Το παρόν report αφορά στην επεξήγηση των μεθόδων και των αντίστοιχων αποτελέσματων της κάθε μεθόδου της άσκησης ξεχωριστά. Οι μέθοδοι που περιγράφονται παρακάτω εφαρμόστηκαν πάνω σε ένα πειραματικό σύνολο δεδομένων που αποτελείται από πραγματικές εικόνες ανθρώπων, το οποίο χρησιμοποιήθηκε ως αντικείμενο ενός διαγωνισμού στο περιβάλλον *Kaggle*.

Αρχικά πραγματοποιήθηκαν όλα τα απαραίτητα imports για την ανάπτυξη των μεθόδων και στη συνέχεια αρχικοποιήθηκαν οι κατάλληλες μεταβλητές. Πιο συγκεκριμένα, η μεταβλητή **dir_train_data** αποθηκεύει το directory στο οποίο είναι αποθηκευμένο τοπικά το σύνολο δεδομένων με τα πρόσωπα. Οι υπόλοιπες μεταβλητές είναι λίστες, οι οποίες θα χρησιμοποιηθούν για την μετέπειτα αποθήκευση τιμών.

jupyter Homework2_4259 Last Checkpoint: ένα λεπτό πριν (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
#Author Kosmas Apostolidis
#AM:4259
from PIL import Image
import os
import matplotlib.pyplot as plt
import pandas as pd
from random import randrange
from random import choice
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Input , Dense
from tensorflow.keras.models import Model
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import confusion_matrix
from scipy.spatial.distance import cdist
```

In [2]:
```
dir_train_data = r'C:\Users\cosma\Desktop\EVERYTHING\MachineLearning\ML2022 Homework2   4259\all_faces\train_data' #Local directo
faces_from_train_dataset = [] #A list with faces
y_true = [] #True labels
label = 0   #0...49

agglo_clust_accuracy_scores = []
agglo_clust_precision   = []
agglo_clust_recall      = []
agglo_clust_F_measures  = []
agglo_clust_purity_scores = []

#Distance metrics used to K-Means algorithm
distance_metrics = ['euclidean' , 'cosine']

list_accuracy_scores  = []
list_F_measures       = []
list_purity_scores    = []
list_precision_scores = []
list_recall_scores    = []
```

Στην ακόλουθη εικόνα υλοποιήθηκε μία συνάρτηση η οποία επιλέγει τυχαία από το σύνολο δεδομένων φωτογραφίες ενός τυχαίου ατόμου που επιλέχθηκε από το σύνολο των 4000 ξεχωριστών ατόμων. Η πρώτη παράμετρος **list_faces** είναι μία λίστα η οποία περιέχει όλες τις διαθέσιμες φωτογραφίες του ατόμου και η **rnd_person** είναι μία σταθερά η οποία έχει αποθηκευμένο το άτομο που επιλέχθηκε.

In [3]:
```
def get_face_from_train_dataset(list_faces,rnd_person):
    rnd_face   = randrange(50)   #Generate random integer between 0 and 49
    if rnd_face > len(list_faces) - 1:
        rnd_face           = randrange(len(list_faces) - 1)   #Generate random integer between 0 and len(person_imgs) - 1
        face               = list_faces[rnd_face] #Get the last image of tthe rnd person
        train_colored_face = np.asarray(Image.open(dir_train_data+"\\"+str(rnd_person)+"\\"+str(face)))
        rbg_to_grayscale   = np.mean(train_colored_face , axis = 2) #convert a color image 3D array to a grayscale 2D
        faces_from_train_dataset.append(rbg_to_grayscale) #Append the train_img to the list
        y_true.append(label)
    else:
        face               = list_faces[rnd_face] #Get the random image of the train dataset
        train_colored_face = np.asarray(Image.open(dir_train_data+"\\"+str(rnd_person)+"\\"+str(face)))
        rbg_to_grayscale   = np.mean(train_colored_face , axis = 2) #convert a color image 3D array to a grayscale 2D
        faces_from_train_dataset.append(rbg_to_grayscale) #Append the img to the list
        y_true.append(label)
```

Στην παρακάτω εικόνα απεικονίζεται η τυχαία επιλογή των 10 ατόμων του συνόλου και μετά για καθέναν από αυτούς λαμβάνονται τυχαία 50 εικόνες των προσώπων τους.

In [4]:
```
#Here i choose 10 random people from the dataset and for each one of them i choose 50 random pictures
#Get 50 random images of 10 random people
for i in range(10):
    counter     = 0
    person      = randrange(4000)     #Generate random integer between 0 and 3999
    list_faces_from_train_dataset = os.listdir(dir_train_data+"\\"+str(person))
    while counter < 50:
        get_face_from_train_dataset(list_faces_from_train_dataset,person)
        counter += 1
    label += 1
```

Ακόμη, στην ακόλουθη εικόνα απεικονίζεται η επιτυχημένη εφαρμογή της μεθόδου PCA(***Principal Component Analysis***) στο διάνυσμα διάστασης **500*4096**, με την διάσταση του διανύσματος να μειώνεται κάθε φορά σε **100** , **50** και **25** αντίστοιχα.

In [7]:
```
# Principal Component Analysis(PCA) finds the directions of maximum variance in high-dimensional data
# and projects it onto a new subspace with equal or fewer dimensions than the original one

