**Sanitation Checks of Gujarat and UP**

# PROJECT REPORT

*Submitted in fulfilment for the JComponent of*

*CSE4001 – Parallel and Distributed Computing*

**CAL Course**

***in***

**B.TECH**

*by*

# DEVAM GAURANG JOSHI- 15BCE0199

# BHAVYA BATTA – 15BCB0089

*Under the guidance of*

# Prof. Manoov R

# SCOPE

****

**School of Information Technology and Engineering**

Fall Semester 2017-18

# Abstract

We are trying to implement the clustering and classification techniques on a given data set. The project, takes into consideration the data set of Achievement of IHHL (Individual Household Latrine) of the states of Gujarat and Uttar Pradesh. We then take the difference in actual values to the optimum safe value and make two clusters, namely - Sanitised and Unsanitised.

In the end, we list out the districts according to the two clusters obtained and hence the necessary steps the government needs to take in follow up with the scheme of Swachh Bharat Abhiyan.

# Only Gujarat met toilet-building target under Swachh Bharat: REPORT

Even as Prime Minister Narendra Modi praised efforts to clean up the Chandrapur fort in the October 29 edition of Mann ki Baat, a new report from TERI University, USAID and Coca-Cola, found that states are struggling the larger task of proper waste management under the Swachh Bharat Mission.

Barring Gujarat, none of the states in India met their targets for toilet-building in urban areas according to the report titled: ‘State of Urban Water and Sanitation in India’ released Friday.

The Swachh Bharat Mission (SBM) is a flagship programme of the Modi government, and many BJP-ruled states fare better on most indicators. However, three-quarters of the states did not even meet half of their targets for building toilets in households, until April 2017. Not even Gujarat met its target for common toilets and public toilets.

“Today, India is fighting a decisive battle for cleanliness and hygiene through the Swachh Bharat Mission. It is time for each one of us to commit ourselves towards ensuring public hygiene, personal hygiene and environmental hygiene. It is remarkable to note that the government, politicians, celebrities, NGOs, private enterprises are all working together to achieve a common goal,” Manoj Kumar, Additional Secretary, Ministry of Housing and Urban Affairs, said.

The urban development ministry is the nodal agency that is responsible for SBM in 4041 towns across India. The central government revised targets for toilet building downwards for urban areas from 1.04 crore to 66.42 lakh based on state government evaluations, which is a 36% decrease than the initial targets when SBM was launched.

States are struggling to meet even these lower targets. The report noted that along with construction of toilets, the government needs to revise the water supply to urban and rural areas to make the toilets usable.

The government has adopted a toilets- first approach, “because unless toilets are built we cannot move forward,” Arun Kansal, a co-author of the study and head of department, Department of Regional Water Studies, TERI University, said.

One of the key goals of SBM is to make the country ODF (Open Defecation Free) by 2019. But the numbers do not present a clear picture because it isn’t clear if places that have declared themselves ODF are actually free of open defecation may not be so. After a local body declares its area ODF, it has to be certified by the government and then independently verified. Third party verification has happened for very few states.

The report also highlighted that access to improved water sources has remained stagnant for 15 years between 2000 and 2015. Access to piped water which is considered the best for drinking water actually fell in the 15-year period.

# Problem Statement

The Modi government has stressed on the subject of cleanliness and have declared the famous scheme namely, Swachh Bharat Abhiyan for it. Still there is an urgent need to check if the scheme is working efficiently on the ground level. Hence, in this project we’re checking sanitation levels of the gram panchayats of two major states i.e. Gujarat and Uttar Pradesh.

# Input & Output Methods

### Input Method

br = new BufferedReader(new FileReader(csvFile)); while ((line = br.readLine()) != null) {

// use comma as separator

String[] DataLine = line.split(cvsSplitBy); double X=Double.parseDouble(DataLine[6]); String gramPanchayatName = DataLine[5];

// double Y=Double.parseDouble(DataLine[7]);

dataWriter.append(new ClusterCenter(new Vector(0,0)), new Vector(X, 0));

}

///////Input Is Taken From The AllGujarat File///////

### Output Method

package com.clustering.mapreduce;

import java.io.BufferedReader; import java.io.\*;

import java.io.FileNotFoundException; import java.io.FileReader;

import java.io.IOException; import java.util.ArrayList;

import com.clustering.model.ClusterCenter; import com.clustering.model.Vector;

public class Classifier {

public static void main(int Max,int Min,double BigC,double SmallC)

