Φτιάξαμε τρία αρχεία για τον k means και άλλα τρία για τον agglomerative hierarchical clustering. Κάθε αρχείο στο όνομα του έχει και ένα αριθμό 100, 50 ή 25 που αναπαριστά την μείωση διάστασης των δεδομένων για M = 100, 50 ή 25.

Δεδομένα

Αρχικά κάναμε import τις βιβλιοθήκες που θα χρησιμοποιήσουμε. Στην συνέχεια διαβάζουμε τους φακέλους από τα train_data και ξεκινάμε την επεξεργασία τους. Τα αρχεία .jpg διαιρέθηκαν με το 255 για να έχουν τιμές από 0 μέχρι 1 και μετά από αυτό δεν κάναμε κάποια άλλη επεξεργασία καθώς αποτελούν εικόνες που φορτώνονται από φακέλους. Οι φάκελοι που βάλαμε είναι οι 280, 295, 472, 936, 2103, 2237, 3998, 18, 3479, 1234. Πριν τους επιλέξουμε, έγινε έλεγχος για το αν όντως περιέχουν τουλάχιστον 50 φωτογραφίες ανθρώπων. Οι εντολές που φαίνονται στην εικόνα από κάτω αποτελούν τον τρόπο διαχείρισης μας για τις εικόνες. Δηλαδή, με βάση τον σύνδεσμο που μας δώσατε, πήραμε την εντολή για το διάβασμα μιας εικόνας και το εφαρμόσαμε για κάθε μια ξεχωριστά (500 συνολικά) διαιρώντας την με το 255 και τις προσθέσαμε στο train_photos που κρατάει τις εικόνες για την εκπαίδευση-πρόβλεψη.

```
# file 280
photo1 = np.asarray(Image.open('../input/images/train_data/280/0184_03.jpg')) / 255
photo2 = np.asarray(Image.open('../input/images/train_data/280/0068_01.jpg')) / 255
photo3 = np.asarray(Image.open('../input/images/train_data/280/0094_01.jpg')) / 255
photo4 = np.asarray(Image.open('../input/images/train_data/280/0035_03.jpg')) / 255
photo5 = np.asarray(Image.open('../input/images/train_data/280/0171_01.jpg')) / 255
photo6 = np.asarray(Image.open('../input/images/train_data/280/0157_04.jpg')) / 255
photo7 = np.asarray(Image.open('../input/images/train_data/280/0064_01.jpg')) / 255
photo8 = np.asarray(Image.open('../input/images/train_data/280/0082_01.jpg')) / 255
photo9 = np.asarray(Image.open('../input/images/train_data/280/0116_01.jpg')) / 255
photo10 = np.asarray(Image.open('../input/images/train_data/280/0254_07.jpg')) / 255

train_photos.append(photo1)
train_photos.append(photo2)
train_photos.append(photo3)
train_photos.append(photo4)
train_photos.append(photo5)
train_photos.append(photo6)
train_photos.append(photo7)
train_photos.append(photo8)
train_photos.append(photo9)
train_photos.append(photo10)
```

Αλλάξαμε και το χρώμα κάθε εικόνας σε μονοχρωματικό χωρίς απώλεια πληροφορίας με βάση την σχέση που μας δώσατε και φαίνεται στον κώδικα από κάτω.

```
i = 0
while i < len(train_photos):
    red = train_photos[i][:, :, 0]
    green = train_photos[i][:, :, 1]
    blue = train_photos[i][:, :, 2]

    train_photos[i] = (
        0.299 * red
        + 0.587 * green
        + 0.114 * blue)
    i = i + 1
```

