**KNN implementation using python**

import numpy as np

from collections import Counter

#Euclidian Distance

def euclidean\_distance(x1, x2):

return np.sqrt(np.sum((x1 - x2)\*\*2))

class k\_nearest\_neighbors:

def \_\_init\_\_(self, k):

self.k = k

def knn\_fit(self, X\_train, y\_train):

self.X\_train = X\_train

self.y\_train = y\_train

def knn\_predict(self, X):

predicted\_lables = [self.\_predict(x) for x in X]

return np.array(predicted\_lables)

#helper method

def \_predict(self, x):

#compute distances

distances = [euclidean\_distance(x, x\_train) for x\_train in self.X\_train]

#get k nearest samples, labels

k\_indices = np.argsort(distances)[:self.k]

k\_nearest\_labels = [self.y\_train[i] for i in k\_indices]

#majority vote, most common class label

majority\_vote = Counter(k\_nearest\_labels).most\_common(1)

return majority\_vote[0][0]