MAPPER CODE:-

```
#!/usr/bin/env python
import sys
import math

def euc_dis(x, m):
	dis = 0
	for i in range(len(x)):
		dis = dis + math.sqrt(math.pow(float(x[i]) - float(m[i]), 2))
	return dis

def get_nearest_cluster(centroid, x):
	closestTo = -1
	mindist = sys.maxint
	for j in range(0, len(centroid)):
		euc_distance = euc_dis(centroid[j], x)
		if(euc_distance < mindist):
			mindist = euc_distance
			closestTo = j
	return closestTo


def get_centroids():
	f = open("centroid.txt", 'r')
	data = f.read()
	centroid = []
	for i in data.strip().split("\n"):
		row = i.strip().split("\t")
		centroid.append(row)
	return centroid

centroids = get_centroids()

for line in sys.stdin:
	line = line.strip()
	data = line.split('\t')
	gen_id = data[0]
	ground_truth = data[1]
	attributes = data[2:]
	closestTo = get_nearest_cluster(centroids, attributes)
	attr_str = '#'.join(attributes)
	print '%s\t%s\t%s' % (closestTo, gen_id, attr_str)
```

REDUCER CODE :

```
#!/usr/bin/env python

import sys

def get_centroids(k):
	k = k - 1
	f = open("new_dataset1.txt", 'r')
	data = f.read()
	centroid = []
	for i in data.strip().split("\n"):
		row = i.strip().split("\t")
		centroid.append(row[2:])
		if(k == 0):
			return centroid
		k -= 1

centroids = get_centroids(4)

summ = [0]*len(centroids[0])
count = 0
cluster_points = []
last_cluster = None
current_cluster = None

open('centroid1.txt', 'w').close()

for line in sys.stdin:
	# remove leading and trailing whitespace
	line = line.strip()
	# parse the input we got from mapper.py
	current_cluster, gen_id, attributes = line.split('\t')
	attributes = attributes.split('#')

	if(last_cluster == current_cluster):
		count = count + 1
		cluster_points.append(gen_id)
		for i in range(0, len(attributes)):
			summ[i] = summ[i] + float(attributes[i])
	else:
		if(last_cluster):
			new_centroid = [x/float(count) for x in summ]
			cent = ""
			for i in new_centroid:
```

```
                        cent = cent + str(float("{0:.4f}".format(i))) + "\t"
                cent = cent + "\t\n"
                with open("centroid1.txt", "a") as myfile:
                        myfile.write('%s' % cent)
                print '%s\t%s\t' % (last_cluster, cluster_points)
                cluster_points = []
        summ = [0]*len(centroids[0])
        count = 1
        cluster_points.append(gen_id)
        for i in range(0, len(attributes)):
                summ[i] += float(attributes[i])
        last_cluster = current_cluster

if(last_cluster and last_cluster == current_cluster):
        new_centroid = [x/float(count) for x in summ]
        print '%s\t%s\t' % (last_cluster, cluster_points)
        cent = ""
        for i in new_centroid:
                cent = cent + str(float("{0:.4f}".format(i))) + "\t"
        cent = cent + "\t\n"
        with open("centroid1.txt", "a") as myfile:
                myfile.write('%s' % cent)
        cluster_points = []
```

BASH.py CODE :-

```
import os
import filecmp
import time
import numpy as np
import plotly
from plotly.graph_objs import *
from sklearn.decomposition import PCA as sklearnPCA

def preprocess(filename):
   inpdata = np.genfromtxt(filename,delimiter = '\t')
   X = np.loadtxt(filename,delimiter = '\t', usecols = range(2, inpdata.shape[1]), dtype =
'S15')
   gen_id = np.loadtxt(filename,delimiter = '\t', usecols = [0], dtype = 'S15')
   ground_truth = np.loadtxt(filename,delimiter = '\t', usecols = [1], dtype = 'S15')
   return X, gen_id, ground_truth
```

```
def runPCA(X):
        sklearn_pca = sklearnPCA(n_components=2)
        Y_sklearn = sklearn_pca.fit_transform(X)
        return Y_sklearn

#This function draws the scatter plot. plotting code taken from plot.ly website
def draw_scatter_plot(Y, labels):
  unique_labels = set(labels)
  points = []
  for name in unique_labels:
    x = []
    y = []
    for i in range(0, len(labels)):
      if(labels[i] == name):
        x.append(Y[i,0])
        y.append(Y[i,1])