train_pca_100 = PCA(n_components = 100) #Keep the first 100
pca_100_converted_training_data = train_pca_100.fit_transform(faces_centered_from_train_dataset)
print("The data reduced from "+str(faces_centered_from_train_dataset.shape)+" to "+str(pca_100_converted_training_data.shape))
```

```
The data reduced from (500, 4096) to (500, 100)
```

In [8]:
```
train_pca_50 = PCA(n_components = 50)
pca_50_converted_training_data = train_pca_50.fit_transform(faces_centered_from_train_dataset)
print("The data reduced from "+str(faces_centered_from_train_dataset.shape)+" to "+str(pca_50_converted_training_data.shape))
```

```
The data reduced from (500, 4096) to (500, 50)
```

In [9]:
```
train_pca_25 = PCA(n_components = 25)
pca_25_converted_training_data = train_pca_25.fit_transform(faces_centered_from_train_dataset)
print("The data reduced from "+str(faces_centered_from_train_dataset.shape)+" to "+str(pca_25_converted_training_data.shape))
```

```
The data reduced from (500, 4096) to (500, 25)
```

Στην συνέχεια, έγινε η υλοποίηση του αλγόριθμου ***K-Means***.Αναλυτικότερα, επιλέγονται τυχαία **k** δεδομένα τα οποία αποτελούν τα αρχικά κέντρα. Ύστερα, χρησιμοποιώντας κάποια μετρική απόστασης(**Euclidean** ή **Cosine**) στα δεδομένα εκπαίδευσης, υπολογίζεται η απόστασή τους από τα k αρχικά κέντρα, τοποθετώντας τελικά κάθε δεδομένο στο κοντινότερο κέντρο(***centroid***).Στη συνέχεια ενημερώνονται οι τοποθεσίες των θέσεων των κέντρων, λαμβάνοντας τον μέσο όρο των σημείων σε κάθε cluster. Η παραπάνω διαδικασία επαναλαμβάνεται έως ότου τα κέντρα των clusters σταματούν να ενημερώνονται.

In [11]:
```
#Defining kmeans function
def kmeans(x,k,distance_metric,no_of_iterations):
    idx = np.random.choice(len(x), k, replace=False)
    #Randomly choosing Centroids
    centroids = x[idx, :] #Step 1

    #finding the distance between centroids and all the data points
    distances = cdist(x, centroids ,distance_metric) #Step 2

    #Centroid with the minimum Distance
    points = np.array([np.argmin(i) for i in distances]) #Step 3

    #Repeating the above steps for a defined number of iterations
    #Step 4
    for _ in range(no_of_iterations):
        centroids = []
        for idx in range(k):
            #Updating Centroids by taking mean of Cluster it belongs to
            if np.all(x[points == idx]) == 0:
                temp_cent = 0
                centroids.append(temp_cent)
            else:
                temp_cent = x[points==idx].mean(axis=0)
                centroids.append(temp_cent)

        centroids = np.vstack(centroids) #Updated Centroids

        distances = cdist(x, centroids , distance_metric)
        points = np.array([np.argmin(i) for i in distances])

