Classification

K-Nearest Neighbour

Capgemini

Lesson Objectives

On completion of this lesson, you will be able to understand :

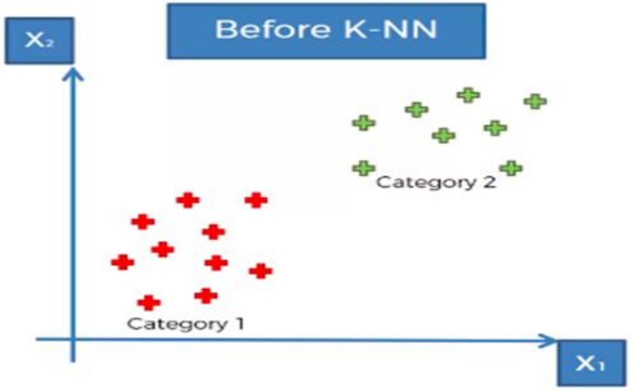
- The key-steps of how to create a KNN Model :
- Train the Model
- Test the Model

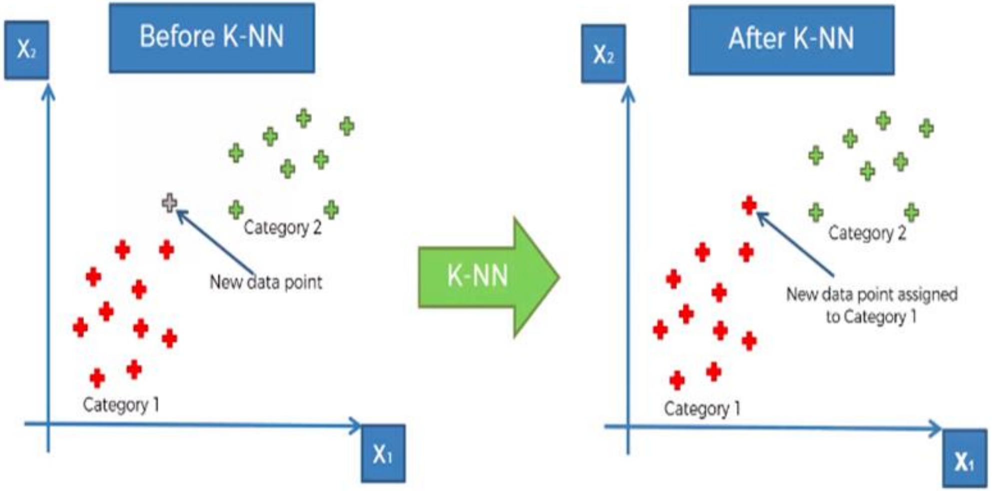
K Nearest Neighbour

- **K**NN is a classification-algorithm to predict the category of an observation **by storing the cases of the training-data and classifies all future/new cases(test cases)**

- It classifies a new data-point or observation based on a similarity-measurement

- K-Nearest Neighbour helps us identify that the said new data-point lies or falls into which particular group





K Nearest Neighbour : Sequence of steps

Step 1:

Choose some value for K or the number of nearest Neighbours

Step 2 :

Find out the K=5 Nearest Neighbours from the new element or data-point.

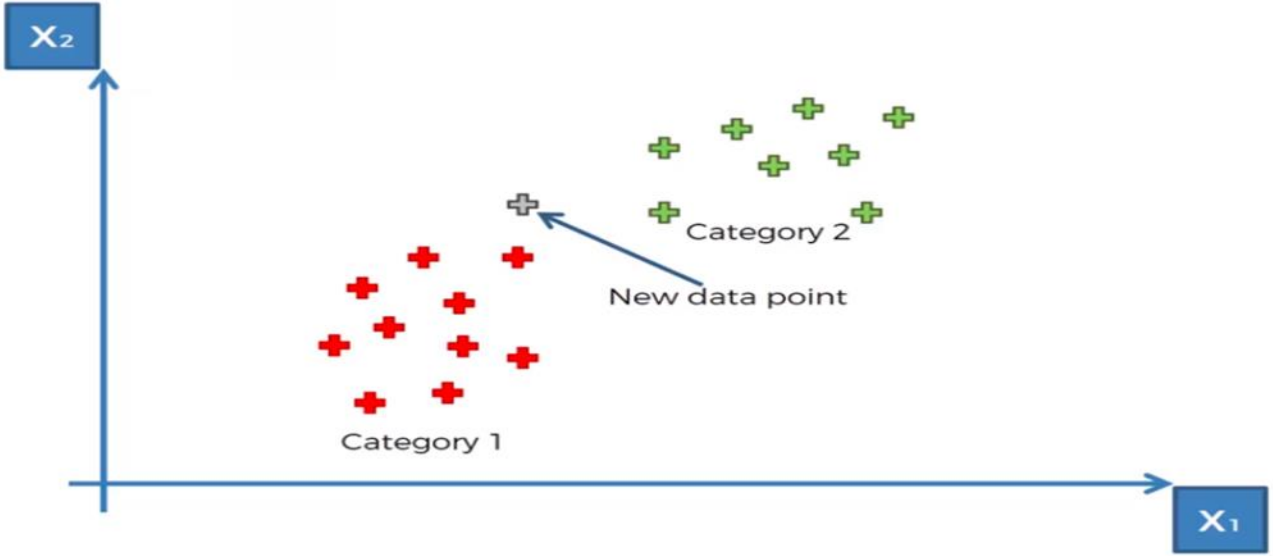
Step 3 :

