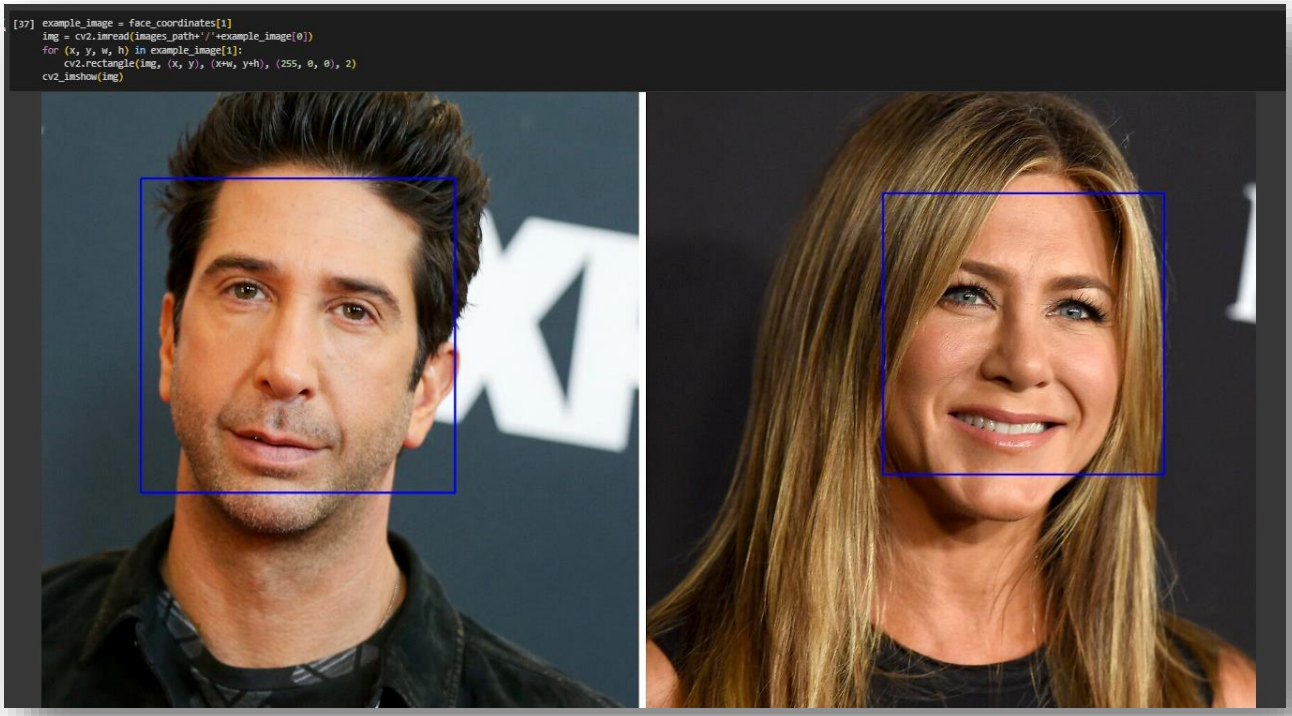
;208886333
;209299478

עבודה בעצי החלטה וניתוח אשכולות – עבודה 4

חלק א' – עיבוד מקדים

לאחר טעינת התמונות ומציאת הפרצופים בכל תמונה בעזרת האלגוריתם, שמרנו את הקואורדינטות של הפרצופים בכל תמונה ברשימה ייעודית ובחרנו תמונה אחת ולהלן התוצאות:



כעת נשמור בתיקייה נפרדת רק את הפרצופים עצמם ללא הרקע ונבדוק כמה פרצופים הצלחנו למצוא אל מול מס' התמונות המקורי (200) :

```
images_count = 0

for filename in os.listdir(images_path):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        images_count += 1

print("Number of images:", images_count)


faces_count = 0

for filename in os.listdir(output_folder):
    if filename.endswith('.jpg'):
        faces_count += 1

print("Number of faces:", faces_count)
```

```
Number of images: 200
Number of faces: 360
```

;208886333
;209299478

בעת נשתמש בface_recognition על מנת לחלץ פיצ'רים מכל פרצוף. האלגוריתם מחזיר וקטור בגודל 128 עבור כל פרצוף שזיהה.
לאחר מעבר על כל הפרצופים מתוך תיקיית faces הייעודית, נבדוק עבור אילו תמונות לא התקבל וקטור ואת אלו נמחק מהתיקייה מאחר והוא לא זיהה אותם כפרצופים.

לאחר הסינון נשארנו עם 266 פרצופים מתוך 360 שהיו בהתחלה.

```
faces_count = 0

for filename in os.listdir(output_folder):
    if filename.endswith('.jpg'):
        faces_count += 1

print("Number of faces after face recognition:", faces_count)
```

```
Number of faces after face recognition: 266
```

לבסוף שמרנו בDF את הפיצ'רים שחילצנו עבור כל פרצוף. לדוגמה:

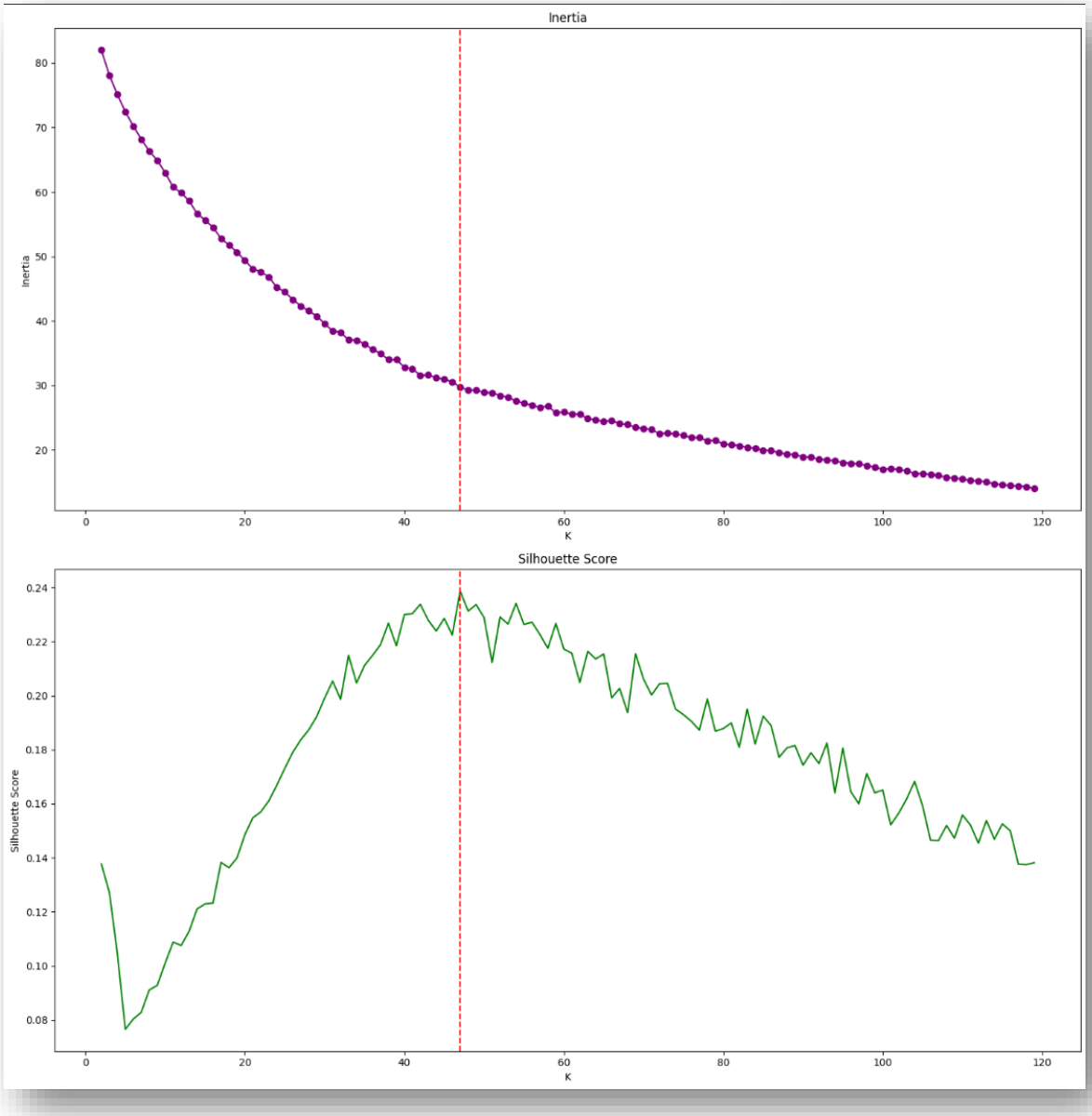
	Filename	FaceFeatures
0	image_18_face1.jpg	[-0.09191038459539413, 0.1404050886631012, 0.0...
1	image_5_face1.jpg	[-0.10219243168830872, 0.02683393657207489, 0....
2	image_12_face1.jpg	[-0.059195376932621, 0.07870084792375565, 0.10...
3	image_13_face1.jpg	[-0.03133663535118103, 0.10940520465373993, -0...
4	image_32_face1.jpg	[-0.12032586336135864, 0.06269033253192902, 0....
...	...	...
261	image_148_face1.jpg	[-0.10877057909965515, 0.1628803014755249, 0.1...
262	image_165_face1.jpg	[0.0030873119831085205, 0.1246829479932785, 0....
263	image_165_face2.jpg	[-0.046106331050395966, 0.14375776052474976, 0...
264	image_168_face1.jpg	[-0.13206258416175842, 0.2580793797969818, -0....
265	image_170_face1.jpg	[-0.15177203714847565, 0.14412905275821686, 0....

