**File Name: kmean.py**

**Python version: 2.7.x**

**Code description: This code contains K-Mean algorithm for clustering the data**

**NOTE: Possible indentation error could occur in python code because of difference of indentation width between text editor and the word document**

#!/usr/bin/python

import sys,csv,math,random,time

import numpy as np

import normalize

class Point:

def \_\_init\_\_(self, coordinates):

self.coordinates = coordinates

self.n = len(coordinates)

def \_\_repr\_\_(self):

return str(self.coordinates)

class Cluster:

def \_\_init\_\_(self, points):

if len(points) == 0: raise Exception("ERROR:The cluster cluster is empty")

self.points = points

self.n = points[0].n

for p in points:

if p.n != self.n: raise Exception("ERROR: uneven dimensions")

self.centroid = self.calculateCentroid()

def \_\_repr\_\_(self):

return str(self.points)

def updateCentroids(self, points):

old\_centroid = self.centroid

self.points = points

self.centroid = self.calculateCentroid()

return calcDistance(old\_centroid, self.centroid)

def calculateCentroid(self):

reduced\_coordinates = lambda i:reduce(lambda x,p : x + p.coordinates[i],self.points,0.0)

centroid\_coordinates = [reduced\_coordinates(i)/len(self.points) for i in range(self.n)]

return Point(centroid\_coordinates)

def kmeans(points, k, cutoff):

initial = random.sample(points, k)

clusters = [Cluster([p]) for p in initial]

total\_iteration = 0

while True:

total\_iteration = total\_iteration + 1

lists = [ [] for c in clusters]

for p in points:

smallest\_distance = calcDistance(p,clusters[0].centroid)

index = 0

for i in range(len(clusters[1:])):

distance = calcDistance(p, clusters[i+1].centroid)

if distance < smallest\_distance:

smallest\_distance = distance

index = i+1

lists[index].append(p)

biggest\_shift = 0.0

for i in range(len(clusters)):

shift = clusters[i].updateCentroids(lists[i])

biggest\_shift = max(biggest\_shift, shift)

if biggest\_shift < cutoff:

break

print "Total Iteration =" , total\_iteration

return clusters

def calcDistance(a, b):

if a.n != b.n: raise Exception("ERROR: unable to comparable points")

ret = reduce(lambda x,y: x + pow((a.coordinates[y]-b.coordinates[y]), 2),range(a.n),0.0)

return math.sqrt(ret)

def getinfo():

cr = csv.reader(open("normalized\_twitter\_dataset.csv","rb"))

return cr

def getDataFromFile(row):

return Point([np.float32(x) for x in row])

def main():

normalize.normalizeData('twitter\_dataset.csv')

cutoff = 0.5

k = int(raw\_input("Input Number of cluster:"))

start\_time = time.time()

reader = getinfo()

points = []

point\_counter = 0

for row in reader:

points.append(getDataFromFile(row[1:]))

point\_counter = point\_counter + 1

clusters = kmeans(points, k, cutoff)

#total\_points ,final\_centroids = clusters

cluster\_per = []

end\_time = time.time() - start\_time

for i,c in enumerate(clusters):

count = 0

for p in c.points:

print " Cluster: ",i + 1,"\t Point :", p

count = count + 1

print "no of instance for cluster" , i + 1 ,":" ,count

cluster\_per.append(np.float32(((np.float32(count) / np.float32(point\_counter)) \* 100)))

print "Time:" , end\_time, "Seconds"

print "Number of Clusters:" , k

for idx,cp in enumerate(cluster\_per):

print "cluster", idx + 1 , ":" , cp , "%"

if \_\_name\_\_ == "\_\_main\_\_":

main()

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**File Name: normalize.py**

**Python version: 2.7.x**

**Code description: This code contains algorithm to normalize the data**

**NOTE: Possible indentation error could occur in python code because of difference of indentation width between text editor and the word document**

#!/usr/bin/python

from numpy import vstack,array

from scipy.cluster.vq import kmeans, vq, whiten

import csv

def normalizeData(fileName):

data\_arr = []

meal\_name\_arr = []

with open(fileName, 'rb') as f:

reader = csv.reader(f)

for row in reader:

data\_arr.append([float(x) for x in row])

meal\_name\_arr.append([row[0]])

data = vstack( data\_arr )

meal\_name = vstack(meal\_name\_arr)

# normalization

data = whiten(data)

File = "normalized\_" + fileName

fo = open(File, "w")

for subd in data:

for idx, d in enumerate(subd):

if(idx < 7):

fo.write(str(d) + ',')

else:

fo.write(str(d))

fo.write('\n')

