Assignment 3: Report

# **K nearest neighbor:**

## Dataset:

The data balance-scale was used, this data set was generated to model psychological experimental results. Each example is classified as having the balance scale tip to the right, tip to the left, or be balanced. The attributes are the left weight, the left distance, the right weight, and the right distance. The correct way to find the class is the greater of (left-distance \* left-weight) and (right-distance \*right-weight). If they are equal, it is balanced.

## Data-Splitting:

Data was divided into two chunks, one for training and other for testing, the training one consisted of 75% of total data, meanwhile the testing was remaining data.

## Total Classes and k-value:

Since, there were total of three classes in dataset, L B R, which were change to numeric form: 0 1 2.

The code was tested on k value ranging from 4 to 9.

The value of K can’t be equal to number of classes, in case if each feature vector from each class is nearest to testing vector, then the result has high chance to be incorrect. The value of k was kept minimum due to very high similarity between datasets.

## KNN Algorithm:

The training feature vectors and their classes were sent to the constructor of KNN, the data was saved in KNN class member variables.

The testing data was sent to classification function of the KNN class, which sent each individual feature vector in the testing dataset to the classify\_Data function, classify data function calculated the Euclidean distance from current feature vector to every other feature vector in the training data set, then picked up the k number of nearest training feature vector (**k number of vectors having least Euclidean distance**), then checked, to which class our data was nearest (that is class with max vectors in k selected vectors). And our testing feature was included in that class. This was done for every feature vector, and in the end, we got class for each vector set and then we compared the classes of testing dataset found by knn against the classes that were actually given to find the accuracy.

Then the **confusion matrix** was created for the data, and precision, recall and f1-score was found by using the formulas: TP / TP + FP, TP / TP + FN, and (2 \* Precision \* Recall) / (Precision + Recall) respectively, where False positive was our row for that class, and false negative was column.

The result can be seen by running the python code.

# **K means clustering:**

Since, this algo is based on clustering itself, the data was not divided into chunks, only the class column was dropped, the data was shuffled, then it was divided into k number of chunks, each chunk was used to get an initial centroid.

Then for each feature vector its Euclidean distance from every centroid was calculated, and then that feature vector was clustered to that centroid, in the end when every feature vector was clustered to its nearest centroid, then average of every cluster was calculated, and that was new value of every corresponding centroid. The process continued until 100 iterations were reached or there was same average in two iterations.

After the centroids were found and each feature vector was clustered, Davies Bouldin index was calculated.

There were five steps in Davies Bouldin Index to calculate the quality:

#Step 1: Calculate intra-cluster dispersion

#Step 2: Calculate separation measure

#Step 3: Calculate similarity between clusters

#Step 4: Find most similar cluster for each cluster i

#Step 5: Calculate the Davies-Bouldin Index

After these five steps were perform, the quality was yielded.

This process was repeated for value of k from 3 to 9. Again, large value of k was not used due to high similarity between feature vectors.