{

ArrayList<String> Unsanitized=new ArrayList<String>(); ArrayList<String> Sanitized=new ArrayList<String>(); ArrayList<Integer> Unsdata=new ArrayList<Integer>(); ArrayList<Integer> Sdata=new ArrayList<Integer>(); String csvFile = "/home/hadoop1/Desktop/Data/AllUP.csv"; BufferedReader br = null;

String line = "";

String cvsSplitBy = ",";

try {

br = new BufferedReader(new FileReader(csvFile)); while ((line = br.readLine()) != null) {

String[] DataLine = line.split(cvsSplitBy); int X=Integer.parseInt(DataLine[6]);

String gramPanchayatName = DataLine[3]; double Y=Double.parseDouble(DataLine[7]);

if(X>=Min && X<=Max)

{

Unsanitized.add(gramPanchayatName); Unsdata.add(X);

}

else

{

Sanitized.add(gramPanchayatName); Sdata.add(X);

}

}

System.out.println("Unsanitized"); try {

FileWriter fw1 = new FileWriter("/home/hadoop1/Desktop/Data/Bosu/ UnsanitizedUP.csv");

FileWriter fw2 = new FileWriter("/home/hadoop1/Desktop/Data/Bosu/ SanitizedUP.csv");

fw1.write(BigC+"\n");

for(int i=0;i<Unsanitized.size();i++)

{

fw1.write(Unsanitized.get(i) + "," + Unsdata.get(i) + "\n"); if(i<10)

{

System.out.print(","+Unsanitized.get(i));

}

}

fw1.close(); fw2.write(SmallC+"\n"); System.out.println("Sanitized"); for(int i=0;i<Sanitized.size();i++)

{

fw2.write(Sanitized.get(i) + "," + Sdata.get(i) + "\n"); if(i<10)

{

System.out.print(","+Sanitized.get(i));

}

}

fw2.close();

} catch (IOException e) { e.printStackTrace();

}

} }} /////Outpfiles have Been Generated With The Names Of GramPanchayats One Unsanitized File And The Other Is Sanitized File///

# K-Means Clustering Code

### K-Means Clustering Job

package com.clustering.mapreduce; import java.io.BufferedReader;

import java.io.FileNotFoundException; import java.io.FileReader;

import java.io.IOException; import java.util.ArrayList;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory; import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FileStatus;

import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.fs.Path; import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.SequenceFile; import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.lib.input.SequenceFileInputFormat; import org.apache.hadoop.mapreduce.lib.output.SequenceFileOutputFormat; import com.clustering.model.ClusterCenter;

import com.clustering.model.Vector; public class KMeansClusteringJob {

private static final Log LOG =LogFactory.getLog(KMeansClusteringJob.class); public static void main(String[] args) throws IOException,InterruptedException, ClassNotFoundException {

int iteration = 1;

Configuration conf = new Configuration(); conf.set("num.iteration", iteration + "");

Path in = new Path("files/clustering/import/data"); Path center = new Path("files/clustering/import/center/cen.seq"); conf.set("centroid.path", center.toString());

Path out = new Path("files/clustering/depth\_1"); Job job = new Job(conf); job.setJobName("KMeans Clustering"); job.setMapperClass(KMeansMapper.class); job.setReducerClass(KMeansReducer.class); job.setJarByClass(KMeansMapper.class); SequenceFileInputFormat.addInputPath(job, in); FileSystem fs = FileSystem.get(conf);

if (fs.exists(out)) fs.delete(out, true); if (fs.exists(center)) fs.delete(out, true); if (fs.exists(in)) fs.delete(out, true);

final SequenceFile.Writer centerWriter =SequenceFile.createWriter(fs,

conf, center, ClusterCenter.class, IntWritable.class);

final IntWritable value = new IntWritable(0); centerWriter.append(new ClusterCenter(new Vector(2,5.0)),value); centerWriter.append(new ClusterCenter(new Vector(500,1.0)),value); centerWriter.close();

final SequenceFile.Writer dataWriter = SequenceFile.createWriter(fs,conf, in, ClusterCenter.class,Vector.class);

String csvFile = "/home/hadoop1/Desktop/Data/AllGujarat.csv"; BufferedReader br = null;

String line = "";

String cvsSplitBy = ","; try {

br = new BufferedReader(new FileReader(csvFile)); while ((line = br.readLine()) != null) {

// use comma as separator

String[] DataLine = line.split(cvsSplitBy); double X=Double.parseDouble(DataLine[6]); String gramPanchayatName = DataLine[5];

// double Y=Double.parseDouble(DataLine[7]);

dataWriter.append(new ClusterCenter(new Vector(0,0)), new Vector(X, 0));

}

}catch (FileNotFoundException e) { e.printStackTrace();

} catch (IOException e) { e.printStackTrace();