;208886333
;209299478

חלק ב' – מציאת מספר אשכולות אופטימלי

אנחנו בחרנו בטווח עבור K של (2,120) משום שאנו מבינות שיש לפחות 2 פרצופים בסך הכל (ניתן לראות זאת מהתמונה שבחרנו להציג בשלב הקודם) וכן מאחר ויש עוד 266 פרצופים נוספים אנחנו מניחות שאו שפרצוף יופיע פעם אחת או פעמיים לפחות ולכן בחרנו לקחת בערך את החצי מ260 ובחרנו ב-120.

אימנו את המודל של KMeans עבור כל K כזה בטווח שלנו, וחישבנו מדדי silhouette וinertia עבור כל אחד. שמרנו את הערכים עבור כל K בDF. (הקו האדום בגרף הנ"ל מסמן את ערך הK הנבחר)



על מנת לדעת מי הוא הK האופטימלי יש להתחשב בשני המדדים.

מדד Silhouette הוא מדד המשמש להערכת איכות תוצאות קלאסטרינג. המדד בודק כמה נקודה דומה לנקודות שאיתה באותו הקלאסטר ביחס לאשכולות אחרים. המדד נע בין (1-) ל-1 כאשר 1 הוא הטוב ביותר, כלומר אנחנו מחפשים את הערך הגבוה ביותר במדד זה. ערך גבוה מעיד על קרבה טובה לנקודות באותו האשכול וקרבה נמוכה לנקודות באשכולות אחרים. כלומר הוא מעיד על הפרדה טובה.

מדד Inertia הוא מדד המשמש להערכת איכות החלוקה ובודק האם הנקודות סווגו לאשכול בו הן קרובות לנק' האחרות בו ורחוקות מהנקודות באשכולות האחרים. המדד מחשב את סכום המרחקים בריבוע בין כל דגימה באשכול למרכז שלו. המטה

;208886333
;209299478

```python
best_k = results_df['K'].iloc[results_df['Silhouette Score'].idxmax()]
best_score = results_df['Silhouette Score'].max()

print("Best K: ", best_k)
print("Highest Silhouette Score: ", best_score)
```

```
Best K:  47
Highest Silhouette Score:  0.23868505388415515
```

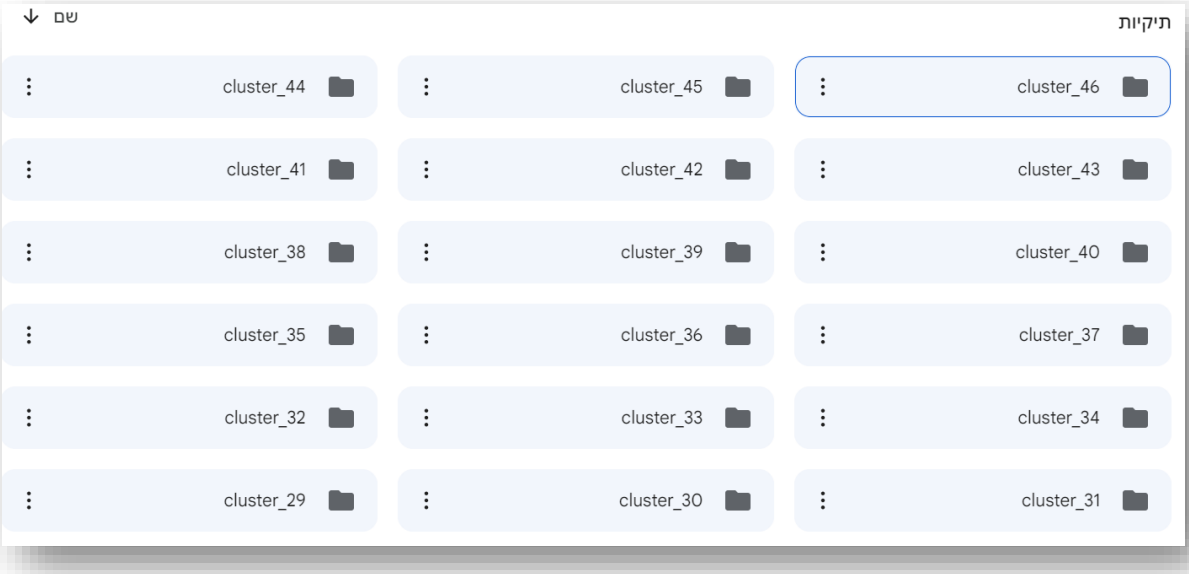
למזער את הערך. במדד זה אנחנו נסתכל על הגרף המייצג אותו ונחפש את ה"מרפק" אשר מעיד על כך שמאותה נקודה והלאה הערכים לא משתנים בצורה מהותית וזוהי הנק' הטובה לבחור את מספר האשכולות.

כלומר, לצורך בחירת K נסתכל תחילה מי הוא הK שנתן את מדד הSilhouette הגבוה ביותר. נמצא שהוא K=47.
לאחר מכן נתבונן בגרף של הinertia וניתן לראות (עפ"י שיטת elbow) שישנו "מרפק" באזור ה40+ .
מאחר והK הטוב ביותר עבור המדד הראשון הוא 47, וערך זה לא רחוק מאוד מה-40+ בחרנו לקחת את ערך זה.