Count the number of K Nearest Neigbours of the new data-point on a class-wise basis

Step 4 :

Assign the new data-point to the class where you found the most number of K Nearest Neighbours

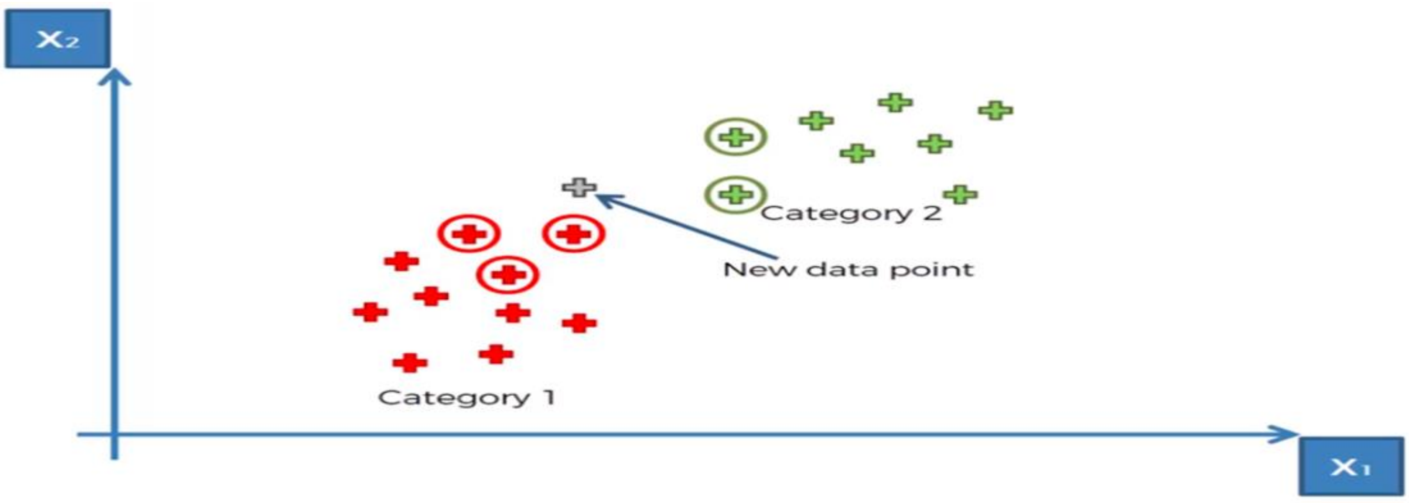
Step 1: Choose some value for K or the number of Neighbours



Randomly taking K =5

Step 2 : Find out the K=5 Nearest Neighbours from the new element or data-point. Euclidean distance or any other method can be used to calculate the distance.