}finally {

if (br != null) { try {

br.close();

} catch (IOException e) { e.printStackTrace();

}

}

dataWriter.close(); SequenceFileOutputFormat.setOutputPath(job, out); job.setInputFormatClass(SequenceFileInputFormat.class); job.setOutputFormatClass(SequenceFileOutputFormat.class); job.setOutputKeyClass(ClusterCenter.class); job.setOutputValueClass(Vector.class); job.waitForCompletion(true);

long counter = job.getCounters().findCounter(KMeansReducer.Counter.CONVERGED).getValue(); iteration++;

while (counter > 0) {

conf = new Configuration();

conf.set("centroid.path", center.toString()); conf.set("num.iteration", iteration + ""); job = new Job(conf);

job.setJobName("KMeans Clustering " + iteration); job.setMapperClass(KMeansMapper.class); job.setReducerClass(KMeansReducer.class); job.setJarByClass(KMeansMapper.class);

in = new Path("files/clustering/depth\_" +(iteration - 1) + "/"); out = new Path("files/clustering/depth\_" + iteration); SequenceFileInputFormat.addInputPath(job, in);

if (fs.exists(out)) fs.delete(out, true);

SequenceFileOutputFormat.setOutputPath(job, out); job.setInputFormatClass(SequenceFileInputFormat.class); job.setOutputFormatClass(SequenceFileOutputFormat.class); job.setOutputKeyClass(ClusterCenter.class); job.setOutputValueClass(Vector.class); job.waitForCompletion(true);

iteration++;

counter = job.getCounters()

.findCounter(KMeansReducer.Counter.CONVERGED).getValue();

}

Path result = new Path("files/clustering/depth\_" + (iteration - 1) + "/");

FileStatus[] stati = fs.listStatus(result); for (FileStatus status : stati) {

if (!status.isDir() &&

!status.getPath().toString().contains("/\_")) { Path path = status.getPath(); LOG.info("FOUND " + path.toString()); SequenceFile.Reader reader = new SequenceFile.Reader(fs, path,

conf);

ClusterCenter key = new ClusterCenter(); Vector v = new Vector();

ArrayList<Integer> inputs = new ArrayList<Integer>(); while (reader.next(key, v)) {

LOG.info(key + " / " + v);

inputs.add((int)(v.getVector()[0]));

}

ArrayList<Double> finalClusters = Algorithm.main(inputs, 2); LOG.info("fC -> ");

LOG.info(finalClusters); reader.close();

}

}

}

}

}

### K-Means Mapper

package com.clustering.mapreduce; import java.io.IOException;

import java.util.LinkedList; import java.util.List;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.SequenceFile; import org.apache.hadoop.mapreduce.Mapper;

import com.clustering.model.ClusterCenter; import com.clustering.model.DistanceMeasurer; import com.clustering.model.Vector;

public class KMeansMapper extends Mapper<ClusterCenter, Vector, ClusterCenter, Vector>{ List<ClusterCenter> centers = new LinkedList<ClusterCenter>();

@Override

protected void setup(Context context) throws IOException, InterruptedException {

super.setup(context);

Configuration conf = context.getConfiguration(); Path centroids = new Path(conf.get("centroid.path")); FileSystem fs = FileSystem.get(conf);

SequenceFile.Reader reader = new SequenceFile.Reader(fs,centroids,conf); ClusterCenter key = new ClusterCenter();

IntWritable value = new IntWritable(); while (reader.next(key,value))

{

centers.add(new ClusterCenter(key));

}

reader.close();

}

@Override

protected void map(ClusterCenter key, Vector value, Context context) throws IOException, InterruptedException {

ClusterCenter nearest = null;

double nearestDistance = Double.MAX\_VALUE; for (ClusterCenter c : centers) {

double dist = DistanceMeasurer.measureDistance(c, value);

if (nearest == null) { nearest = c; nearestDistance = dist;

} else {

if (nearestDistance > dist) { nearest = c; nearestDistance = dist;

} 17

}

}

context.write(nearest, value);

}

}

### K-Means Reducer

package com.clustering.mapreduce; import java.io.IOException;

import java.util.LinkedList; import java.util.List;

import org.apache.hadoop.conf.Configuration; import org.apache.hadoop.fs.FileSystem; import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable; import org.apache.hadoop.io.SequenceFile; import org.apache.hadoop.mapreduce.Reducer; import com.clustering.model.ClusterCenter; import com.clustering.model.Vector;

public class KMeansReducer extends Reducer<ClusterCenter, Vector, ClusterCenter, Vector>{ public static enum Counter{