    x = np.array(x)
    y = np.array(y)
    point = Scatter(
      x = x,
      y = y,
      mode='markers',
      name=name,
      marker=Marker(size=12, line=Line(color='rgba(217, 154, 217,
123)',width=0.5),opacity=0.9))
    points.append(point)
  data = Data(points)
  layout = Layout(xaxis=XAxis(title='PC1', showline=True),
          yaxis=YAxis(title='PC2', showline=True))
  fig = Figure(data=data, layout=layout)
  plotly.offline.plot(fig)

def get_centroids(ids):
        k = 1
        f = open("new_dataset1.txt", 'r')
        data = f.read()
        centroid = []
        for i in data.strip().split("\n"):
                if(k in ids):
                        row = i.strip().split("\t")
                        centroid.append(row[2:])
                k += 1
        return centroid

def calculateJackard(ground_truth, heirarical_ground_truth):
        m00 = 0
```

```
        m01 = 0
        m10 = 0
        m11 = 0
        for i in range(0, len(ground_truth)):
                for j in range(0, len(ground_truth)):
                        if((ground_truth[i] != ground_truth[j]) and (heirarical_ground_truth[i]
!= heirarical_ground_truth[j])):
                                m00 += 1
                        elif((ground_truth[i] == ground_truth[j]) and
(heirarical_ground_truth[i] != heirarical_ground_truth[j])):
                                m01 += 1
                        elif((ground_truth[i] != ground_truth[j]) and
(heirarical_ground_truth[i] == heirarical_ground_truth[j])):
                                m10 += 1
                        elif((ground_truth[i] == ground_truth[j]) and
(heirarical_ground_truth[i] == heirarical_ground_truth[j])):
                                m11 += 1
        jaccard = m11 / float(m11 + m10 + m01)
        rand = (m11 + m00) / float(m11 + m10 + m01 + m00)
        print(" Jaccard is : " + str(jaccard)),
        print(" Rand is : " + str(rand))

centroids = get_centroids([3, 5, 9])

total_lines = 0

with open('new_dataset1.txt') as f:
   total_lines = sum(1 for _ in f)

open('centroid.txt', 'w').close()
for c in centroids:
        cent = ""
        for i in c:
                cent = cent + str(i) + "\t"
        cent = cent + "\t\n"
        with open("centroid.txt", "a") as myfile:
                myfile.write('%s' % cent)

os.system("hdfs dfs -rm -r kmeansinp")
os.system("hdfs dfs -mkdir kmeansinp")
os.system("hdfs dfs -put $HOME/new_dataset1.txt kmeansinp")

start = time.time()
while True:
        os.system("hdfs dfs -rm -r kmeansout")
        os.system("hdfs dfs ")
```

```
        os.system("hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.6.4.jar -mapper $HOME/kmeansmap.py -reducer $HOME/kmeansred.py -input kmeansinp -output kmeansout")
        #os.system("cat cho.txt | python kmeansmap.py | sort | python kmeansred.py")
        #open('centroid.txt', 'w').close()
        if(filecmp.cmp('centroid.txt', 'centroid1.txt') == True):
                break
        with open("centroid1.txt") as f:
                lines = f.readlines()
                lines = [l for l in lines]
                with open("centroid.txt", "w") as f1:
                        f1.writelines(lines)

print("Time to run is : "),
print("--- %s seconds ---" % (time.time() - start))
os.system("rm -r part-00000")
os.system("rm -r _SUCCESS")
os.system("hdfs dfs -get kmeansout/*")
f = open("part-00000", 'r')
data = f.read()

gen_ids = [0]*total_lines

print(total_lines)
for i in data.strip().split("\n"):
        print(i)
        row = i.strip().split("\t")
        id_list = row[1][1:len(row[1])-1]
        id_list = id_list.replace(' ', '')
        id_list = id_list.replace("'", "")
        id_list = id_list.split(',')
        for k in id_list:
                gen_ids[int(k)-1] = row[0]

X, ids, ground_truth = preprocess("new_dataset1.txt")
Y_pca = runPCA(X)
calculateJackard(ground_truth, gen_ids)
draw_scatter_plot(Y_pca, gen_ids)
print(gen_ids)
```