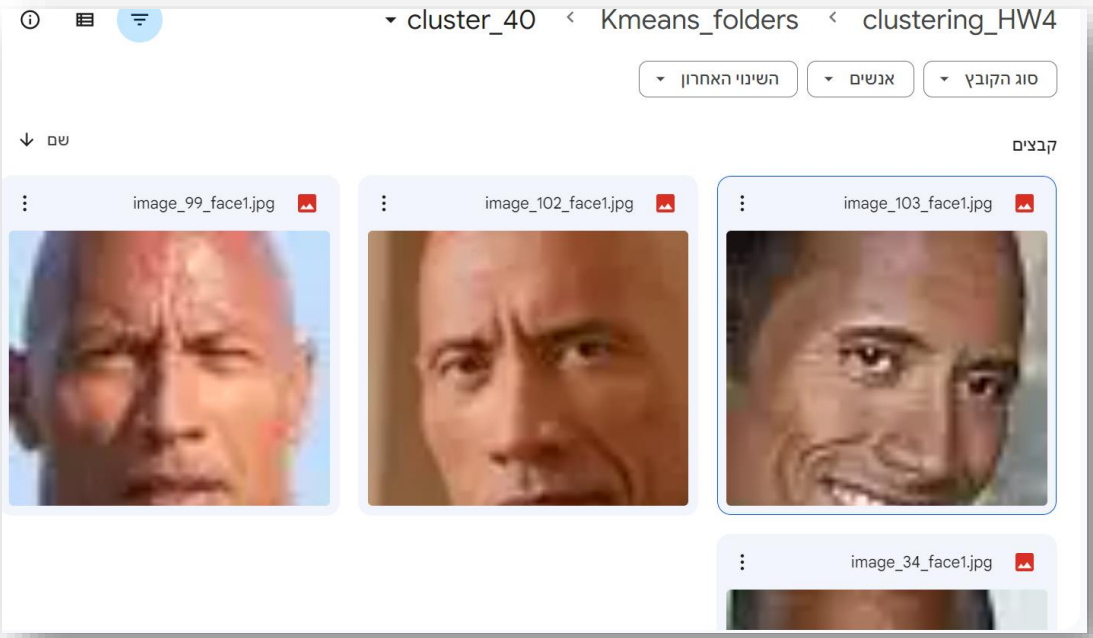
חלק ג' – בניית מודל עם הK הנבחר

יצרנו מודל עם הK הנבחר, יצרנו תיקיית יעד בשם Kmeans_folder אשר תכיל בתוכה את כל התיקיות ובכל תת תיקייה יהיו כל הפרצופים המשויכים לאותו הקלאסטר.

מאחר וK=47 אז יש לנו 47 תיקיות (מתחיל מאינדקס 0 ולכן מגיע עד 46).

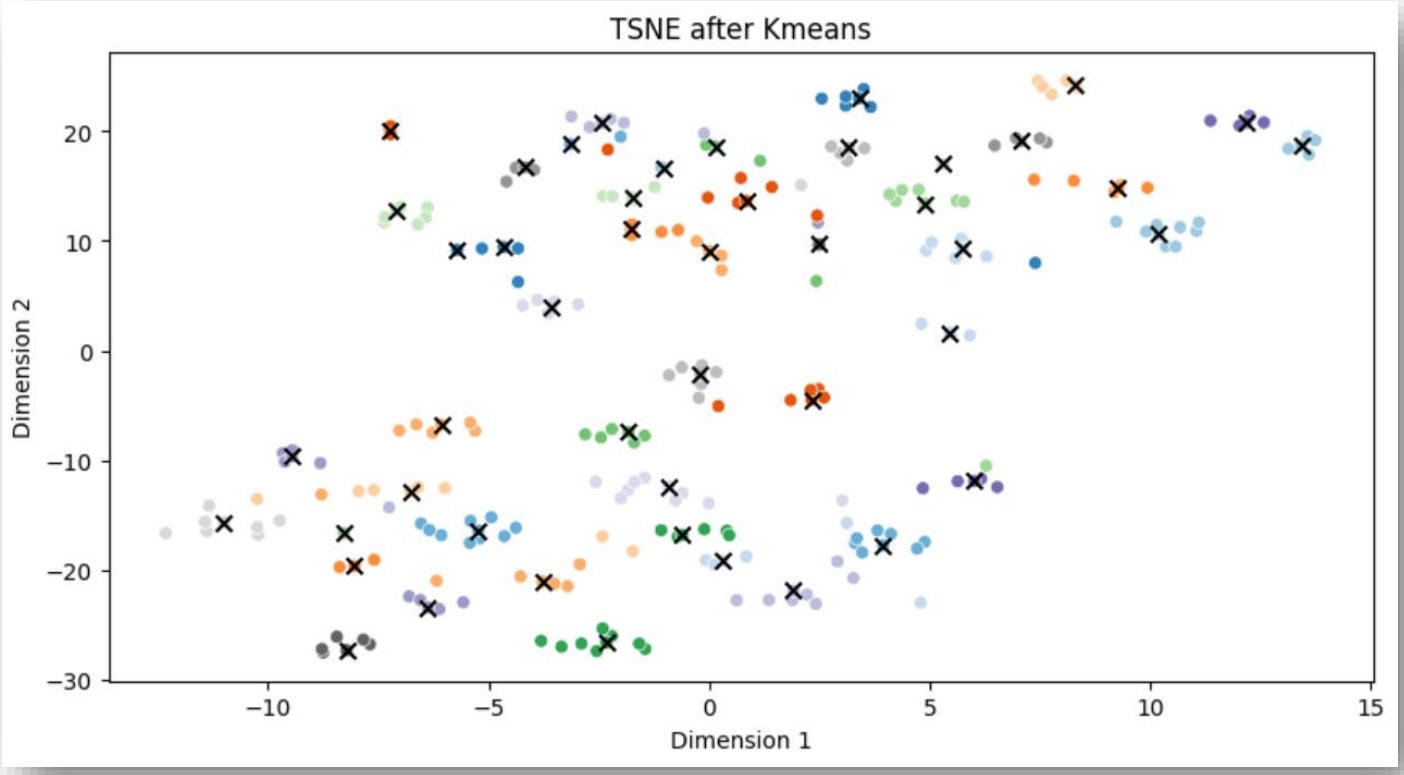


להלן דוגמה מאחת התיקיות:



;208886333
;209299478
לאחר מכן ביצענו הורדת ממדים באמצעות TSNE ל-2 ממדים, והיינו צריכות לחשב מרכזים (medoids) מחדש בהתאם לתוצאות שהתקבלו מתוך הורדת הממדים.