CONVERGED

}

List<ClusterCenter> centers = new LinkedList<ClusterCenter>(); protected void reduce(ClusterCenter key, Iterable<Vector> values, Context context) throws IOException,

InterruptedException{

Vector newCenter = new Vector();

List<Vector> vectorList = new LinkedList<Vector>(); int vectorSize = key.getCenter().getVector().length; newCenter.setVector(new double[vectorSize]); for(Vector value :values){

vectorList.add(new Vector(value));

for(int i=0;i<value.getVector().length;i++){ newCenter.getVector()[i]+=value.getVector()[i];

}

}

for(int i=0;i<newCenter.getVector().length;i++){ newCenter.getVector()[i] = newCenter.getVector()[i]/vectorList.size();

}

ClusterCenter center = new ClusterCenter(newCenter); centers.add(center);

for(Vector vector:vectorList){ context.write(center, vector);

}

if(center.converged(key)) context.getCounter(Counter.CONVERGED).increment(1);

} 18

protected void cleanup(Context context) throws IOException,InterruptedException{ super.cleanup(context);

Configuration conf = context.getConfiguration(); Path outPath = new Path(conf.get("centroid.path")); FileSystem fs = FileSystem.get(conf); fs.delete(outPath,true);

final SequenceFile.Writer out = SequenceFile.createWriter(fs, context.getConfiguration(),

outPath, ClusterCenter.class, IntWritable.class); final IntWritable value = new IntWritable(0); for(ClusterCenter center:centers){ out.append(center, value);

}

out.close();

}

}

**19**

# OUTPUT

### OutputLog

17/10/30 20:48:53 INFO mapred.LocalJobRunner:

17/10/30 20:48:53 INFO mapred.Task: Task:attempt\_local760962930\_0010\_r\_000000\_0 is done. And is in the process of commiting

17/10/30 20:48:53 INFO mapred.LocalJobRunner:

17/10/30 20:48:53 INFO mapred.Task: Task attempt\_local760962930\_0010\_r\_000000\_0 is allowed to commit now

17/10/30 20:48:53 INFO output.FileOutputCommitter: Saved output of task 'attempt\_local760962930\_0010\_r\_000000\_0' to files/clustering/depth\_10 17/10/30 20:48:53 INFO mapred.LocalJobRunner: reduce > reduce

17/10/30 20:48:53 INFO mapred.Task: Task 'attempt\_local760962930\_0010\_r\_000000\_0' done.

17/10/30 20:48:54 INFO mapred.JobClient: map 100% reduce 100%

17/10/30 20:48:54 INFO mapred.JobClient: Job complete: job\_local760962930\_0010 17/10/30 20:48:54 INFO mapred.JobClient: Counters: 20

17/10/30 20:48:54 INFO mapred.JobClient: Map-Reduce Framework 17/10/30 20:48:54 INFO mapred.JobClient: Spilled Records=52

17/10/30 20:48:54 INFO mapred.JobClient: Map output materialized bytes=1098 17/10/30 20:48:54 INFO mapred.JobClient: Reduce input records=26

17/10/30 20:48:54 INFO mapred.JobClient: Virtual memory (bytes) snapshot=0 17/10/30 20:48:54 INFO mapred.JobClient: Map input records=26

17/10/30 20:48:54 INFO mapred.JobClient: SPLIT\_RAW\_BYTES=146 17/10/30 20:48:54 INFO mapred.JobClient: Map output bytes=1040 17/10/30 20:48:54 INFO mapred.JobClient: Reduce shuffle bytes=0

17/10/30 20:48:54 INFO mapred.JobClient: Physical memory (bytes) snapshot=0 17/10/30 20:48:54 INFO mapred.JobClient: Reduce input groups=24

17/10/30 20:48:54 INFO mapred.JobClient: Combine output records=0 17/10/30 20:48:54 INFO mapred.JobClient: Reduce output records=26 17/10/30 20:48:54 INFO mapred.JobClient: Map output records=26 17/10/30 20:48:54 INFO mapred.JobClient: Combine input records=0 17/10/30 20:48:54 INFO mapred.JobClient: CPU time spent (ms)=0 17/10/30 20:48:54 INFO mapred.JobClient: Total committed heap usage (bytes)=2101346304

17/10/30 20:48:54 INFO mapred.JobClient: File Input Format Counters 17/10/30 20:48:54 INFO mapred.JobClient: Bytes Read=1357 17/10/30 20:48:54 INFO mapred.JobClient: FileSystemCounters

