מיני פרויקט
נושאים בלמידה לא מונחית

<u>מבוא</u>

הפרויקט שבחרנו הוא מימוש 3 אלגוריתמי $clustering$ ידועים, השוואה של התוצאות שלנו עם תוצאות המימוש של ספריית $SKLearn$, וניתוח התוצאות.

האלגוריתמים שבחרנו הם:
1. $Gaussian\ Mixture\ Models\ EM$
2. $Mean\ Shift$
3. $Affinity\ Propagation$

את המימוש נכתוב בשפת $Python$.
הקוד יצורף למסמך זה.

אנו נבחן את האלגוריתמים הנ״ל על 5 סוגים של מדגמים שאנו מייצרים, ונציג את התוצאות אשר מצאנו בהם משמעות ולמדנו מהם על אופן פעולת האלגוריתם.
המדגמים אותם נבדוק:
- שני מעגלים אחד בתוך השני.
- שלושה גאוסיאנים בעלי שונות בסדרי גודל שונים.
- שלושה גאוסיאנים בעלי שונות בסדר גודל דומה.
- שלושה גאוסיאנים אשר עברו טרנספורמציה לינארית.
- שני ״ירחים״ משולבים אחד בשני.

לבסוף, נשווה את התוצאות שאנו מקבלים מהאלגוריתמים השונים וננסה לבחור את האלגוריתם הטוב ביותר עבור כל מדגם.

GMM EM – Gaussian Mixture Models Expectation Maximization

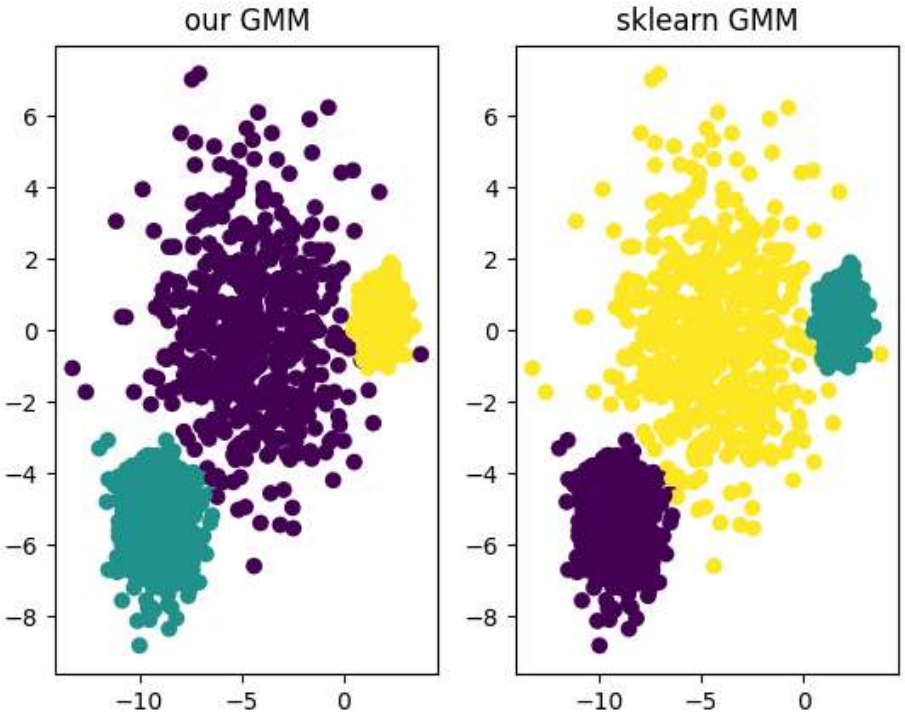
$GMM\ EM$ הוא אלגוריתם אשר לומד $Gaussian\ Mixture\ Model$ לייצוג המדגם.
האלגוריתם מקבל פרמטר K - ולומד GMM המורכב מ-K קומפוננטות גאוסיאניות.

האלגוריתם הוא אלגוריתם איטרטיבי כאשר בכל איטרציה אנו מחשבים את הסתברות השייכות של כל נקודה במדגם לכל אחד מ K הגאוסיאנים בהינתן הפרמטרים של הגאוסיאנים ($E\ step$), ולאחר מכן מחשבים מחדש את הפרמטרים של הגאוסיאנים בהינתן השייכות של הנקודות במדגם ($M\ step$).

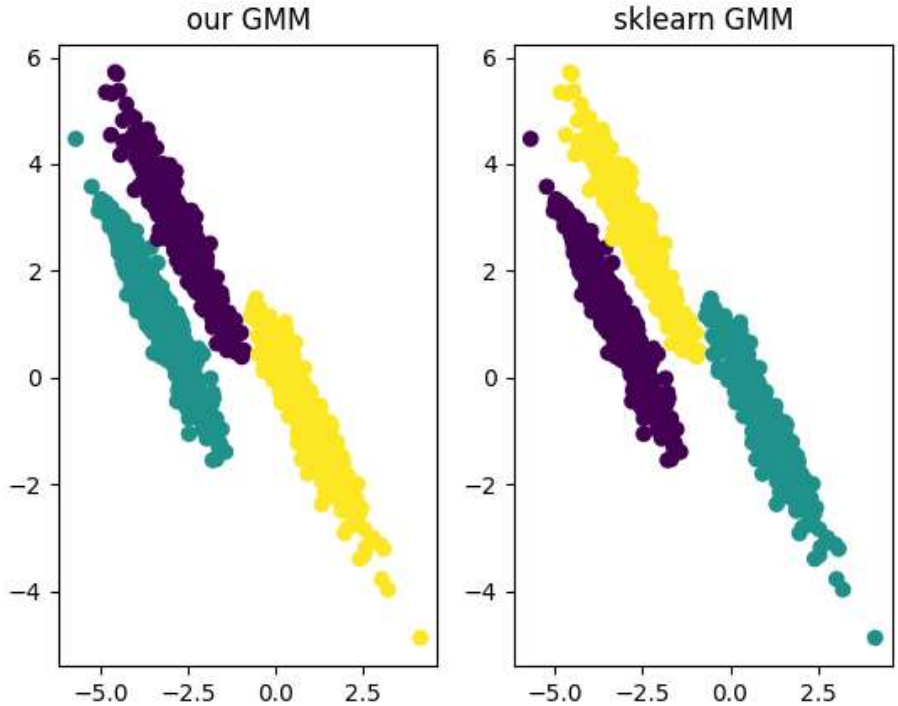
אנו בחרנו לממש את האלגוריתם בצורה כזו שב- $E\ step$ אנו מבצעים $hard\ assignment$. כלומר, אנו קובעים את השייכות של נקודה במדגם לגאוסיאן אליו ההסתברות לשייכות היא הגבוהה ביותר.

נתבונן בתוצאות ההרצה של המימוש שלנו לעומת מימוש הספרייה:

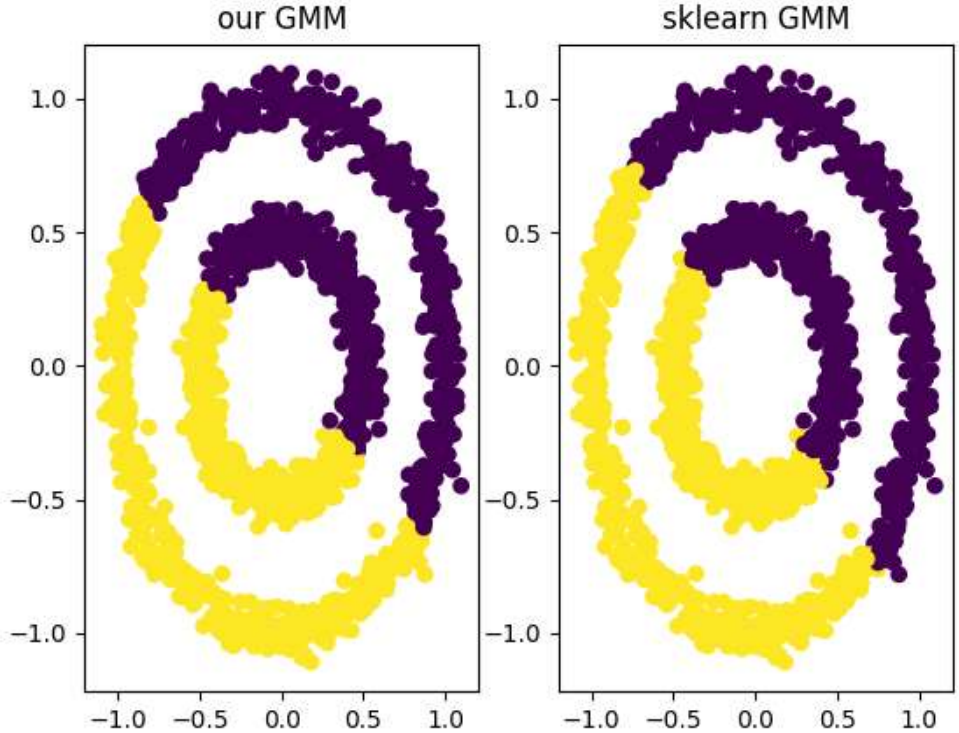
1.1



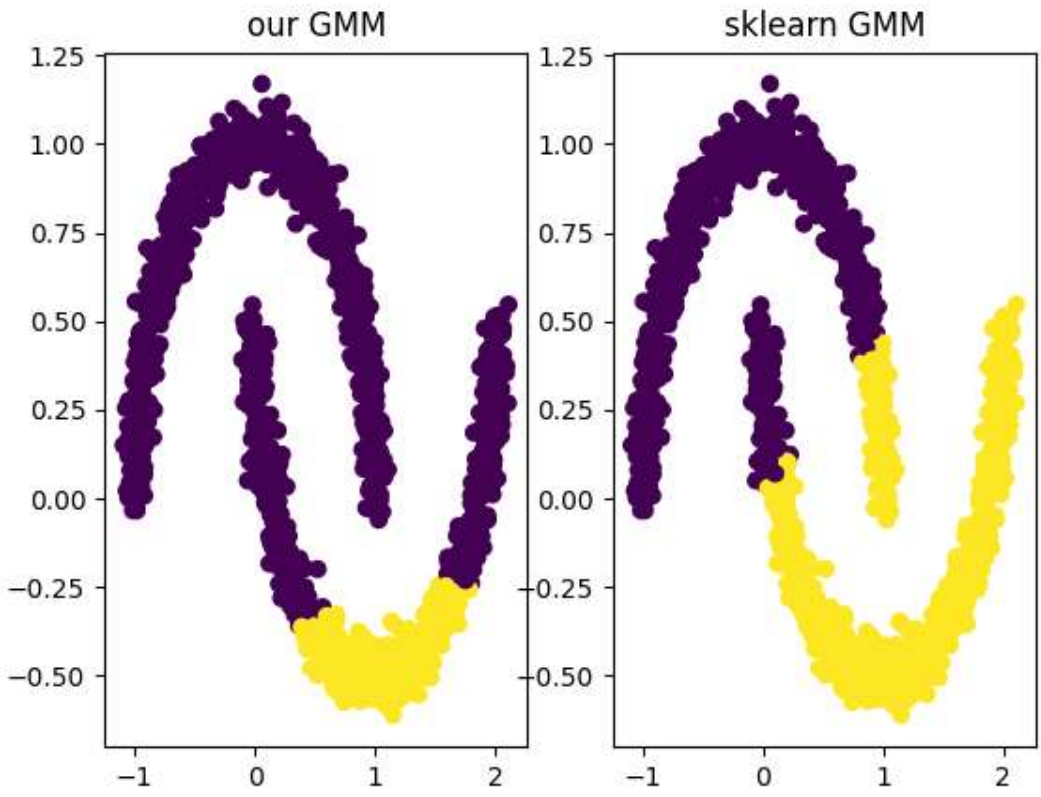
1.2


our GMM
sklearn GMM
6
4
2
0
−2
−4
−5.0
−2.5
0.0
2.5

1.3


our GMM
sklearn GMM
1.0
0.5
0.0
−0.5
−1.0

.1.4



<u>ניתוח התוצאות:</u>
ניתן להבחין כי עבור המדגמים 1.1, 1.2, שני המימושים הצליחו ממש לזהות את הגאוסיאנים שמהם דגמנו את הנקודות. לעומת זאת, עבור המדגמים של הירחים והעיגולים, האלגוריתם התקשה להבדיל בין העיגול החיצוני לפנימי או בין הירח העליון לתחתון.
מכיוון שהאלגוריתם לומד מודל שמורכב מקומפוננטות גאוסיאניות, אנו יכולים לצפות שעבור מידע שאינו מגיע מהתפלגות אשר מבוססת גאוסיאנים – המודל יהיה מוגבל ביכולת שלו לבצע *clustering* בצורה טובה.

Mean Shift

$Mean\ Shift$ הוא תהליך שמטרתו היא למצוא את ה "שכיח" של פונקציית צפיפות בהינתן מדגם מתוכה. ניתן להשתמש בתהליך זה על מנת לבצע $clustering$.
האלגוריתם מקבל פרמטר $bandwidth$ אשר בא לידי ביטוי בחישוב פונקציית קרנל ובחישוב סביבות המרכזים.
התהליך הוא איטרטיבי ואנו מתייחסים לכל מרכז כאל תוחלת נקודות המדגם אשר בסביבתו. בכל איטרציה אנו מחשבים מחדש את המרכזים באמצעות ביצוע פונקציית קרנל על המרחקים מהמרכז אל הנקודות מהמדגם שבסביבתו.

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

ניתן להסתכל על האלגוריתם כ "מטפס על הר" – בכל איטרציה, אנו מבצעים "הזזה" של המרכזים לעבר אזורים בהם ערך פונקציית הצפיפות הוא גבוה יותר.

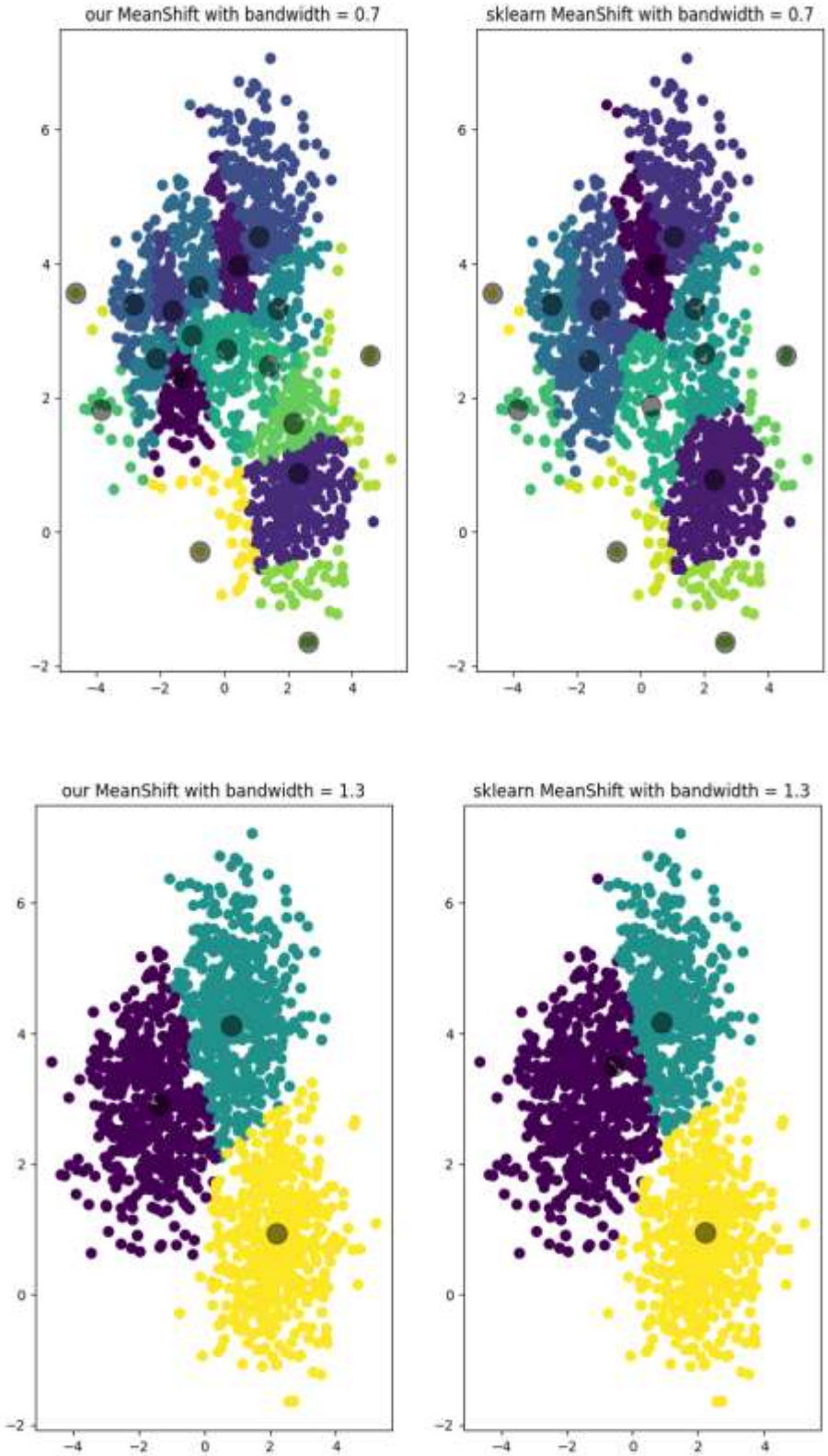
האלגוריתם מתחיל עם כל נקודות המדגם כמרכזים, ולאחר התכנסות משמיט את המרכזים שקרובים אחד לשני (ניתן לקבוע ע"י סף מוגדר מראש). מכיוון שחישובי המרכזים הם בלתי תלויים אחד בשני, ניתן לחשב אותם במקביל.

באופן זה האלגוריתם "מסיק בעצמו" את מספר ה-$clusters$, ולא צריך לקבל אותו כפרמטר כמו $Kmeans$ או GMM.

במימוש שלנו אנו משתמשים בקרנל גאוסיאני, וניסינו כמה ערכי $bandwidth$ שונים.
על מנת להאיץ את מהירות החישוב, אנו משתמשים ב$process\ pool$ עם מספר קבוע של $worker\ processes$ כאשר כל $process$ מקבל מרכז, ומבצע את כל האיטרציות עד שהמרכז מתכנס או שהגיע למקסימום איטרציות. האלגוריתם הוא בעל זמן ריצה של $O(n^2)$, ולכן הרצה מקבילית תורמת משמעותית לזמן הריצה. ההבדל בין זמן הריצה של המימוש מהספרייה לבין המימוש שלנו עומד על בין 1 – 40 שניות, כתלות ב-$bandwidth$ ובמדגם.

נתבונן בתוצאות ההרצה של המימוש שלנו לעומת מימוש הספרייה:

2.1

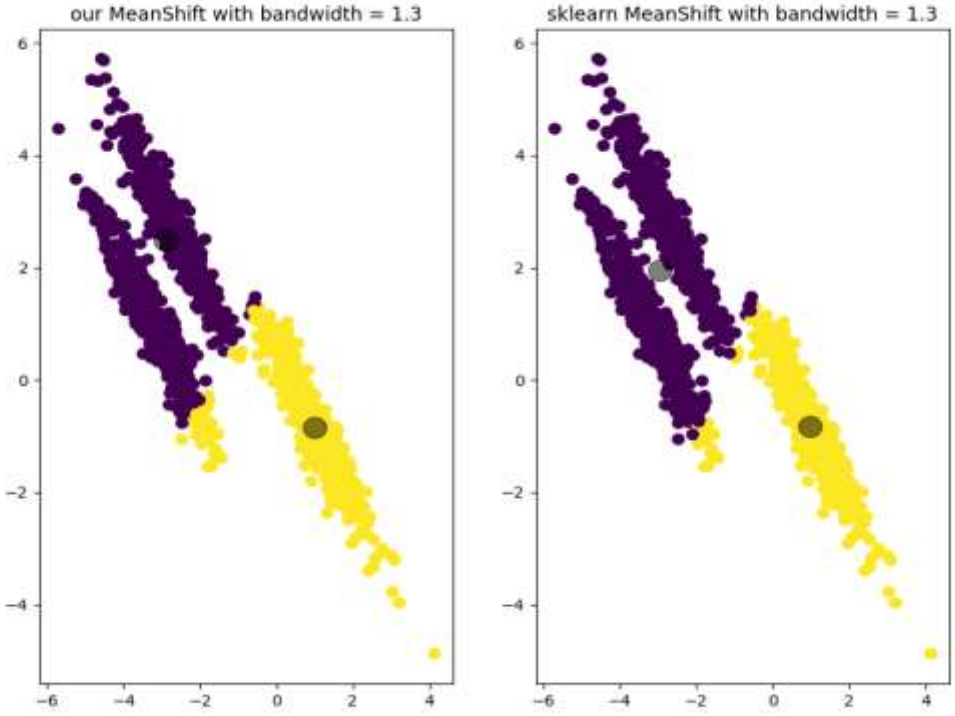

our MeanShift with bandwidth = 0.7
sklearn MeanShift with bandwidth = 0.7
our MeanShift with bandwidth = 1.3
sklearn MeanShift with bandwidth = 1.3

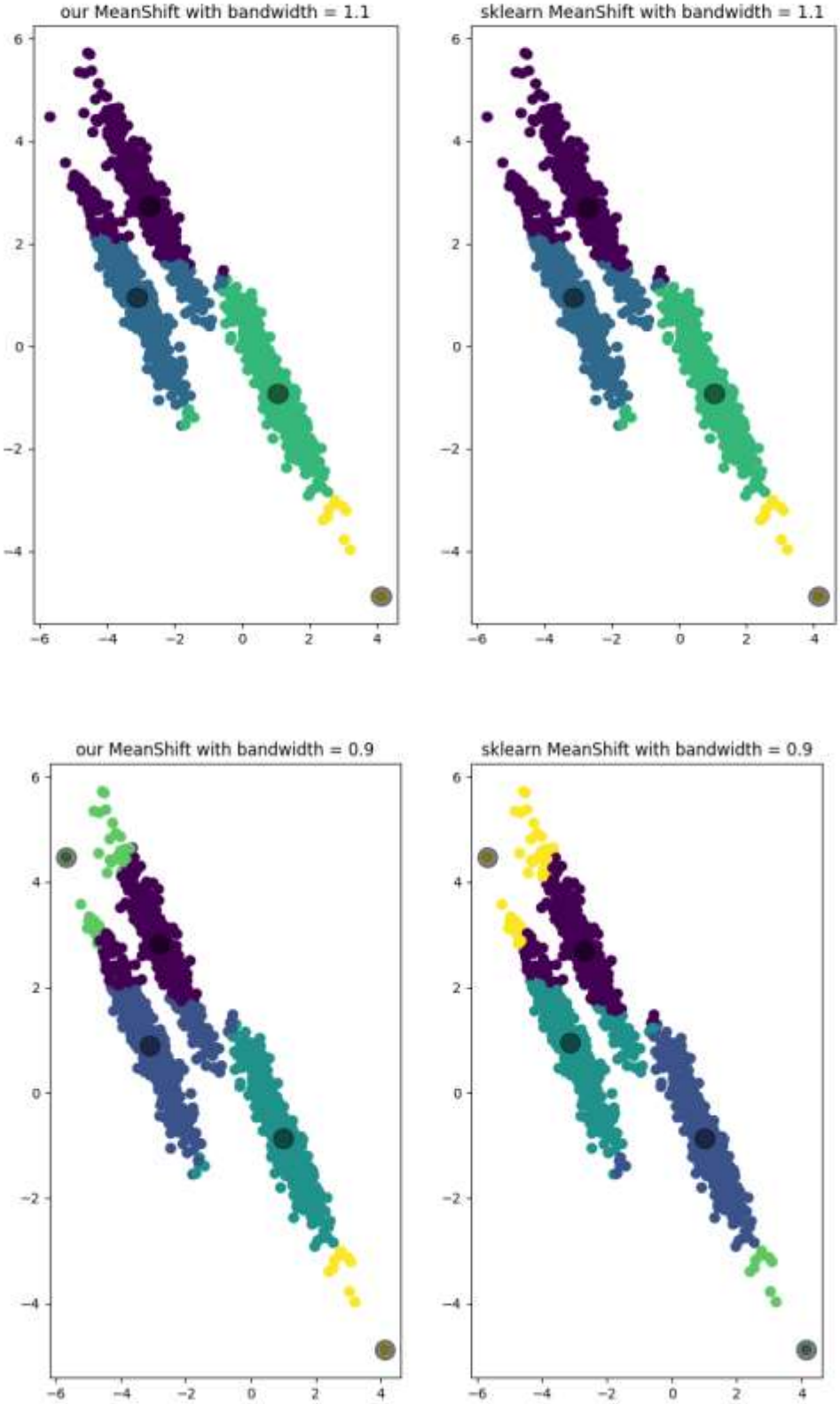
ניתן להבחין במדגם הנ״ל (2.1) כי עבור ערך $bandwidth$ נמוך, שני המימושים מתכנסים למספר רב של מרכזים. לעומת זאת, כאשר הגדלנו את ערך ה$bandwidth$, האלגוריתם התכנס בדיוק ל-3 מרכזים בשני המימושים בצורה אופטימלית.