להלן התוצאות:

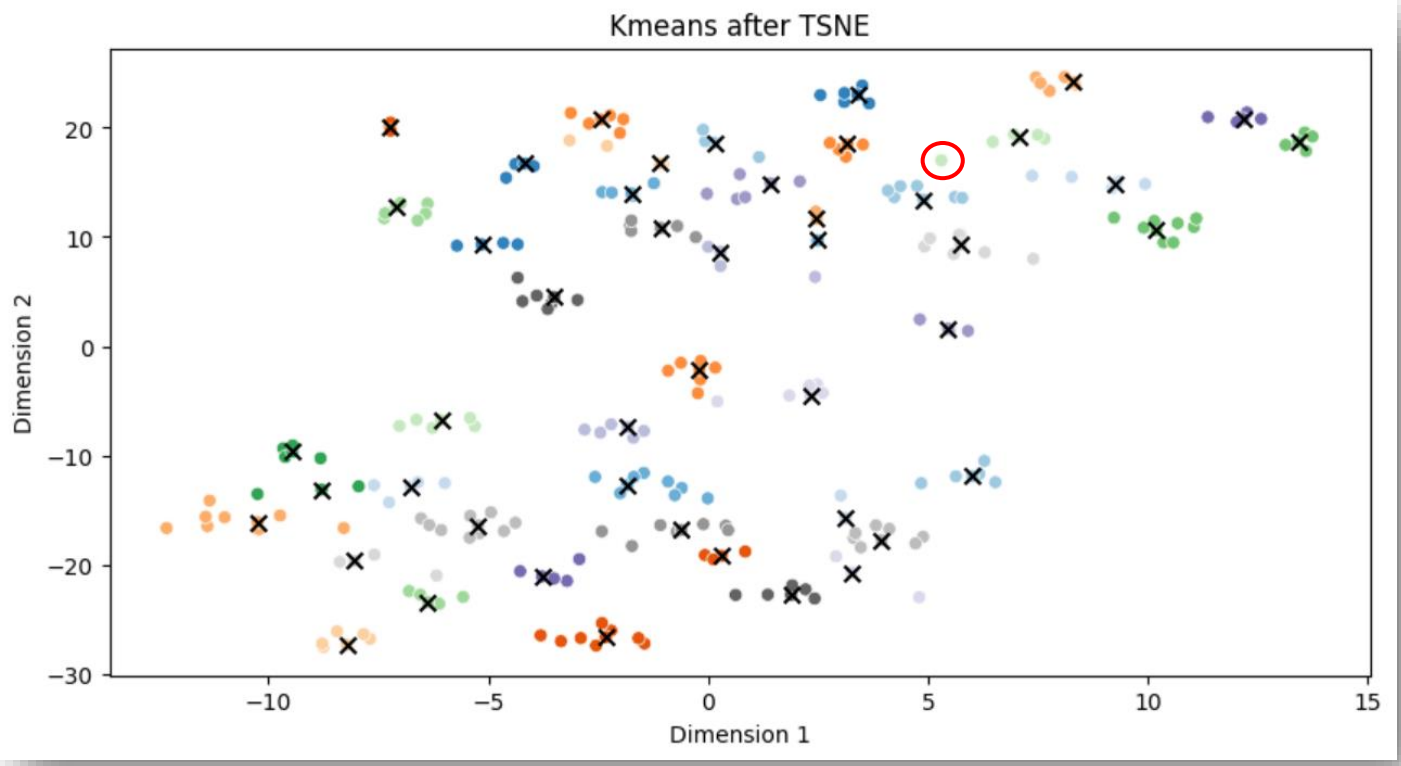


ניתן לראות שבאופן יחסי האלגוריתם הצליח לזהות את המרכזים טוב וניתן להבחין במרכזים השונים (אמנם יש 47 מרכזים ולכן הצבעים חוזרים על עצמם אך ניתן להבחין מתי מתחיל ונגמר קלאסטר מסוים).

;208886333
;209299478

חלק ד' – Kmeans לאחר הורדת הממדים

בשלב זה ניקח את הדאטה שיצא מתוך הורדת הממדים באמצעות TSNE ונכניס אותו לKmeans עם K=47.



ניתן לראות שהמרכזים נמצאים במיקומים טובים יותר ממקודם, כמו כן ניתן להבחין בשיפור בחלוקה לקלאסטרים השונים.

למשל, הנקודה המסומנת באדום הייתה קודם לכן מרכז של קלאסטר אחר וכעת היא נמצאה שייכת לקלאסטר שיש בו מרכז אחר.

כלומר ניתן להבחין בשיפור בביצועי המודל כאשר מריצים אותו לאחר הורדת הממדים.

;208886333
;209299478

חלק ה' – DBSCAN לאחר TSNE

נרצה לבנות מודל DBSCAN ולשם כך נצטרך לבחור את הערכים עבור הפרמטרים eps, min samples.

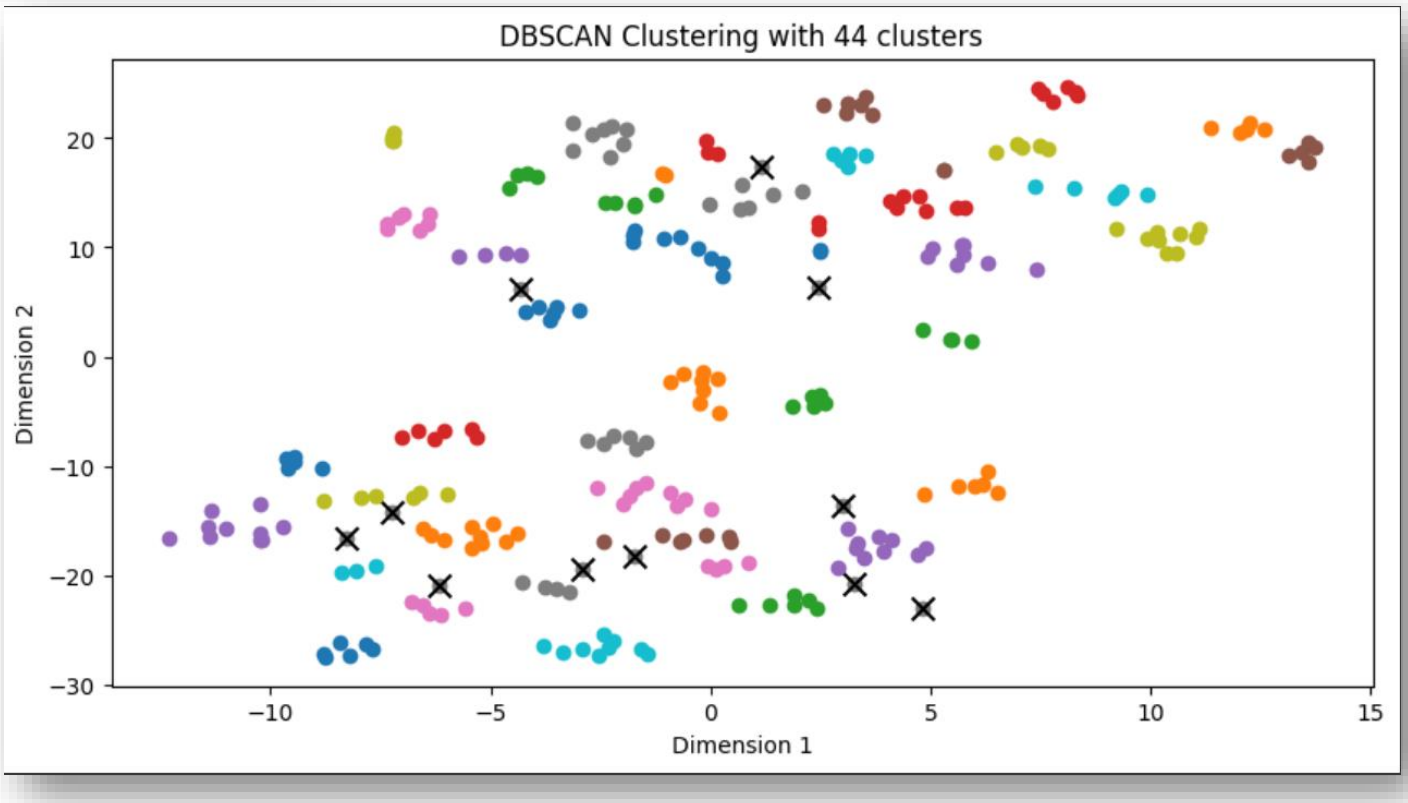
על מנת להבין אילו פרמטרים הכי טובים לנו בחרנו מספר ערכים עבור שניהם ובדקנו את התוצאות:

eps מייצג את המרחק המקסימלי בין 2 נקודות כך שהן עדיין ייחשבו תחת אותו הקלאסטר.
min samples מייצג את מינימום כמות נקודות שצריך לקלאסטר.