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**File Name: twitter.rb**

**Ruby version: 1.9.x**

**Code description: This code contains algorithm to fetch the twitter data and write it into a file**

**NOTE: This code contains authentication Token to my personal twitter data and it is not to be shared**

require "rubygems"

require "twitter"

# Get current year

time = Time.new

YEAR = time.year

# Define Active Period

PERIOD = 31

# Create file for writing info to

pFile = File.new("twitter\_dataset.csv", "a")

Twitter.configure do |config|

config.consumer\_key ="qW1qVVBo5z93rdago5PI7Q"

config.consumer\_secret = "xkEImuRh3WhOrp6q2cmirOKM2jdeTsl7nXZWvTtqI"

config.oauth\_token = "783801600-sQiavEawypG08bo8Qk63LlxG0adVuYuO2iDFChTX"

config.oauth\_token\_secret = "OjzseHQ7hMtHovPhSGbTlgtBAKACvcrZuYONDQBLpw"

end

# Header

puts "@#{ARGV[0]}'s Follower Information:"

Twitter.follower\_ids(ARGV[0]).ids[0..50].each{|a|

# Time Since Last 5 Tweets

tweet\_time\_since\_1 = 0 # Last Tweet

tweet\_time\_since\_2 = 0

tweet\_time\_since\_3 = 0

tweet\_time\_since\_4 = 0

tweet\_time\_since\_5 = 0

begin

# Counter

counter = 1

# User

user = Twitter.user(a)

# Print Screen Name

puts "Screen Name : " + user.screen\_name.to\_s + " (" + user.id.to\_s + ")"

# 1 Print Twitter Age

twitter\_age = user.created\_at.year

puts "User since : " + twitter\_age.to\_s + " [#{YEAR - twitter\_age} year(s)]"

# 2 Print Number of Followers

followers = user.followers\_count

puts "Followers : " + followers.to\_s

# 3 Print Number of Friends

friends = user.friends\_count

puts "Friends : " + friends.to\_s

# 4 Print Number of Tweets

tweet\_count = user.statuses\_count

puts "Tweet Count : " + tweet\_count.to\_s

# 5 Print Number of Tweets User has Favorited

favourites\_count = user.favourites\_count

puts "Favourite Count : " + favourites\_count.to\_s

# 6 Print Number of Lists Author Appears On

listed\_count = user.listed\_count

puts "Listed Count : " + listed\_count.to\_s

# 7 Last Five Tweet Timings

puts "Last 5 Tweets at:"

bool = false

Twitter.user\_timeline(user, :count => 5).each do |tweet|

time\_since = tweet.created\_at

bool = true

tweet\_text = tweet.text

# Calculate Date Since Last Tweet

time\_since\_day = time\_since.day

time\_since\_month = time\_since.month

time\_since\_year = time\_since.year

days\_since1 = (365 \* time\_since\_year + time\_since\_year/4 - time\_since\_year/100 + time\_since\_year/400 + time\_since\_day + (153 \* time\_since\_month + 8)/5)

days\_since2 = (365 \* time.year + time.year/4 - time.year/100 + time.year/400 + time.day + (153 \* time.month + 8)/5)

puts "#{time\_since} [#{days\_since2 - days\_since1} day(s)]"

puts tweet\_text

case counter

when 1

tweet\_time\_since\_1 = days\_since2 - days\_since1

when 2

tweet\_time\_since\_2 = days\_since2 - days\_since1

when 3

tweet\_time\_since\_3 = days\_since2 - days\_since1

when 4

tweet\_time\_since\_4 = days\_since2 - days\_since1

when 5

tweet\_time\_since\_5 = days\_since2 - days\_since1

end

counter += 1

end

end

# Inactive

if(bool == false)

pFile.syswrite("2,#{YEAR - twitter\_age},#{followers},#{friends},#{tweet\_count},#{favourites\_count},#{listed\_count},#{tweet\_time\_since\_1}\r\n")

puts "Status: INACTIVE"

# Active

elsif(PERIOD > tweet\_time\_since\_1)

pFile.syswrite("1,#{YEAR - twitter\_age},#{followers},#{friends},#{tweet\_count},#{favourites\_count},#{listed\_count},#{tweet\_time\_since\_1}\r\n")

puts "Status: ACTIVE"

# Inactive

else

pFile.syswrite("2,#{YEAR - twitter\_age},#{followers},#{friends},#{tweet\_count},#{favourites\_count},#{listed\_count},#{tweet\_time\_since\_1}\r\n")

puts "Status: INACTIVE"

end

puts "-----------------------------------------------------------------------------"

sleep 5

}

pFile.close