import pandas as pd

import sklearn

import numpy as np

from sklearn.cluster import KMeans

def normalize(d) :

''' df\_norm=(((d-d.mean())\*\*2)/d.shape[1])\*\*(1/2)

return df\_norm'''

x = d.values # returns a numpy array

min\_max\_scaler = sklearn.preprocessing.MinMaxScaler()

x\_scaled = min\_max\_scaler.fit\_transform(x)

df = pd.DataFrame(x\_scaled)

return df

'''data\_norm = df # Has training + test data frames combined to form single data frame

normalizer = StandardScaler()

data\_array = normalizer.fit\_transform(data\_norm.as\_matrix())

return pd.DataFrame(data\_array)'''

def mean\_vector(clusters,df) :

mv=[]

for i in range(0,len(clusters)):

denom=len(clusters[i])

sum=[0 for i in range(100)]

for j in range(0,denom):

temp=clusters[i]

templist=df.iloc[temp[j]].tolist()

sum=np.add(sum,templist)

sum=sum/denom

mv.append(sum)

sum=sum.tolist()

sum.clear()

return mv

def euclidean\_distance(train,test) :

train = np.asarray(train)

test = np.asarray(test)

temp=train-test

temp=[x\*\*2 for x in temp]

temp=np.divide(temp,len(temp))

dist=np.sum(temp)

dist=dist\*\*(1/2)

return dist

def cosine\_similarity(train,test):

train = np.asarray(train)

test = np.asarray(test)

temp=train \* test

d=np.sum(temp)

train = [x\*\*2 for x in train]

d1 = np.sum(train)

d1=d1\*\*(1/2)

test = [x\*\*2 for x in test]

d2=np.sum(test)

d2=d2\*\*(1/2)

return d/(d1 \* d2)

def pearson\_correlation(train,test) :

train = np.asarray(train)

test = np.asarray(test)

n=train.size

temp = train \* test

d=np.sum(temp)

d1=np.sum(train)

d2=np.sum(test)

numerator = n\*d-d1\*d2

train = [x \*\* 2 for x in train]

d11 = np.sum(train)

test = [x \*\* 2 for x in test]

d21 = np.sum(test)

denominator = (d11-(d1\*\*2))\*(d21-(d2\*\*2))

denominator =denominator \*\* 1/2

return numerator/denominator

def manhattan\_distance(train,test):

train = np.asarray(train)

test = np.asarray(test)

temp = train - test

temp=np.absolute(temp)

d=np.sum(temp)

return d

def jaccard\_coefficient(train,test ) :

return sklearn.metrics.jaccard\_similarity\_score(train,test)

def evaluate\_nearest(mv,test) :

min=0

flag=0

for i in range(0,len(mv)) :

temp = euclidean\_distance(mv[i],test)

if(min < temp ):

min = temp

flag=i

return mv[flag].tolist()

'''root mean square error'''

def rmse(test,rv):

temp=0

sum=0

n=len(rv)

for i in range(0,n):

temp=(rv[i]-test[i])\*\*2

temp=temp/n

sum=sum+temp

sum=sum\*\*1/2

return sum

'''mean absolute error'''

def mae(test,rv):

temp=0

sum=0

n=len(rv)

for i in range(0,n):

temp=(rv[i]-test[i])

temp=temp/n

sum=sum+temp

return abs(sum)

'''precision recall f-measure'''

def prfm(test,rv):

t=listtobinary(test)

r=listtobinary(rv)

tp=0

tn=0

fp=0

fn=0

n=len(t)

for i in range (0,n):

if(t[i]==1 and r[i]==1) :

tp=tp+1

elif(t[i]==1 and r[i]==0) :

fn=fn+1

elif (t[i] == 0 and r[i] == 1):

fp = fp + 1

elif (t[i] == 0 and r[i] == 0):

tn = tn + 1

precision = tp/(tp+fp)

recall = tp/(tp + fn)

fm= (2 \* precision \*recall)/(precision+recall)