ראינו מקודם בKmeans שיצא לנו ש-K=47 הוא הטוב ביותר, אנחנו מניחות שהתוצאה כאן לא צריכה להיות רחוקה מזה ולכן נלך על הטווח של בין 40 ל-50 בכמות הקלאסטרים.
בנוסף, נרצה כמה שפחות נקודות רעש.

בחרנו בפרמטרים : eps = 1.5, min samples = 2 אשר נותן 44 קלאסטרים ו11 נקודות רועשות.

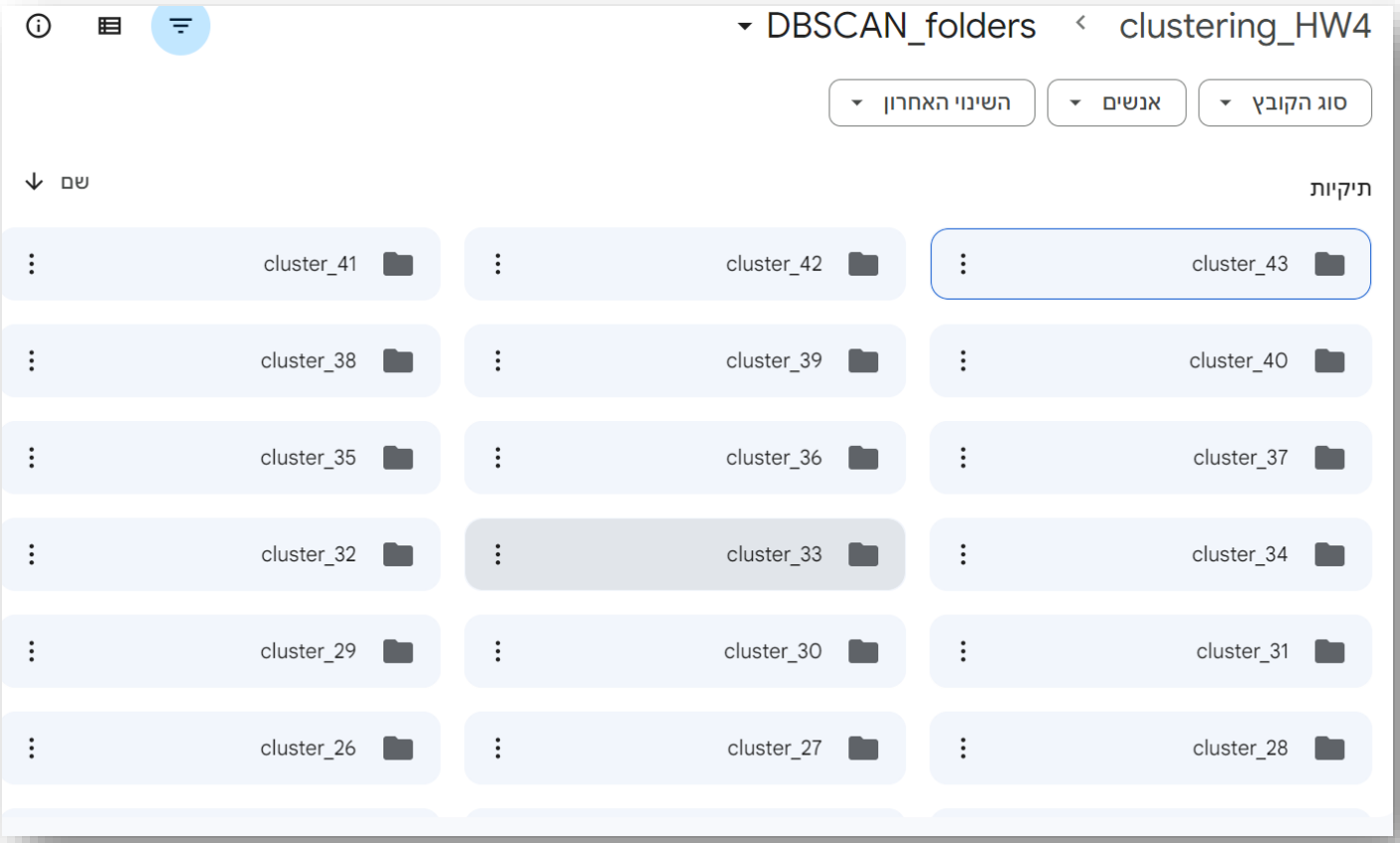
```
with eps = 0.5 and min samples =  2
number of clusters: 55
number of noise points:  136
with eps = 0.5 and min samples =  3
number of clusters: 16
number of noise points:  214
with eps = 0.5 and min samples =  5
number of clusters: 0
number of noise points:  266
with eps = 1.0 and min samples =  2
number of clusters: 50
number of noise points:  24
with eps = 1.0 and min samples =  3
number of clusters: 40
number of noise points:  44
with eps = 1.0 and min samples =  5
number of clusters: 25
number of noise points:  118
with eps = 1.5 and min samples =  2
number of clusters: 44
number of noise points:  11
with eps = 1.5 and min samples =  3
number of clusters: 40
number of noise points:  19
with eps = 1.5 and min samples =  5
number of clusters: 31
number of noise points:  58
with eps = 2 and min samples =  2
number of clusters: 30
number of noise points:  3
with eps = 2 and min samples =  3
number of clusters: 29
number of noise points:  5
with eps = 2 and min samples =  5
number of clusters: 27
number of noise points:  29
```