17/10/30 20:48:54 INFO mapred.JobClient: FILE\_BYTES\_WRITTEN=1102168

17/10/30 20:48:54 INFO mapred.JobClient: FILE\_BYTES\_READ=65108

17/10/30 20:48:54 INFO mapred.JobClient: File Output Format Counters 17/10/30 20:48:54 INFO mapred.JobClient: Bytes Written=1357

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: FOUND file:/home/hadoop1/ workspace/Bigdataproject/files/clustering/depth\_10/part-r-00000

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[70.0, 0.0]]] / Vector [vector=[70.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[71.0, 0.0]]] / Vector [vector=[71.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[270.0, 0.0]]] / Vector [vector=[270.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[153.0, 0.0]]] / Vector [vector=[153.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[1001.0, 0.0]]] / Vector [vector=[1001.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[469.0, 0.0]]] / Vector [vector=[469.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[0.0, 0.0]]] / Vector [vector=[0.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[0.0, 0.0]]] / Vector [vector=[0.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[66.0, 0.0]]] / Vector [vector=[66.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[288.0, 0.0]]] / Vector [vector=[288.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[13.0, 0.0]]] / Vector [vector=[13.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[30.0, 0.0]]] / Vector [vector=[30.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[50.0, 0.0]]] / Vector [vector=[50.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[59.0, 0.0]]] / Vector [vector=[59.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[84.0, 0.0]]] / Vector [vector=[84.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[136.0, 0.0]]] / Vector [vector=[136.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[100.0, 0.0]]] / Vector [vector=[100.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[0.0, 0.0]]] / Vector [vector=[0.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[0.0, 0.0]]] / Vector [vector=[0.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[85.0, 0.0]]] / Vector [vector=[85.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[21.0, 0.0]]] / Vector [vector=[21.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[73.0, 0.0]]] / Vector [vector=[73.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[70.0, 0.0]]] / Vector [vector=[70.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[20.0, 0.0]]] / Vector [vector=[20.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[110.0, 0.0]]] / Vector [vector=[110.0, 0.0]]

17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: ClusterCenter [center=Vector [vector=[156.0, 0.0]]] / Vector [vector=[156.0, 0.0]]