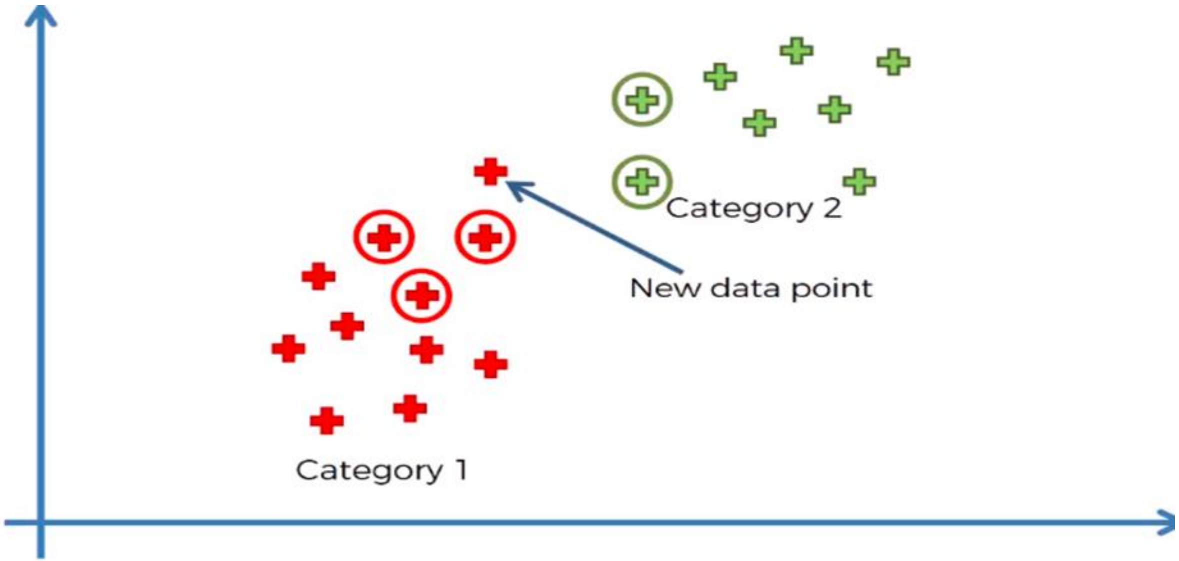
Step 3 : Count the number of K Nearest Neigbours of the new data-point on a class-wise basis



K Nearest Neighbour : Sequence of steps

…..contd.

Step 4 : Assign the new data-point to the class where you found the most number of K Nearest Neighbours



Importing the Libraries

- Change the directory to the path where our data-set lies “C:\BigData\MachineLearning\Machine Learning A-Z Template Folder\Part 3 - Classification\Section 15 - K-Nearest Neighbors (K-NN)”

- Import the basic libraries

```
#Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Importing the Dataset

- Import the Dataset

```
# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
```

- Create the matrix of Independent Variables(IVs)

```
x = dataset.iloc[:,[2,3]].values
```

- Create the vector of Dependent Variable(DV)

```
y = dataset.iloc[:,4].values
```

- Check the values of “x” and “y”

Splitting the Data into Training-set and Test-set

- Import the train_test_split function from the library sklearn.model_selection :

```
# Splitting the Data into Training-set and Test-set
from sklearn.model_selection import train_test_split
```

- Create 4 variables : x_train, x_test, y_train, y_test as follows :

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.25, random_state= 0)
```

- Our training-sets : x_train(as training data from the matrix of features of independent variables)and y_train(as training data from the vector of dependent variables associated with x_train)

- Our test-sets : x_test(as test-data from the matrix of features of independent variables) and y_test(as test-data from the vector of dependent variables associated with x_test)

Feature Scaling

- To prevent any IV from the matrix of IVs from dominating the entire ANN architecture, we need to implement feature-scaling

```
#Feature scaling the Data
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Fitting the K-Nearest Neighbour

- Importing the K-Nearest Neighbour Class

```
 from sklearn.neighbors import KNeighborsClassifier
classifier.fit(x_train, y_train)
```

- Create an object for the Logistic-Regression Model

```
classifier = KNeighborsClassifier(n_neighbors = 5)
```

- Fit the Logistic-Regression Object to the Training-set

```
classifier.fit(x_train, y_train)
```

Testing the K-Nearest Neighbour Model

- Test the model on the Test-dataset

```
#Predictions on the Test-set
y_pred = classifier.predict(x_test)
```

- Check and Compare the values of the variables y_pred and y_test

- Using the Confusion-Matrix function to check the accuracy of the predictions

N=200	Predicted FALSE	Predicted TRUE
Actual FALSE	100	12
Actual TRUE	13	75
Percentage of Accuracy (100+75)/200		0.875

Testing the K-Nearest Neighbour Model ….contd.

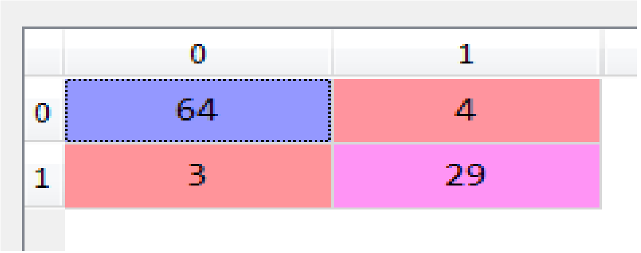
- Import the function to create the Confusion Matrix

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
```

- Create an object to implement the Confusion Matrix for our data

```
cm = confusion_matrix(y_test, y_pred)
```

- Check the value of the “cm” variable



	0	1
0	64	4
1	3	29

Thank You