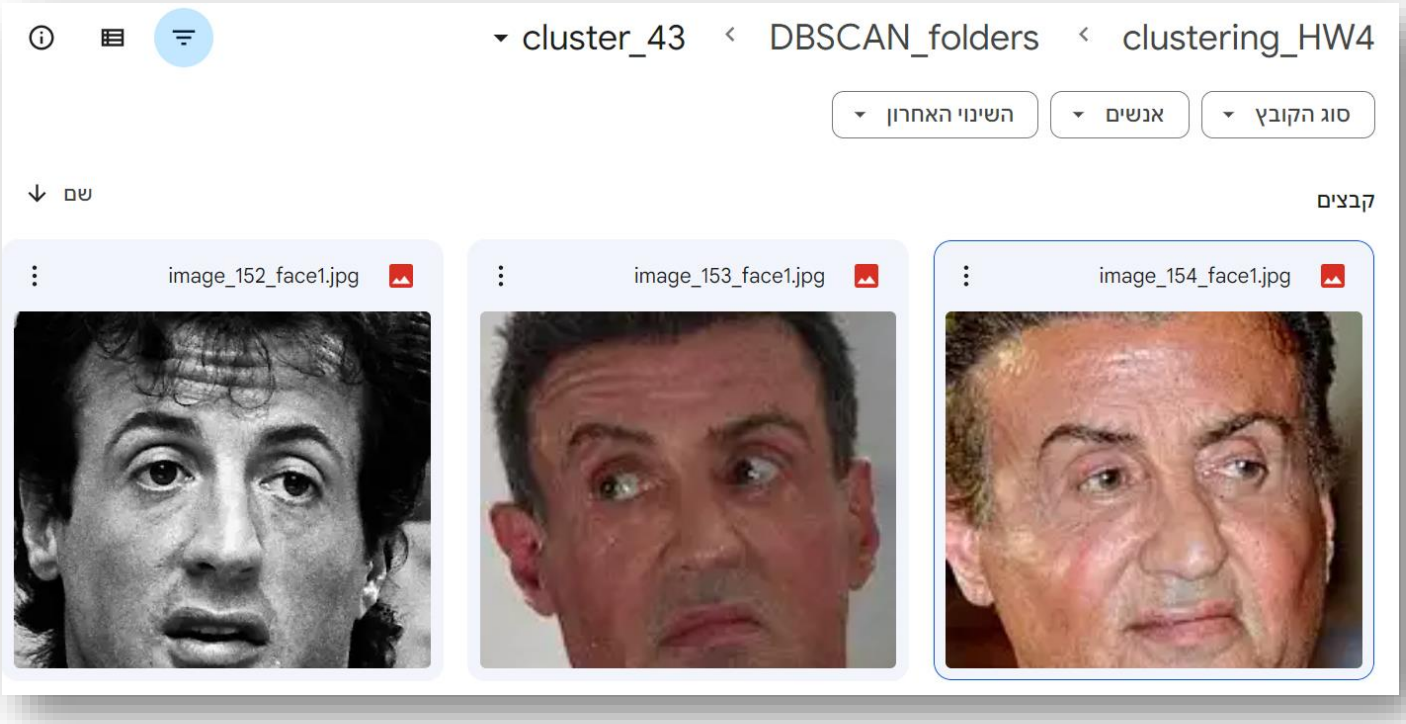
ניתן לראות שבאמת כפי שהגדרנו את הפרמטרים שלנו, הנקודות שויכו באופן טוב לחלוקות השונות, המרחקים בין הנקודות קצרים וניתן לראות שהאלגוריתם הצליח לחלק את הנקודות לנקודות שאכן קרובות אליהן. בנוסף אין הרבה נקודות רעש ואם

;208886333
;209299478
נסתכל בגרף מהסעיף הקודם נוכל לראות שבאמת הנקודות שסומנו כאן בנקודות רעש נראות בנקודות רעש גם שם, כלומר האלגוריתם DBSCAN זיהה אותן בצורה טובה.
ניתן להבחין בצורה קלה לעין בקלאסטרים השונים.

הכנסנו כל קלאסטר לתיקייה משלו: (44 קלאסטרים, מתחיל מאינדקס 0 ולכן עד 43)



דוגמה של קלאסטר אחד:



;208886333
;209299478

חלק ו' – הערכת ביצועים

1. מודל Kmeans על הנתונים המקוריים VS מודל Kmeans על הנתונים לאחר TSNE :

```
Kmeans before TSNE:
Silhouette score:  0.23868505388415515

Kmeans after TSNE:
Silhouette score:  0.61853856
```

בחרנו במדד Silhouette משום שערך גבוה בו מייצג שהחלוקות מובהקות יותר וברורות יותר, יש הפרדה טובה יותר.
אם נראה שיש עליה במדד לאחר הורדת הממדים נוכל להגיד שההורדה שיפרה את יכולת החלוקה לאשכולות.
אכן, ניתן לראות שהמדד עלה.
לכן עבור השוואה זו נגיד שהמודל של Kmeans לאחר הורדת הממדים באמצעות TSNE הוא טוב יותר.

2. מודל Kmeans על הנתונים לאחר הורדת ממדים VS מודל DBSCAN :

```
Kmeans after TSNE:
Silhouette score:  0.61853856

DBSCAN:
Silhouette score:  0.6048911
```

ניתן לראות שהפעם התוצאות קרובות. גם הפעם בחרנו במדד Silhouette מאותו אופן שפירטנו תחת הסעיף הקודם.

למרות שהם צמודים אלגוריתם Kmeans ניצח.

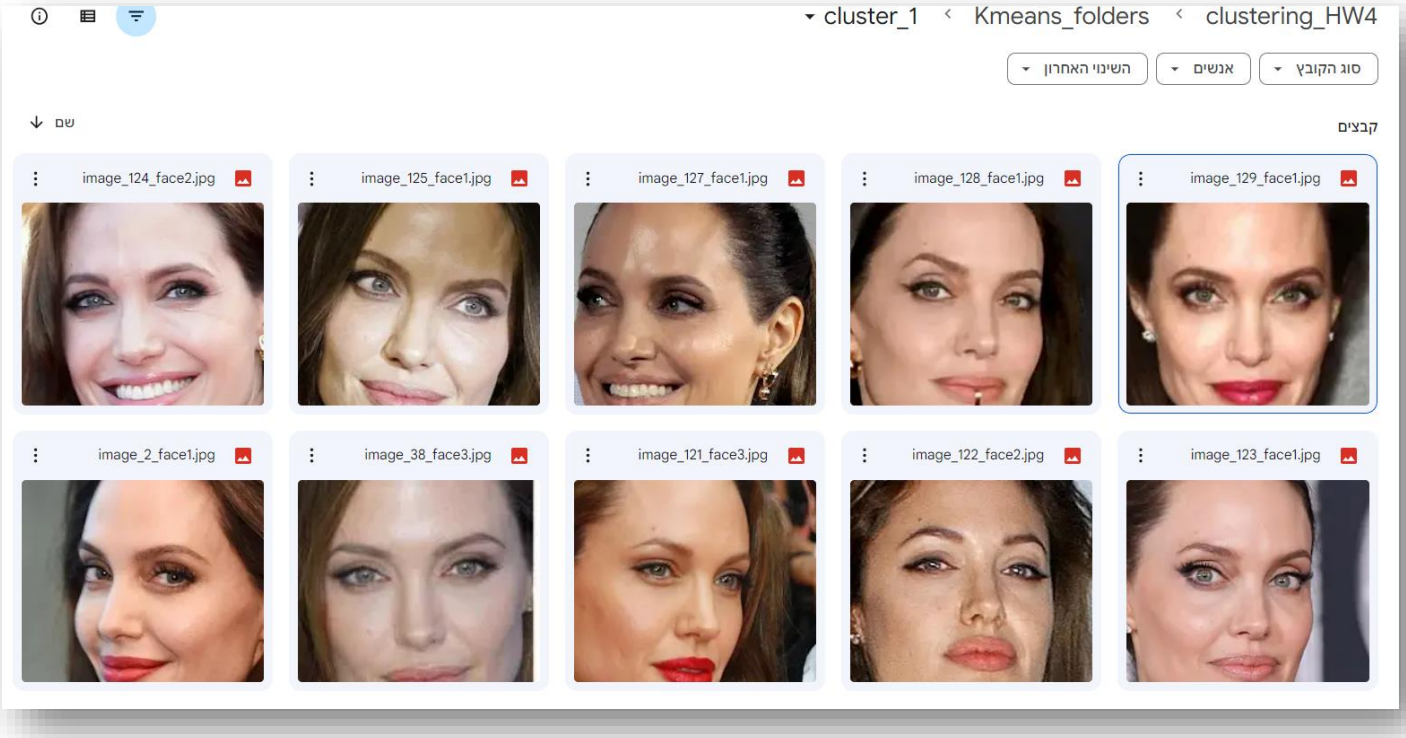
מבין כל ההשוואות, המודל שקיבל את הציון הנמוך ביותר הוא מודל הKmeans לפני הורדת הממדים והמודל שניצח הוא מודל הKmeans לאחר הורדת הממדים.

ניתן להסיק מכך שמודל זה מבצע הפרדה טובה יותר של האשכולות כך שהם צפופים יותר ובמקביל מופרדים יותר מהאחרים.