prfm=[precision,recall,fm]

return prfm

def listtobinary(listpassed):

temp=listpassed

for i in range(0, len(temp)):

if (temp[i] != 0):

temp[i] = 1

return temp

def convert\_iu(testdf) :

n=testdf.shape[0]

testdfl=[]

dftemp=pd.read\_csv('IUMatrix\_sc10.csv')

dftemp.drop(dftemp.columns[0], axis=1, inplace=True)

for i in range (n):

sum = [0 for i in range(100)]

temp=testdf.iloc[i]

tempb=listtobinary(temp)

for j in range (len(temp)):

if(tempb[j]!= 0) :

ktemp=dftemp.iloc[j]

ktemp= [int(k) for k in ktemp]

sum=np.add(sum,ktemp)

testdfl.append(sum)

testdf2=pd.DataFrame(testdfl)

return testdf2

def evaluation\_metrics(recdf,testdf) :

return 0

df=pd.read\_csv('IUMatrix\_sc10.csv')

df=df.sample(frac=0.7)

print(df)

'''im adding a line here for top-n'''

dft=df.T

tsidcol = list(dft.columns.values)

#df.drop([0],0)

df.drop(df.columns[0], axis=1, inplace=True)

d=normalize(df)

print(d)

'''centroids=[d.iloc[i].tolist() for i in range (0,5)]

centroids=np.asarray(centroids)

print(centroids)'''

cluster\_count=167

row\_count=d.shape[0]

km = sklearn.cluster.KMeans(n\_clusters=cluster\_count)

km.fit(d)

# Get cluster assignment labels

labels =km.labels\_.tolist()

print(labels)

print(labels[0])

print(d.shape[1])

clusters=[[] for y in range(cluster\_count)]

for j in range(0,row\_count):

temp=labels[j]

clusters[temp].append(j)

print(clusters,sep='\n')

meanvectors=pd.DataFrame()

meanvectors = mean\_vector(clusters,d)

print(meanvectors)

sdf=pd.read\_csv('postmatriclistsc10.csv')

sdf=sdf.sort\_values(by=['count'],ascending = False)

columns = eval(input("enter columns"))

sdf=sdf.head(columns)

sidlist=sdf['songid'].tolist()

print(sidlist)

testdf=pd.read\_csv('UIMatrix\_sc10.csv')

testdf=testdf[testdf.columns.intersection(sidlist)]

testdf=testdf.\_get\_numeric\_data()

testdf=testdf.sample(frac=0.3)

testdf=convert\_iu(testdf)

rvdfl=[]

testdfl=[]

for i in range(testdf.shape[0]) :

test=testdf.iloc[i]

testdfl.append(test)

recommend\_vector=[]

recommend\_vector = evaluate\_nearest(meanvectors,test)

#print(recommend\_vector)

print(i)

rvdfl.append(recommend\_vector)

rvdf=pd.DataFrame(rvdfl)

testdf=pd.DataFrame(testdfl)

print('Recommended vector :')

print(rvdf)

'''rmse'''

sum=0

for i in range(testdf.shape[0]):

sum=sum+rmse(testdf.iloc[i],rvdf.iloc[i])

rmse=sum/testdf.shape[0]

print('rmse :',rmse)

'''mae'''

sum=0

for i in range(testdf.shape[0]):

sum=sum+mae(testdf.iloc[i],rvdf.iloc[i])

mae=sum/testdf.shape[0]

print('mae :',mae)

'''prfm'''

precisiont=0

recallt=0

fmt=0

for i in range(testdf.shape[0]):

templist=[]

templist=prfm(testdf.iloc[i],rvdf.iloc[i])

precisiont=precisiont+templist[0]

recallt = recallt + templist[1]

fmt = fmt + templist[2]

precision=precisiont/testdf.shape[0]

recall=recallt/testdf.shape[0]

fm=fmt/testdf.shape[0]

print("precision",precision)

print("recall",recall)

print("fmeasure",fm)