**Big Data Hadoop Project-2 Submission**

**k-Means Cluster - LoadAcre Mobile**

**Step 1:**

**Data cleaning:**

The data file consists of 5 features whereas the program needs only latitude and longitude. So, the below python script copies the respective columns and puts it in a new file.

**Python Script:**

import csv

list1=[]

with open('/Users/kaushiksekar/Documents/Hadoop Files/Projects/BDHS\_Projects/Project for submission/Project 2/network\_ml.csv') as csvfile:

for row in csvfile:

row1=row.split(',')

list1.append(str(row1[3])+','+str(row1[4]))

row1=''

with open('/Users/kaushiksekar/Documents/Hadoop Files/Projects/BDHS\_Projects/Project for submission/Project 2/network\_ml\_clean.csv','w') as writeFile:

for a in list1:

writeFile.write(a)

**Step 2:**

**Details:**

Import the data and use k-means cluster algorithm from the spark-ml package. This will form clusters based on data and predict the cluster in which each data will be in.

**Program:**

package org.sparkcourse.realtimeproject.MobileNetworkCoverage;

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaRDD;

import org.apache.spark.api.java.JavaSparkContext;

import org.apache.spark.mllib.clustering.KMeans;

import org.apache.spark.mllib.clustering.KMeansModel;

import org.apache.spark.mllib.linalg.Vector;

import org.apache.spark.mllib.linalg.Vectors;

import org.apache.spark.api.java.function.\*;

public class NetworkClustering

{

public static void main( String[] args )

{

SparkConf sparkConf=new SparkConf().setMaster("local").setAppName("Network Clustering");

JavaSparkContext sc=new JavaSparkContext(sparkConf);

String path = "/Users/kaushiksekar/Documents/Hadoop Files/Projects/BDHS\_Projects/Project for submission/Project 2/network\_ml\_clean.csv";

JavaRDD<String> data = sc.textFile(path);

JavaRDD<Vector> parsedData = data.map(

new Function<String, Vector>() {

public Vector call(String s) {

String[] sarray = s.split(",");

double[] values = new double[sarray.length];

for (int i = 0; i < sarray.length; i++) {

values[i] = Double.parseDouble(sarray[i]);

}

return Vectors.dense(values);

}

}

);

parsedData.cache();

int numberOfClusters = 3;

int numberOfIterations = 20;

KMeansModel clusters = KMeans.train(parsedData.rdd(), numberOfClusters, numberOfIterations);

for (Vector center : clusters.clusterCenters()) {

System.out.println(" " + center);

}

double WSSSE = clusters.computeCost(parsedData.rdd());

System.out.println("Within Set Sum of Squared Errors = " + WSSSE);

JavaRDD<Integer> clusterNumbers=clusters.predict(parsedData);

int count=1;

for(int clusterNumber:clusterNumbers.collect()) {

count++;

System.out.println(clusterNumber);

if(count==11) {

break;

}

}

sc.close();

}

}