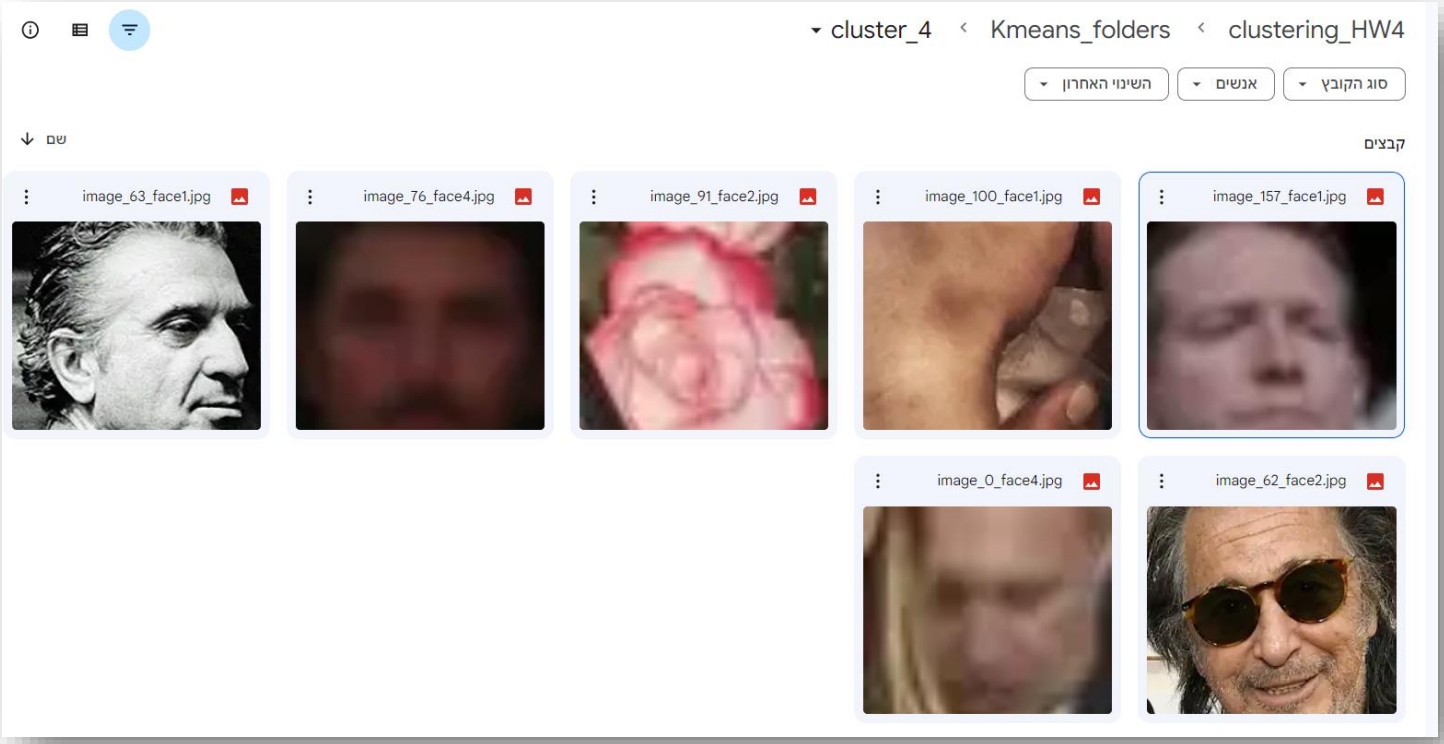
;208886333
;209299478

עבור Kmeans על הנתונים המקוריים :

1. חלוקה טובה:



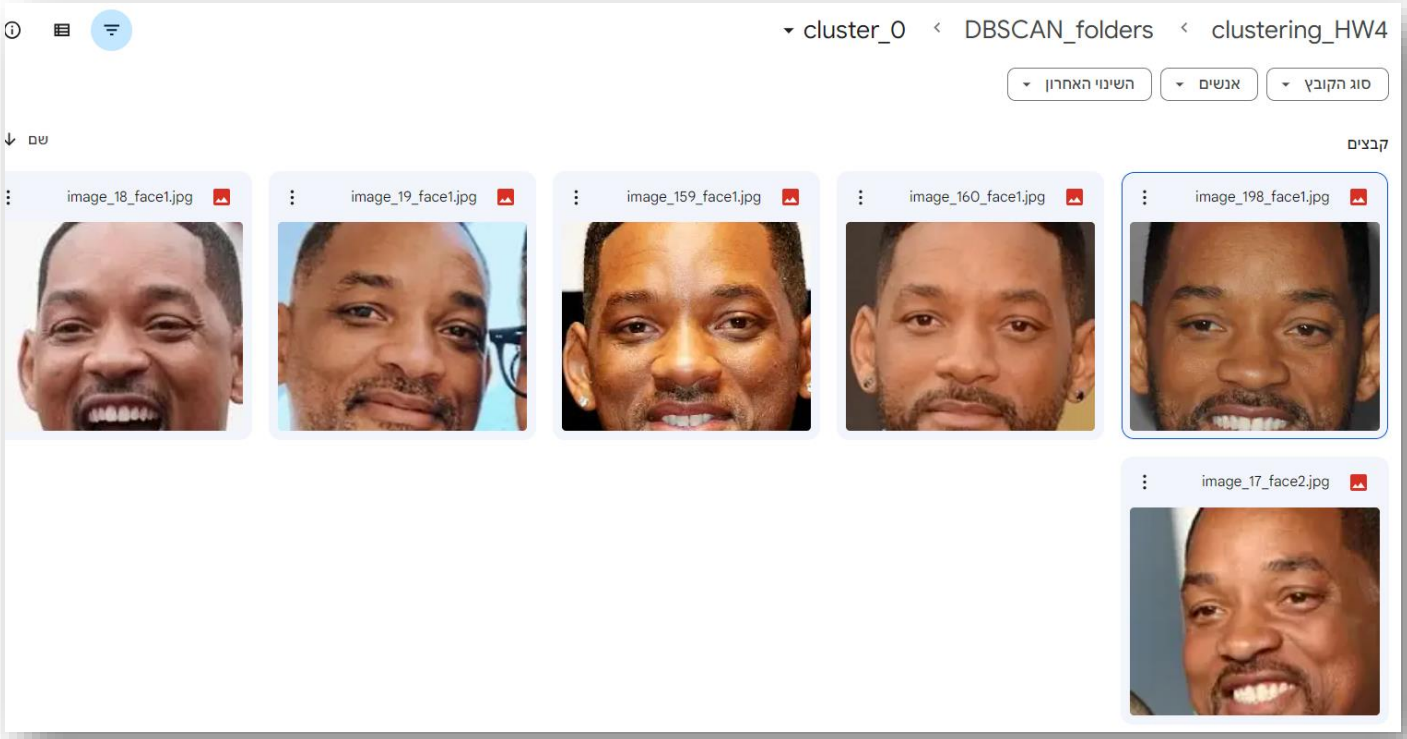
2. חלוקה פחות טובה:



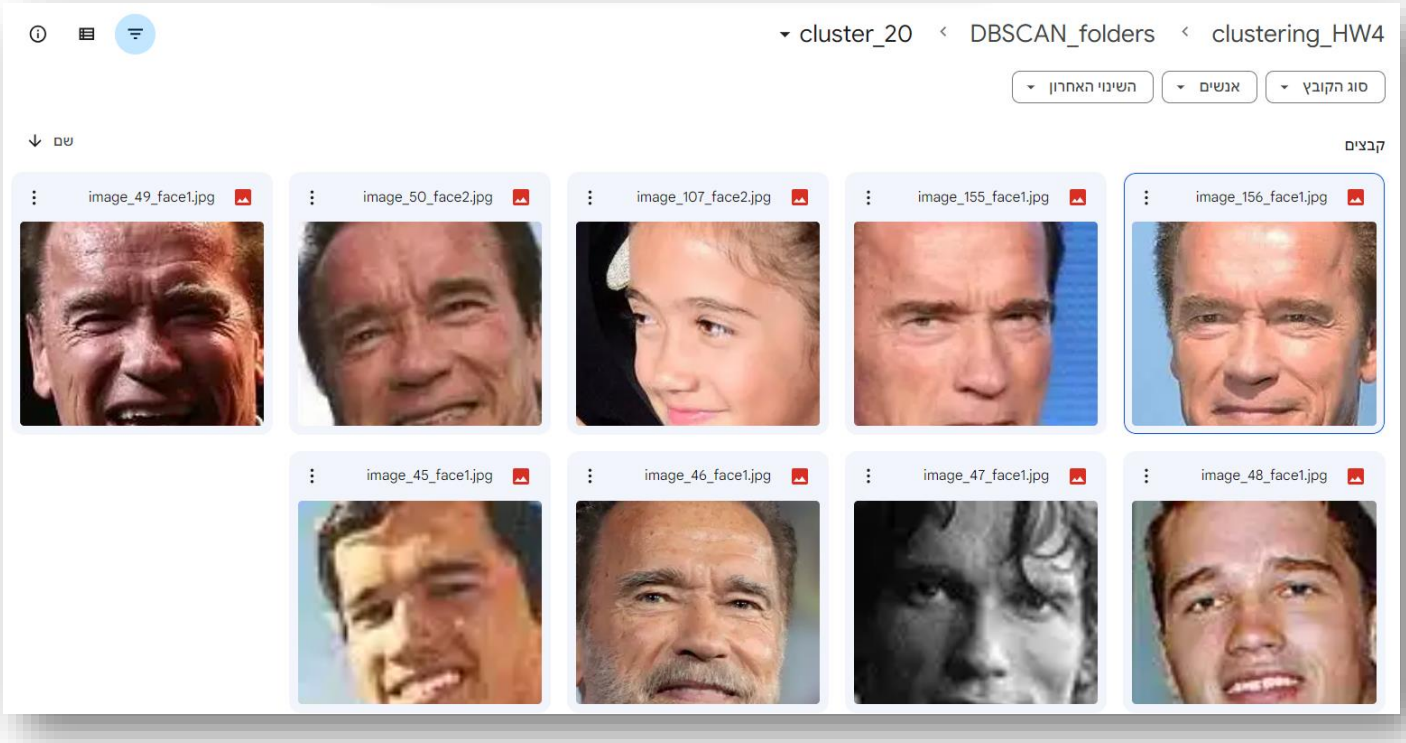
;208886333
;209299478

עבור DBSCAN על הנתונים לאחר הורדת הממדים:

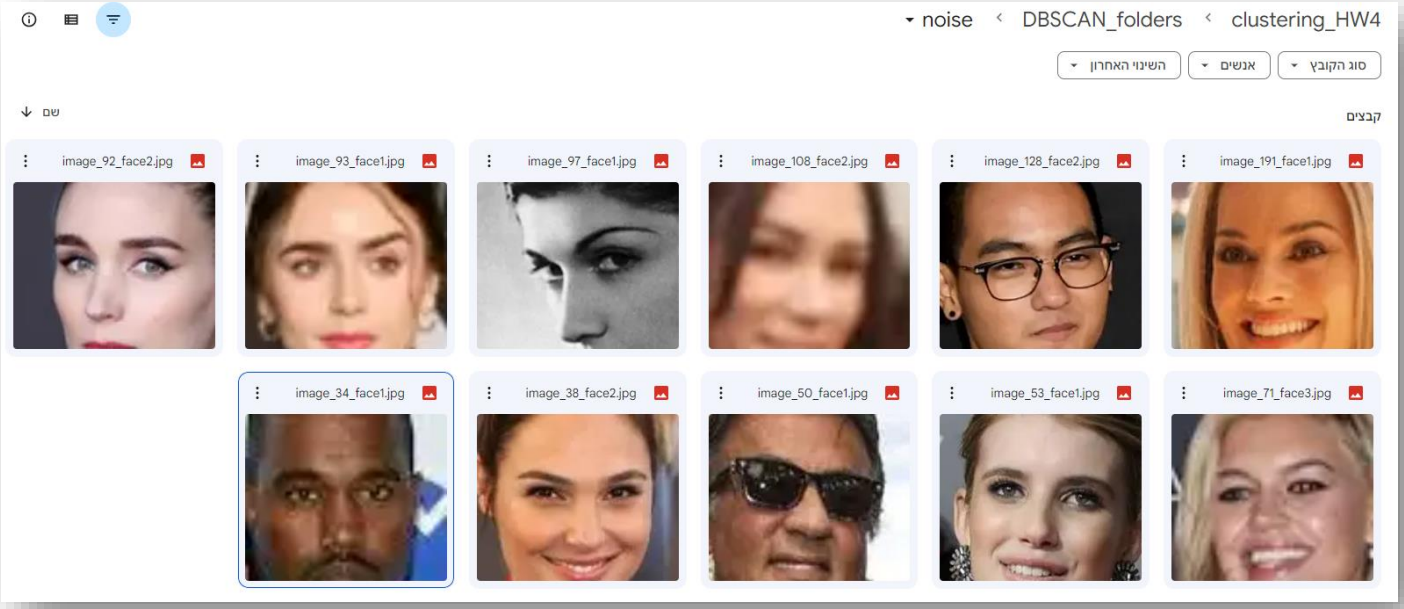
1. חלוקה טובה:



2. חלוקה פחות טובה:



;208886333
;209299478
להלן תיקיית noise:



להערכתנו, מודל זיהוי הפרצופים הצליחו לזהות טוב פרצופים אמיתיים כאשר התמונה הייתה באיכות טובה. כמו כן, ייתכן והגודל המלבן שהגדרנו בחיפוש הפרצופים הקשה בחלק מהתמונות וגרם לפספוס של חלקים בפנים שהיו יכולים לעזור לאלגוריתם בזיהוי.
ניתן לראות בדוגמה של חלוקה לא טובה שהפרצופים שם מטושטשים – דבר המקשה על האלגוריתם להחליט את הפיצ'רים הנכונים.
מכאן, שאם הגדרת הפי'צרים איננה נכונה, המערך של הנתונים שעובר לאלגוריתמים DBSCAN וכן Kmeans לא מדויק לחלוטין. האלגוריתמים מחפשים קרבה בין הנתונים הללו ולכן כאשר ישנם תמונות מטושטשות למשל, ייתכן ובמהלך זיהוי הפרצופים ניתנו להם פיצ'רים דומים ולכן במעמד חלוקת האשכולות הם זוהו כקרובים.

בנוגע לnoise – אנחנו מניחות שהאלגוריתם לא הצליח למצוא התאמה מול שאר התמונות. ייתכן שחלק מהמידע אבד במהלך הורדת הממדים ובכך איבדו מאפיינים שהיוו את הקשר בין התמונות ולכן הוא לא זיהה אותם כחלק מקבוצה אחרת.
בנוסף, מאחר והגדרנו min samples = 2 ייתכן שדמות שהופיעה רק בתמונה אחת לא שויכה לאף קבוצה.