At this step Value of clusters

K1{ 70 0 0 66 13 30 50 59 0 0 21 70 20 }

K2{ 71 270 153 1001 469 288 84 136 100 85 73 110 156 }

Value of m

m1=30.692307692307693 m2=230.46153846153845

At this step Value of clusters

K1{ 70 71 0 0 66 13 30 50 59 84 100 0 0 85 21 73 70 20 110 }

K2{ 270 153 1001 469 288 136 156 }

Value of m

m1=48.526315789473685 m2=353.2857142857143

At this step Value of clusters

K1{ 70 71 153 0 0 66 13 30 50 59 84 136 100 0 0 85 21 73 70 20 110 156 }

K2{ 270 1001 469 288 }

Value of m m1=62.13636363636363 m2=507.0

At this step Value of clusters

K1{ 70 71 270 153 0 0 66 13 30 50 59 84 136 100 0 0 85 21 73 70 20 110 156 }

K2{ 1001 469 288 }

Value of m m1=71.17391304347827 m2=586.0

At this step Value of clusters

K1{ 70 71 270 153 0 0 66 288 13 30 50 59 84 136 100 0 0 85 21 73 70 20 110 156 }

K2{ 1001 469 }

Value of m m1=80.20833333333333 m2=735.0

At this step Value of clusters

K1{ 70 71 270 153 0 0 66 288 13 30 50 59 84 136 100 0 0 85 21 73 70 20 110 156 }

K2{ 1001 469 }

Value of m m1=80.20833333333333 m2=735.0

**22**

The Final Clusters By Kmeans are as follows: K1{

70 71 270 153 0 0 66 288 13 30 50 59 84 136 100 0 0 85 21 73 70 20 110 156 }K2{

1001 469 }2880

Unsanitized

,AKABARA,ARRUA KHAS,ARSENA,ATUS,BABROD,BARAULI,BASTAI,BHILAWATI,BIARA,BIDHAPURS

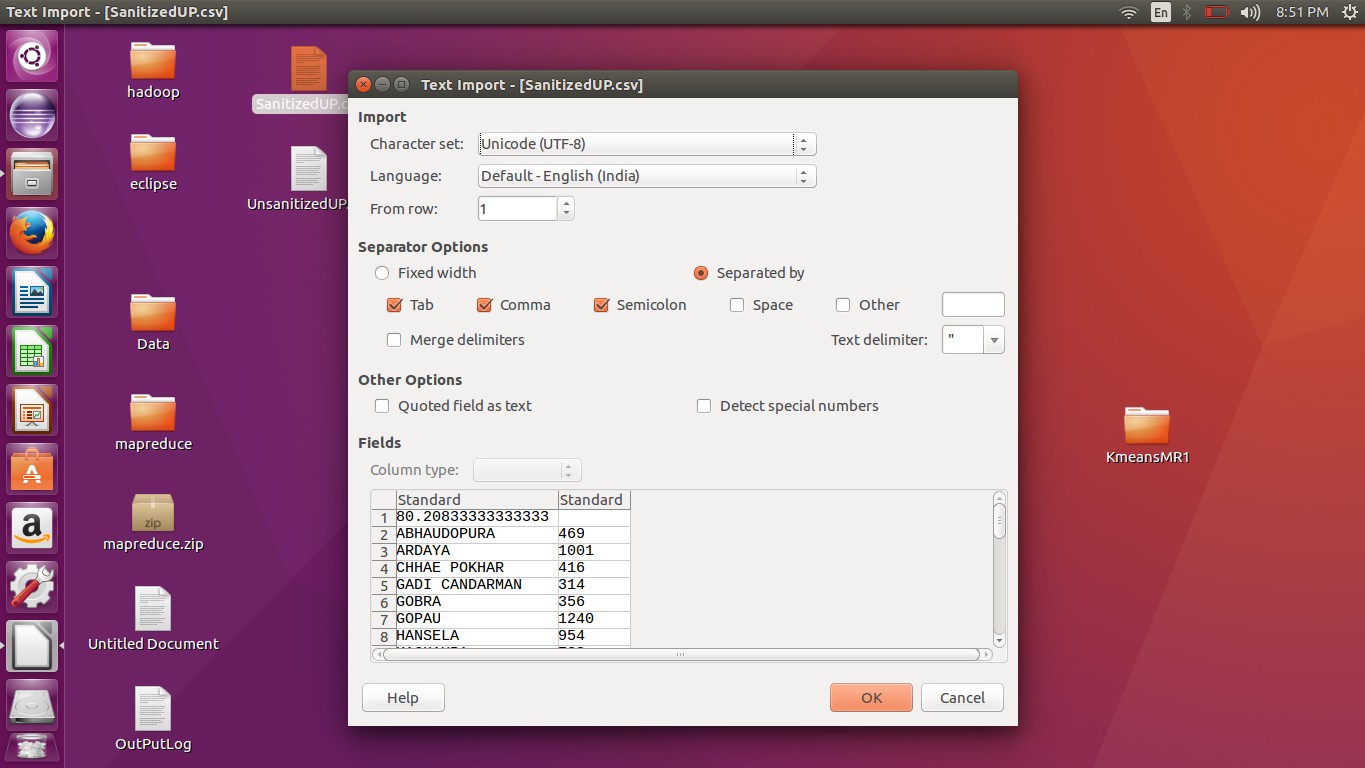
anitized

,ABHAUDOPURA,ARDAYA,CHHAE POKHAR,GADI CANDARMAN,GOBRA,GOPAU,HANSELA,KACHAURA,KATHAWARI,KHARBAI17/1

0/30 20:48:54 INFO mapreduce.KMeansClusteringJob: fC ->

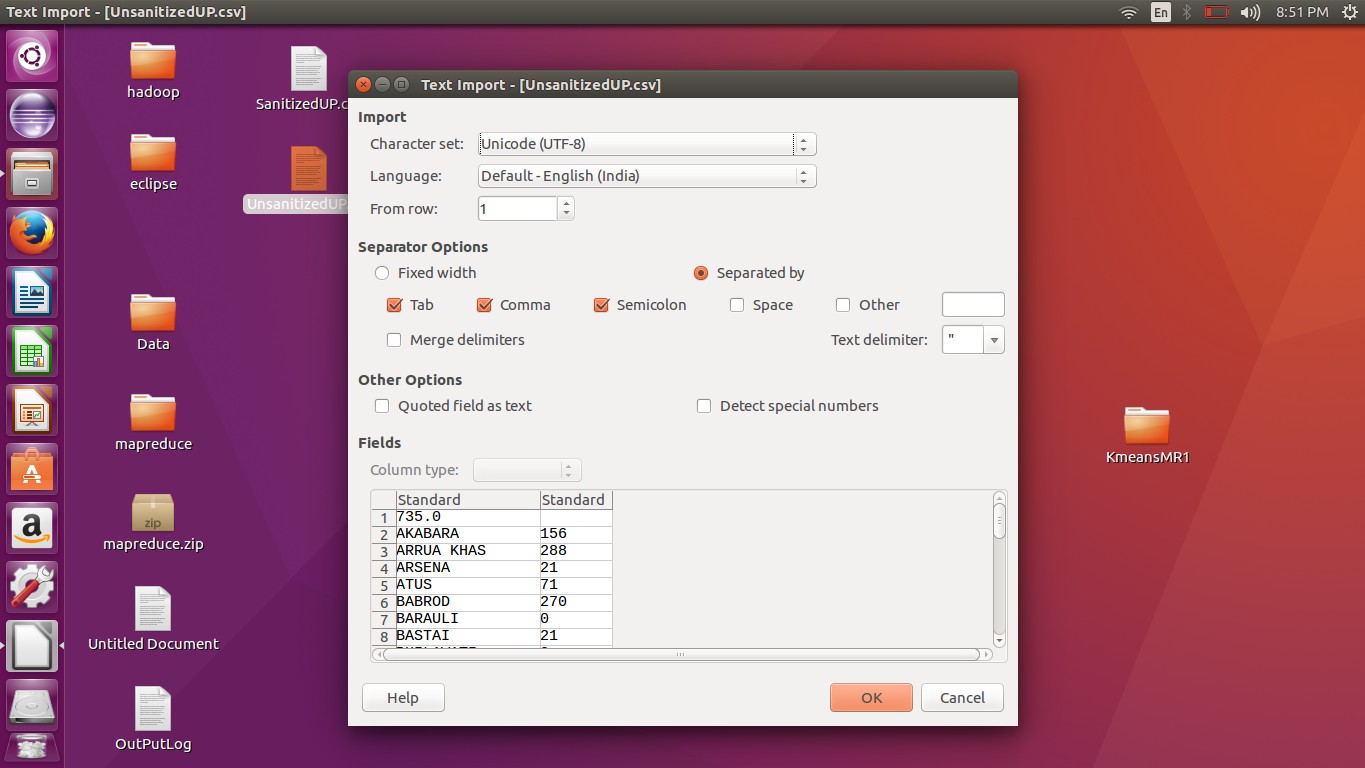
17/10/30 20:48:54 INFO mapreduce.KMeansClusteringJob: [80.20833333333333, 735.0]

