National University of Computer and Emerging Sciences

******

Parallel and Distributed Computing

Project Report

*for*

**K-Nearest Neighbour Parallel(KNN-P) Algorithm**

**Group Members:**

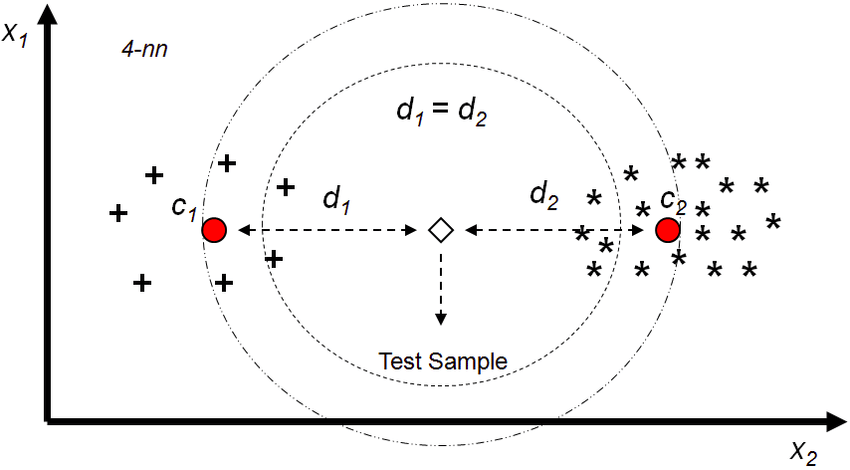
* Ahmed Umer 20K-1736
* Zain Vohra 20K-1712

**Introduction**

K-Nearest Neighbour(KNN) is a supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. *Similarity* or *closeness* between data points is calculated by applying distance formulas on different data points.

**Problem Statement**

KNN calculates the distance between each test data with all the training data to find the closest *or nearest* neighbors of the test data in order to classify it. Due to rigorous calculation, KNN is often known as the Lazy *learning algorithm*. Distance calculation for every instance in test data is independent of each other. Therefore, distance calculation can be parallelized in order to enhance the KNN algorithm.



**Methodology**

First of all, the training dataset is divided among all the available processors using the MPI\_Scatter method. After division of training data, each row of the test dataset is passed through MPI\_SCATTER once again to calculate the distance. Each process calculates the distance of the test dataset from part of the training dataset available to the processor. Distances calculated by each processor are sorted using quickSort. After that all distances are gathered at the master/parent processor and are sorted once again. Class of each test case predicted in each loop based on the K number of neighbors.

**MPI Code Snippets**

1. **MPI\_init**

MPI\_Init(NULL, NULL)

1. **MPI\_Comm\_rank**

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank)

1. **MPI\_Comm\_Size**

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

1. **MPI\_Scatter**

MPI\_Scatter(X\_train,

data\_size\_process,

MPI\_FLOAT,

data\_per\_process,

data\_size\_process,

MPI\_FLOAT,

0,

MPI\_COMM\_WORLD);

1. **MPI\_Gather(for distance\_per\_process)**

MPI\_Gather(distance\_per\_process,

row\_size\_process,

MPI\_FLOAT,

Overall\_distance,

row\_size\_process,

MPI\_FLOAT,

0,

MPI\_COMM\_WORLD);

1. **MPI\_Gather(for labels\_per\_process)**

MPI\_Gather(labels\_per\_process,

row\_size\_process,

MPI\_FLOAT,

overall\_labels,

row\_size\_process,

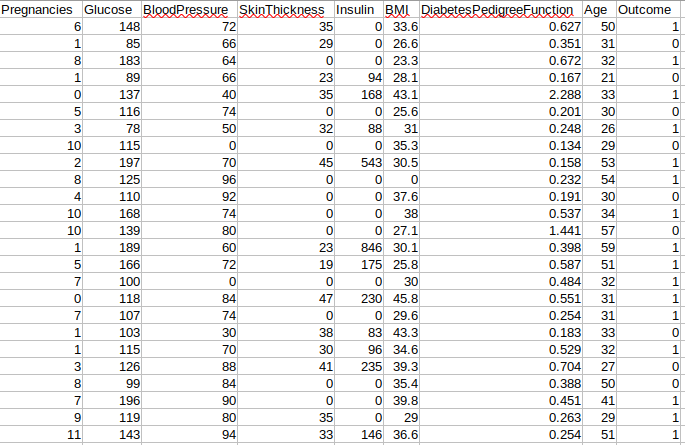
MPI\_FLOAT,

0,

MPI\_COMM\_WORLD);

**Dataset**

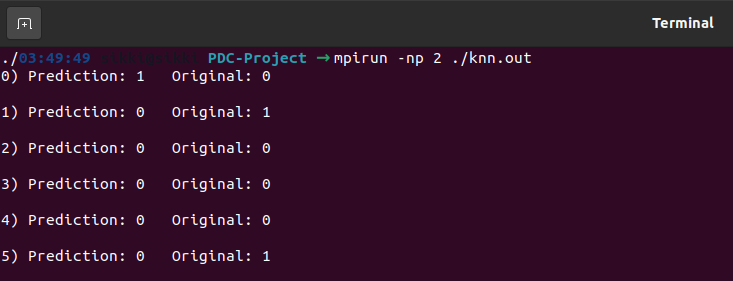
1. Diabetes Dataset



This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

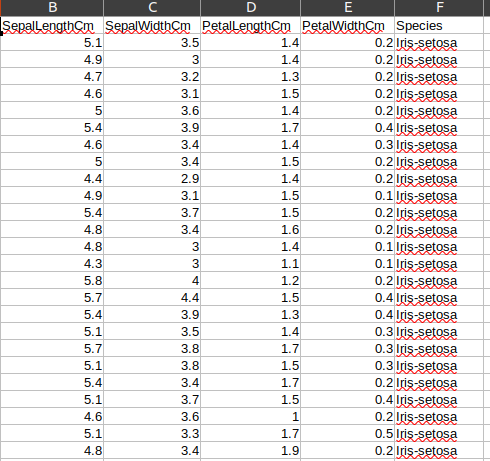
The datasets consist of 8 medical predictor variables and one target variable, Outcome. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

KNN-P on Diabetes Dataset:





1. IRIS Dataset



The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

* SepalLengthCm
* SepalWidthCm
* PetalLengthCm
* PetalWidthCm
* Species

KNN-P on Diabetes Dataset:



**Conclusion**

KNN is K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification of predictive problems in the industry. KNN calculates distance between each instance of test data with all of training data instances and classifies the test data according to the closest neighbors. Distance calculation makes KNN, a *lazy learning* algorithm. However, distance calculation for each instance of data is independent of all others. Therefore, KNN can be a parallized to produce faster results. Our project has implemented a parallel K-Nearest neighbor algorithm on two different datasets of different shapes. The algorithm gave convincing results.