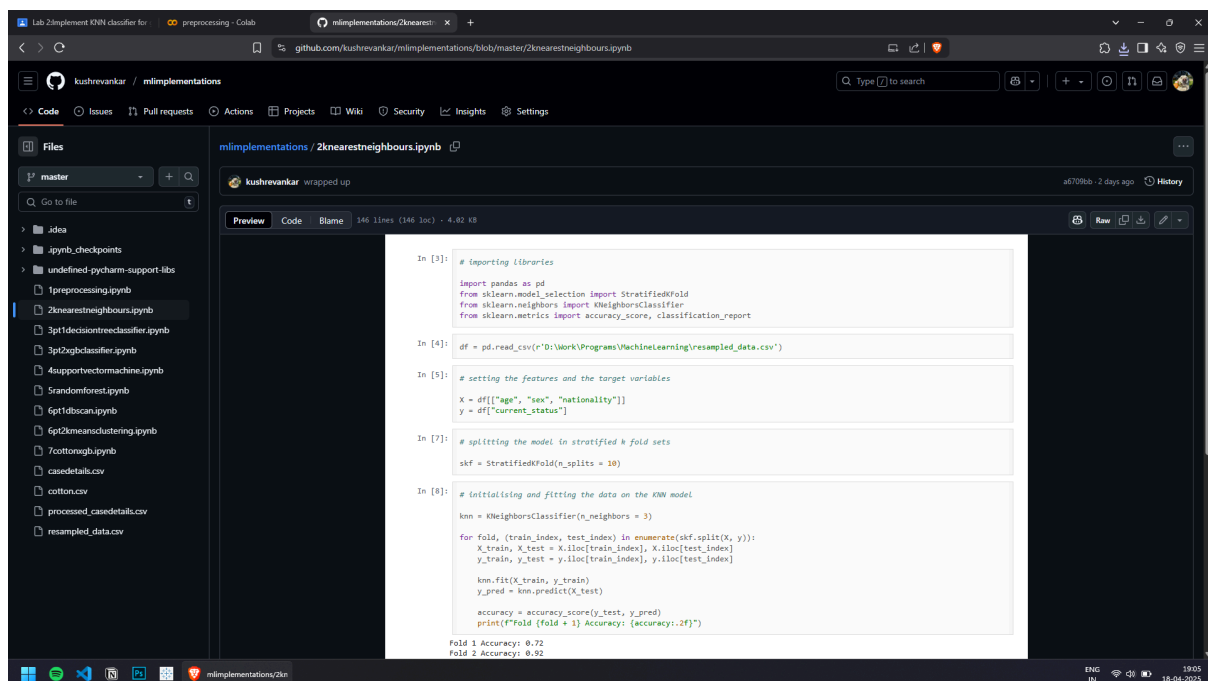
## Disadvantages of KNN Algorithm:

- Always needs to determine the value of K which may be complex some time.
- The computation cost is high because of calculating the distance between the data points for all the training samples.

## FAQs:

1. In k-NN, what does "k" stand for, and how does its value affect the algorithm's predictions? Explain why using too few or too many neighbors can be problematic.
2. How does k-NN compare the similarity between data points? Discuss different distance metrics (e.g., Euclidean, Manhattan) and their suitability for various types of data.
3. Imagine you're a new data point. How does k-NN determine which other data points are your "nearest neighbors"? Describe the steps involved in this process.
4. How does k-NN use the information from its nearest neighbors to make predictions for new data points? Explain the different approaches for classification and regression tasks.
5. What are some key advantages and limitations of the k-NN algorithm? Discuss scenarios where it might be a good choice or where other algorithms might be more suitable.

## Code & Output:



The screenshot displays a Jupyter Notebook titled 'mimplementations / 2knearestneighbours.ipynb' within a web browser. The interface includes a sidebar with a file explorer showing various .ipynb files and a main area with a code editor and a preview output. The code in the notebook is as follows:

```
In [3]: # importing libraries
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

In [4]: df = pd.read_csv(r"D:\Work\Programs\MachineLearning\resampled_data.csv")

In [5]: # setting the features and the target variables
X = df[["age", "sex", "nationality"]]
y = df["current_status"]

In [7]: # splitting the model in stratified k fold sets
skf = StratifiedKFold(n_splits = 10)

In [8]: # initialising and fitting the data on the KNN model
knn = KNeighborsClassifier(n_neighbors = 3)
for fold, (train_index, test_index) in enumerate(skf.split(X, y)):
    X_train, X_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Fold {fold + 1} Accuracy: {accuracy:.2f}")

Fold 1 Accuracy: 0.72
Fold 2 Accuracy: 0.92
```

The screenshot shows a Jupyter Notebook interface with a dark theme. The browser address bar indicates the notebook is located at `github.com/kushrevanka/mmlimplementations/blob/master/2knearestneighbours.ipynb`. The left sidebar contains a file explorer with a list of files and folders, including `2knearestneighbours.ipynb`, which is currently selected. The main area of the notebook displays the following code and output:

```
In [7]: # splitting the model in stratified k fold sets
skf = StratifiedKFold(n_splits = 10)

In [8]: # initialising and fitting the data on the KNN model
knn = KNeighborsClassifier(n_neighbors = 3)

for fold, (train_index, test_index) in enumerate(skf.split(X, y)):
    x_train, x_test = X.iloc[train_index], X.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    knn.fit(x_train, y_train)
    y_pred = knn.predict(x_test)

    accuracy = accuracy_score(y_test, y_pred)
    print(f"Fold {fold + 1} Accuracy: {accuracy:.2f}")

Fold 1 Accuracy: 0.72
Fold 2 Accuracy: 0.92
Fold 3 Accuracy: 0.93
Fold 4 Accuracy: 0.92
Fold 5 Accuracy: 0.92
Fold 6 Accuracy: 0.93
Fold 7 Accuracy: 0.90
Fold 8 Accuracy: 0.89
Fold 9 Accuracy: 0.90
Fold 10 Accuracy: 0.90

In [10]: # classification report (accuracy, precision, recall, f1_score)
classificationreport = classification_report(y_test, y_pred)
print("Classification report: \n", classificationreport)

Classification report:
              precision    recall  f1-score   support

0               0.85         0.96         0.90         579
1               0.96         0.83         0.89         580

accuracy               0.90         0.90         0.90         1159
macro avg              0.90         0.90         0.90         1159
weighted avg           0.90         0.90         0.90         1159
```