### SanitizedOutput

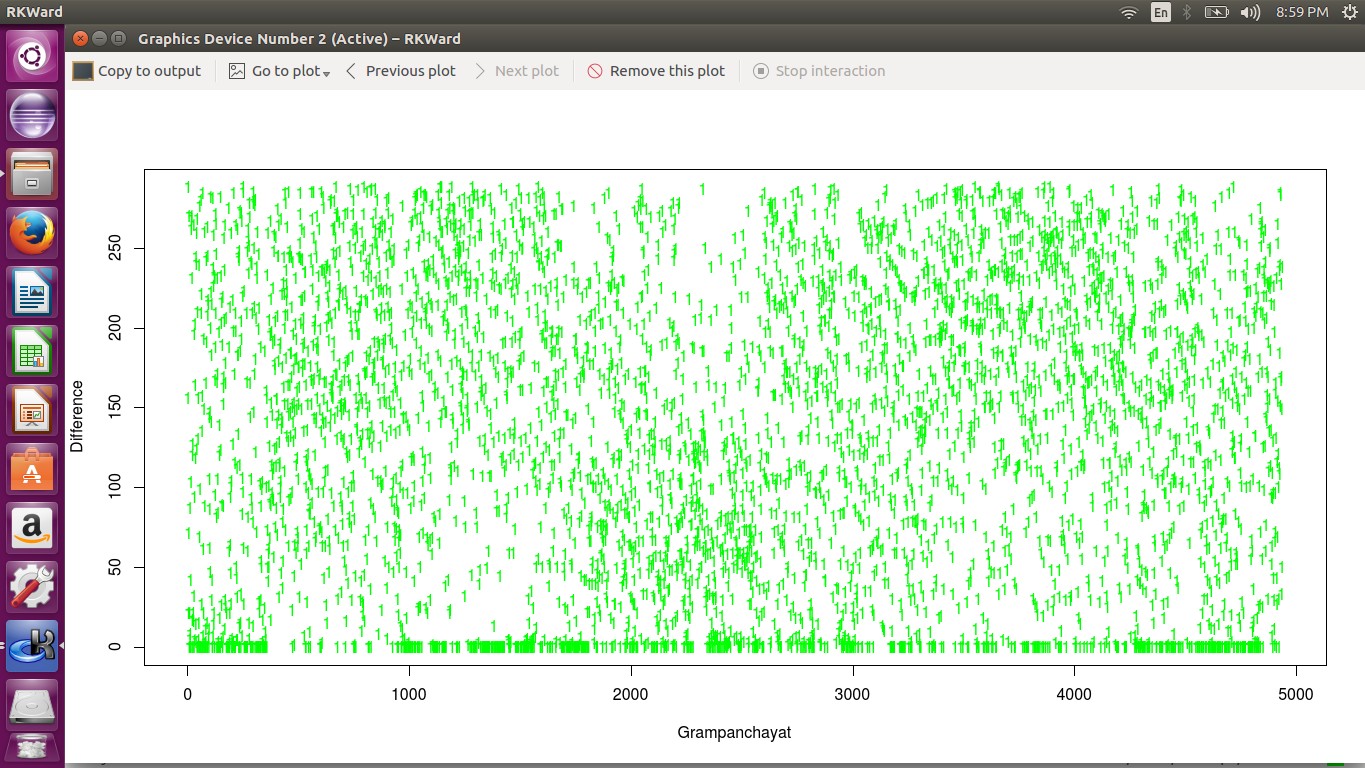
****

**23**

### UnsanitizedOutput

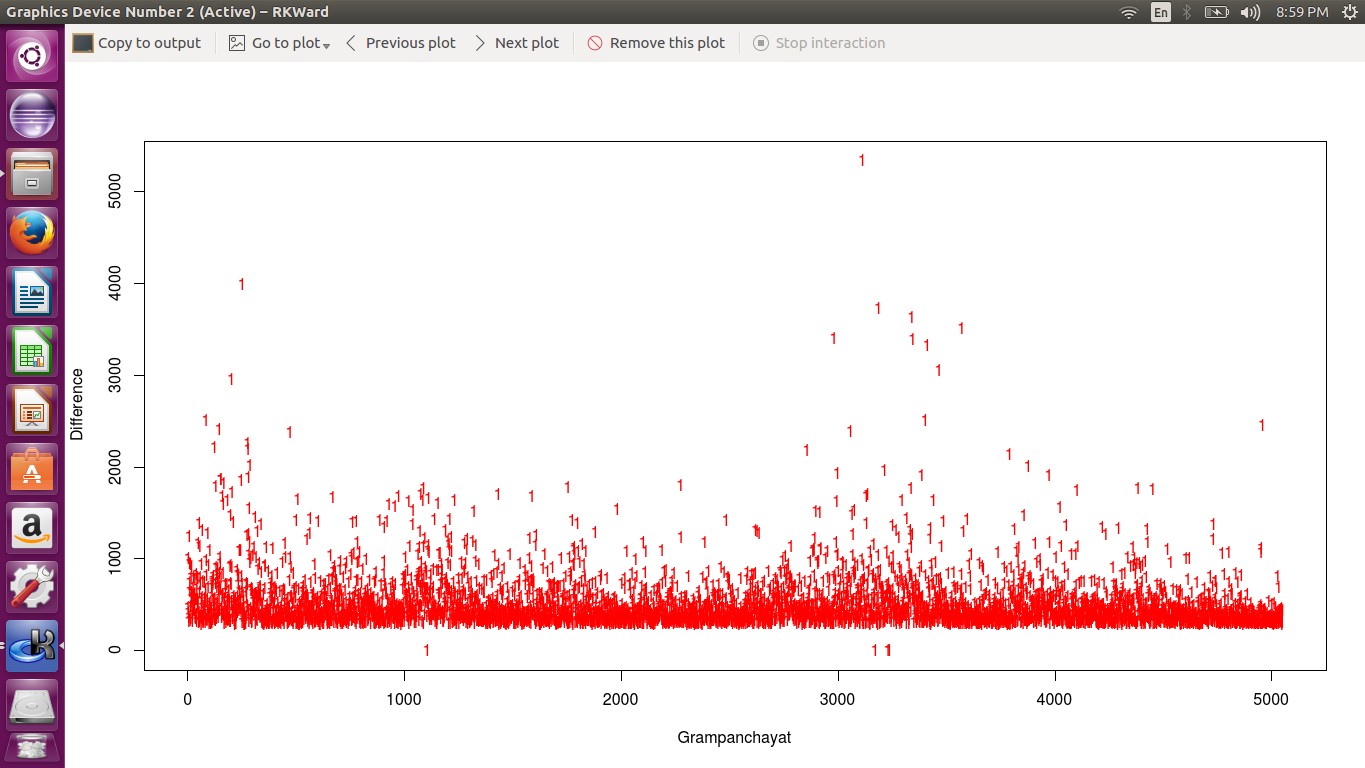
****

**Sanitised Results shown in R**

****

**24**

**Unsanitised Results shown in R**

****

**25**

# RESULT

The two clusters differentiate the gram panchayats on the basis of cleanliness. We get to know which areas are sanitized and which areas need to be worked on.