    return points
```

Στην ακόλουθη εικόνα, όπως υποδηλώνουν και τα ονόματα των μεθόδων, υλοποιήθηκαν τρεις μέθοδοι, όπου η πρώτη υπολογίζει το **purity_score** μεταξύ των πραγματικών κατηγοριών και των προβλέψεων του αλγορίθμου(K-Means ή Agglomerative Hierarchical Clustering στην περίπτωση της άσκησης), η δεύτερη υπολογίζει τα **True Positives** , **False Positives** , **True Negatives** και **False Positives** μέσω του **confusion_matrix** και η τρίτη, έχοντας πλέον υπολογίσει όλες τις κατάλληλες μεταβλητές, υπολογίζει το **accuracy** , **precision** , **recall** και **F-measure**.

In [12]:
```
#Implementation of purity_score function
def purity_score(cm , y_true, y_pred):
    # return purity score
    return np.sum(np.amax(cm, axis=0)) / np.sum(cm)
```

In [13]:
```
#A function that calculates the true positives , false positives , true negatives and false negatives
def calculate_TP_FP_TN_FN(cm):
    FP = cm.sum(axis = 0) - np.diag(cm) #False Positives
    FN = cm.sum(axis = 1) - np.diag(cm) #False Negatives
    TP = np.diag(cm) #True Positives
    TN = cm.sum() - (FP + FN + TP) #True Negatives

    total_TP = sum(TP) #Total True Positives
    total_TN = sum(TN) #Total True Negatives
    total_FP = sum(FP) #Total False Positives
    total_FN = sum(FN) #Total False Negatives

    return total_TP , total_FP , total_TN , total_FN
```

In [15]:
```
def calculate_accuracy_precision_recall_F_measure(TP,FP,TN,FN):
    F_measure , precision , recall , accuracy = 0 , 0 , 0 , 0
    a = 1
    precision  = TP / (TP + FP) #Calculation of precision score
    recall     = TP / (TP + FN) #Calculation of recall score
    F_measure  = (1 + a) / ( (1 / precision) + (a / recall) ) #Caclculation of F_measure score
    accuracy = (TP + TN) / (TP + TN + FP + FN)
    return accuracy , precision , recall , F_measure
```

Στη συνέχεια, εφαρμόστηκε ο αλγόριθμος **K-Means** στα δεδομένα εκπαίδευσης, τόσο με την ευκλείδεια όσο και με την συνημιτονοειδή μετρική, τα αποτελέσματα των οποίων απεικονίζονται στην ακόλουθη εικόνα.

```
K-means with Euclidean distance metric
dimension of data:(500, 100)
accuracy:0.822
precision:0.11
recall:0.11
F-measure:0.10999999999999999
purity:0.26

dimension of data:(500, 50)
accuracy:0.8208
precision:0.104
recall:0.104
F-measure:0.10400000000000001
purity:0.248

dimension of data:(500, 25)
accuracy:0.8168
precision:0.084
recall:0.084
F-measure:0.084
purity:0.258


K-means with Cosine distance metric
dimension of data:(500, 100)
accuracy:0.8216
precision:0.108
recall:0.108
F-measure:0.108
purity:0.274

dimension of data:(500, 50)
accuracy:0.8232
precision:0.116
recall:0.116
F-measure:0.11600000000000002
purity:0.264

dimension of data:(500, 25)
accuracy:0.8124
precision:0.062
recall:0.062
F-measure:0.062
purity:0.284
```

Αντίστοιχα για τον **Agglomerative Clustering** αλγόριθμο:

```
Agglomerative Clustering

dimension of data:(500, 100)
accuracy:0.822
precision:0.11
recall:0.11
F-measure:0.10999999999999999
purity:0.242

dimension of data:(500, 50)
accuracy:0.82
precision:0.1
recall:0.1
F-measure:0.1
purity:0.216

dimension of data:(500, 25)
accuracy:0.8224
precision:0.112
recall:0.112
F-measure:0.112
purity:0.234
```

Τέλος , τα αποτελέσματα από τους παραπάνω υπολογισμούς συμπληρώθηκαν και στον παρακάτω πίνακα.

Method	***dimension of data (M)***	***Purity***	***F-measure***
K-means (*Euclidean distance*)	*100*	*0.26*	*0.11*
	50	*0.248*	*0.104*
	25	*0.258*	*0.084*
K-means (*Cosine distance*)	*100*	*0.274*	*0.108*
	50	*0.264*	*0.116*
	25	*0.284*	*0.062*
Agglomerative Hierarchical Clustering	*100*	*0.242*	*0.101*
	50	*0.216*	*0.1*
	25	*0.234*	*0.112*