Γνωρίζοντας ότι δεν μας χρειάζονται οι test κατηγορίες στην εκπαίδευση και πρόβλεψη αλλά μόνο στον υπολογισμό των purity και f measure, έπρεπε να φτιάξουμε ένα δικό μας test με αριθμούς από το 0 μέχρι το 9 που δείχνουν την κατηγορία με συνολικά 500 γραμμές. Ο κώδικας φαίνεται παρακάτω και στο test προσθέτουμε λίστες που έχουν 50 αριθμούς από την ίδια κατηγορία.

```
test_photos = np.append(test_photos, (zero, one, two, three, four, five, six, seven, eight, nine)).astype(int)
```

Φορτώσαμε και αλλάξαμε την μορφή των δεδομένων για να τα έχουμε σε μορφή 500 γραμμών όπου κάθε γραμμή θα περιγράφεται από 4096 στήλες, δηλαδή ένα διάνυσμα (500, 4096) και φαίνεται από κάτω.

```
train_photos = np.array(train_photos)
train_photos = train_photos.reshape(train_photos.shape[0], train_photos.shape[1] * train_photos.shape[2])
```

Για τις φωτογραφίες που πήραμε έπρεπε να ανεβάσουμε το csv αρχείο του διαγωνισμού στο Kaggle με όνομα images και με τον σύνδεσμο που έχει κάθε photo παίρνουμε τις φωτογραφίες από τον κάθε φάκελο που έχει επιλεχθεί.

PCA Αλγόριθμος

Όπως βλέπουμε από κάτω, για την μείωση της διάστασης κάναμε τον PCA για όλες τις τιμές του M = 100, 50 και 25. Από την βιβλιοθήκη sklearn ο αλγόριθμος PCA δέχτηκε όρισμα τις εικόνες που διαβάσαμε με ονομασία train_photos. Το M είναι η διάσταση που θα έχουμε σε κάθε αρχείο.

```
M = 100
train_photos = decomposition.PCA(n_components = M).fit_transform(train_photos)
```

Kmeans Αλγόριθμος

Από την βιβλιοθήκη sklearn χρησιμοποιήσαμε την υλοποίηση του k means που δέχεται όρισμα το train_photos για εκπαίδευση και πρόβλεψη με αριθμό ομάδων ίσο με 10.

```
num = 10
cluster = cluster.KMeans(n_clusters = num).fit(train_photos).predict(train_photos)
```

Agglomerative hierarchical clustering Αλγόριθμος

Από την βιβλιοθήκη sklearn χρησιμοποιήσαμε την υλοποίηση του Agglomerative hierarchical clustering που δέχεται όρισμα το train_photos για εκπαίδευση και πρόβλεψη με αριθμό ομάδων ίσο με 10. Την στρατηγική την φτιάξαμε με το linkage.

```
num = 10
cluster = cluster.AgglomerativeClustering(linkage = 'ward', n_clusters = num).fit_predict(train_photos)
```

Τέλος εκτυπώνουμε τα δεδομένα με τον παρακάτω κώδικα καθώς εκτυπώνουμε τα purity και f measure όπου για το f measure δίνοντας στο average την τιμή micro υπολογίζει τα ζητούμενα από την άσκηση για τον υπολογισμό του.

```
print("\npurity: ", np.sum(np.amax(cm, axis=0)) / np.sum(cm))
print("f measure: ", metrics.f1_score(test_photos, cluster, average='micro'))
```

Method	*dimension of data (M)*	*Purity*	*F-measure*
K-means (*Euclidean distance*)	*100*	0.252	0.11200000000000002
	50	0.308	0.084
	25	0.252	0.072
K-means (*Cosine distance*)	*100*		
	50		
	25		
Agglomerative Hierarchical Clustering	*100*	0.282	0.062
	50	0.306	0.168
	25	0.258	0.10000000000000002

Η καλύτερη μέθοδος που προκύπτει από τον παραπάνω πίνακα αποτελεσμάτων είναι η Agglomerative hierarchical clustering με M = 50, καθώς έχει λίγο χειρότερο purity από τον k means για M = 50 αλλά έχει πολύ καλύτερο f measure.