In [1]:
```
#Razat Siwakoti (A00046635)
#DMV302 - Assessment 2
#WeightedNN.ipynb created on Jupyter notebook


#source: Bollos00.(2020, July 11)
#https://stackoverflow.com/questions/62844914/nearest-neighbor-using-customized-weights-

#Visual Studio Magazine (2019)
#https://visualstudiomagazine.com/articles/2019/04/01/weighted-k-nn-classification.aspx
```

In [2]:
```
import numpy as np
class WeightedNearestNeighbor:
#Initialize class attribute for training data
    def __init__(self):
        self.x_train = None
        self.y_train = None

 #Fit the model with training data
    def fit(self, x_train, y_train):
        self.x_train = x_train
        self.y_train = y_train

#Get the number of test samples and training samples
    def predict(self, x_test):
        num_test = x_test.shape[0]
        num_train = self.x_train.shape[0]

# Number of neighbors to consider, considering all training samples
        k = num_train  # considering all training samples

# Array to store predicted class labels for each test sample
        y_pred = np.zeros(num_test)

# Loop over each test sample
        for i in range(num_test):
            # Compute distances between x_test[i] and all x_train
            distances = np.sqrt(np.sum((self.x_train - x_test[i])**2, axis=1))

            # Calculate weights based on the provided formula
            # w_i = ||t_i - b|| / ∑(||t_m - b||)
            weights = distances / np.sum(distances)

            # Predict the class label based on weights
            y_pred[i] = np.round(np.sum(self.y_train * weights))

        return y_pred
```

In [4]:
```
# Assuming training data
x_train = np.random.rand(10, 5)  # Example 10 samples in R^5
y_train = np.random.randint(2, size=10)  # Random labels 0 or 1

# Initialize the classifier and fit the training data
classifier = WeightedNearestNeighbor()
classifier.fit(x_train, y_train)

# Assuming test data
x_test = np.random.rand(3, 5)  # Example 3 test samples in R^5
# Predict the class labels for the test data
y_pred = classifier.predict(x_test)
# Print the predicted class labels
print(y_pred)
```

```
[1. 1. 1.]
```

In [ ]: