**LAB # 03**

**K-NEAREST NEIGHBOR (KNN) ALGORITHM**

**OBJECTIVE**

Implementing K-Nearest Neighbor (KNN) algorithm to classify the data set.

**THEORY**

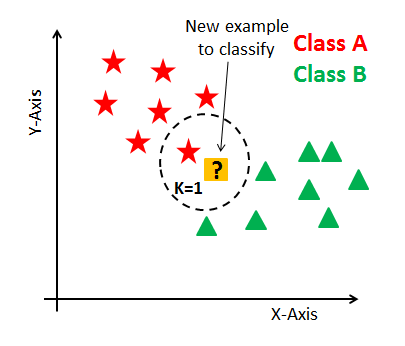
# K-NEAREST NEIGHBOR:

K-Nearest Neighbor (KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN is used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition. In Credit ratings, financial institutes will predict the credit rating of customers. In loan disbursement, banking institutes will predict whether the loan is safe or risky. In political science, classifying potential voters in two classes will vote or won’t vote. KNN algorithm is used for both classification and regression problems. KNN algorithm is based on feature similarity approach.

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In other words, the model structure determined from the dataset. This will be very helpful in practice where most of the real world datasets do not follow mathematical theoretical assumptions. Lazy algorithm means it does not need any training data points for model generation. All training data used in the testing phase. This makes training faster and testing phase slower and costlier. Costly testing phase means time and memory. In the worst case, KNN needs more time to scan all data points and scanning all data points will require more memory for storing training data.

## How does the KNN algorithm work?

In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor. K is generally an odd number if the number of classes is 2. When K=1, then the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to be predicted. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.



Suppose P1 is the point, for which label needs to be predicted. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance etc. KNN has the following basic steps:

1. Calculate distance
2. Find closest neighbors
3. Vote for labels

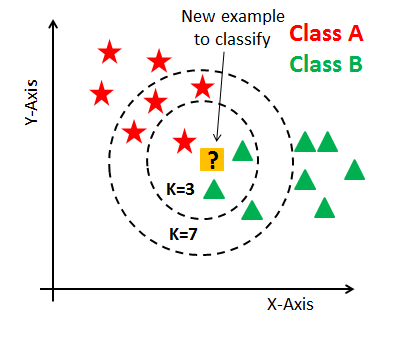


## How do you decide the number of neighbors in KNN?

The number of neighbors (K) in KNN is a hyper parameter that you need to choose at the time of model building. You can think of K as a controlling variable for the prediction model.

Research has shown that no optimal number of neighbors suits all kind of data sets. Each dataset has its own requirements. In the case of a small number of neighbors, the noise will have a higher influence on the result, and a large number of neighbors make it computationally expensive. Research has also shown that a small amount of neighbors are most flexible fit which will have low bias but high variance and a large number of neighbors will have a smoother decision boundary which means lower variance but higher bias.

Generally, Data scientists choose as an odd number if the number of classes is even. You can also check by generating the model on different values of k and check their performance.



# Pseudocode: (K-Neighbors Classifier):

* Import the required libraries.

from sklearn import preprocessing

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, accuracy\_score

* Assign features and label variables.
* Perform label encoding on all columns. Scikit-learn provides LabelEncoder library for encoding. It is used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

#creating labelEncoder

le = preprocessing.LabelEncoder()

# Converting string into numbers.

weather\_encoded=le.fit\_transform(weather)

* Combine all the features in a single variable (list of tuples).

features=list(zip(feature1\_encoded,feature2\_encoded,….))

* Split dataset into training and testing sets

from sklearn.model\_selection import train\_test\_split

features\_train, features\_test,label\_train,label\_test=train\_test\_split(features, play\_encoded, test\_size=0.2, random\_state=42)

* Generate a model using K-Neighbors classifier

model = KNeighborsClassifier(n\_neighbors=3, metric='euclidean')

* Fit the dataset on classifier

model.fit(features\_train, label\_train)

* Perform prediction

predicted = model.predict(features\_test)

* Print prediction

print("Prediction:", predicted)

* Confusion Matrix

conf\_mat = confusion\_matrix(label\_test, predicted)

print("Confusion Matrix:")

print(conf\_mat)

* Accuracy

accuracy = accuracy\_score(label\_test, predicted)

print("Accuracy:", accuracy)

**Confusion Matrix Interpretation:**

\* True Positives (TP): Correctly predicted "Yes" instances.

\* False Positives (FP): Incorrectly predicted "Yes" instances.

\* True Negatives (TN): Correctly predicted "No" instances.

\* False Negatives (FN): Incorrectly predicted "No" instances.

* [[2 0]

[0 2]]

\* True Positives (TP): 2

\* False Positives (FP): 0

\* True Negatives (TN): 2

\* False Negatives (FN): 0

**Accuracy Calculation**:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Accuracy = (2 + 2) / (2 + 2 + 0 + 0)

Accuracy = 1.0 or 100%

**Lab Tasks:**

**Weather Temperature Play**

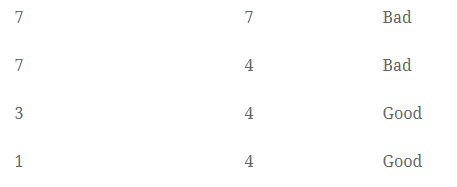
#### 

#### Fig 1

1.Implement K-Nearest Neighbor (KNN) Algorithm on the above dataset in Fig 1 to predict whether the players can play or not when the weather is overcast and the temperature is mild.Also apply confusion Matrix.

2.Here are 4 training samples. The two attributes are acid durability and strength. Now the factory produces a new tissue paper that passes laboratory test with X1=3 and X2=7. Predict the classification of this new tissue.

**X1= Acid durability (sec)**  **X2=Strength (kg/m2)** **Y=Classification**



* Calculate the Euclidean Distance between the query instance and all the training samples. Coordinate of query instance is (3,7).

Text

Description automatically generated with medium confidence

Suppose K = number of nearest neighbors = 3, sort the distances and determine nearest neighbors. Gather the class (Y) of the nearest neighbors. Use majority of the category of nearest neighbors as the prediction value of the query instance