נתבונן כעת בדגם בו דווקא ערך קטן יותר של $bandwidth$ עזר לאלגוריתם למדל את המדגם בצורה טובה יותר.

ניתן לראות שעבור התוצאות למטה (2.2), ערך $bandwidth$ גבוה מדי אפשר לאלגוריתם ללמוד $clusters$ גדולים מדי. לעומת זאת כאשר מקטינים קצת את הערך, האלגוריתם הצליח למדל את ה-$clusters$ עם ירידה טובה יותר לפרטים, אך עדיין לא בצורה מושלמת. כאשר הקטנו אפילו עוד, קיבלנו שוב את התוצאה של יותר מדי מרכזים.

2.2




our MeanShift with bandwidth = 1.1
sklearn MeanShift with bandwidth = 1.1
our MeanShift with bandwidth = 0.9
sklearn MeanShift with bandwidth = 0.9

Affinity Propagation

Affinity Propagation הינו אלגוריתם *clustering* המבוסס על "העברת הודעות" בין נקודות המדגם. ההודעות שנשלחות בין הזוגות של הנקודות מייצגות את מידת ההתאמה של נקודה אחת לייצג נקודה אחרת. בשונה מאלגוריתמי *clustering* כמו *K-Means* ואחרים, אלגוריתם זה אינו דורש לדעת את מספר ה-*clusters* מראש. האלגוריתם מוצא נציגים (*exemplars*) מתוך המדגם הנתון, אשר מייצגים את ה-*clusters* השונים.

האלגוריתם משתמש במטריצת *similarity* שבה נתון עבור כל זוג נקודות מהמדגם מהי מידת הדומות שלהן (נסמן $s(i,j)$), את ה-*similarity* ניתן לחשב עם פונקציה שמקיימת: $s(i,j) > s(i,k) \Leftrightarrow x_i \text{ is more similar to } x_j \text{ than to } x_k$. במימוש שלנו נשתמש בפונקציה $s(i,j) = -\left\|x_i - x_j\right\|^2$.

האלגוריתם הינו איטרטיבי, כך שבכל איטרציה נשלחות ההודעות בין כל הזוגות האפשריים של הנקודות, ומתבצע עדכון של ההתאמה של נקודה לייצג נקודה אחרת.
ישנן שתי קטגוריות של הודעות אותן מעבירות הנקודות אחת לשנייה (לפי סדר זה):
1) ***responsibility***: משמעותו כמה מתאימה נקודה אחת לייצג נקודה אחרת.
עבור נקודות i ו-k מחושב:

$$r(i,k) = s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\}$$

2) ***availability***: משמעותו כמה כדאי לנקודה אחת לבחור בנקודה אחרת לייצג אותה.
עבור נקודות i ו-k מחושב:

$$a(i,k) = min\left\{0, r(k,k) + \sum_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\right\} \; for \; i \neq k$$

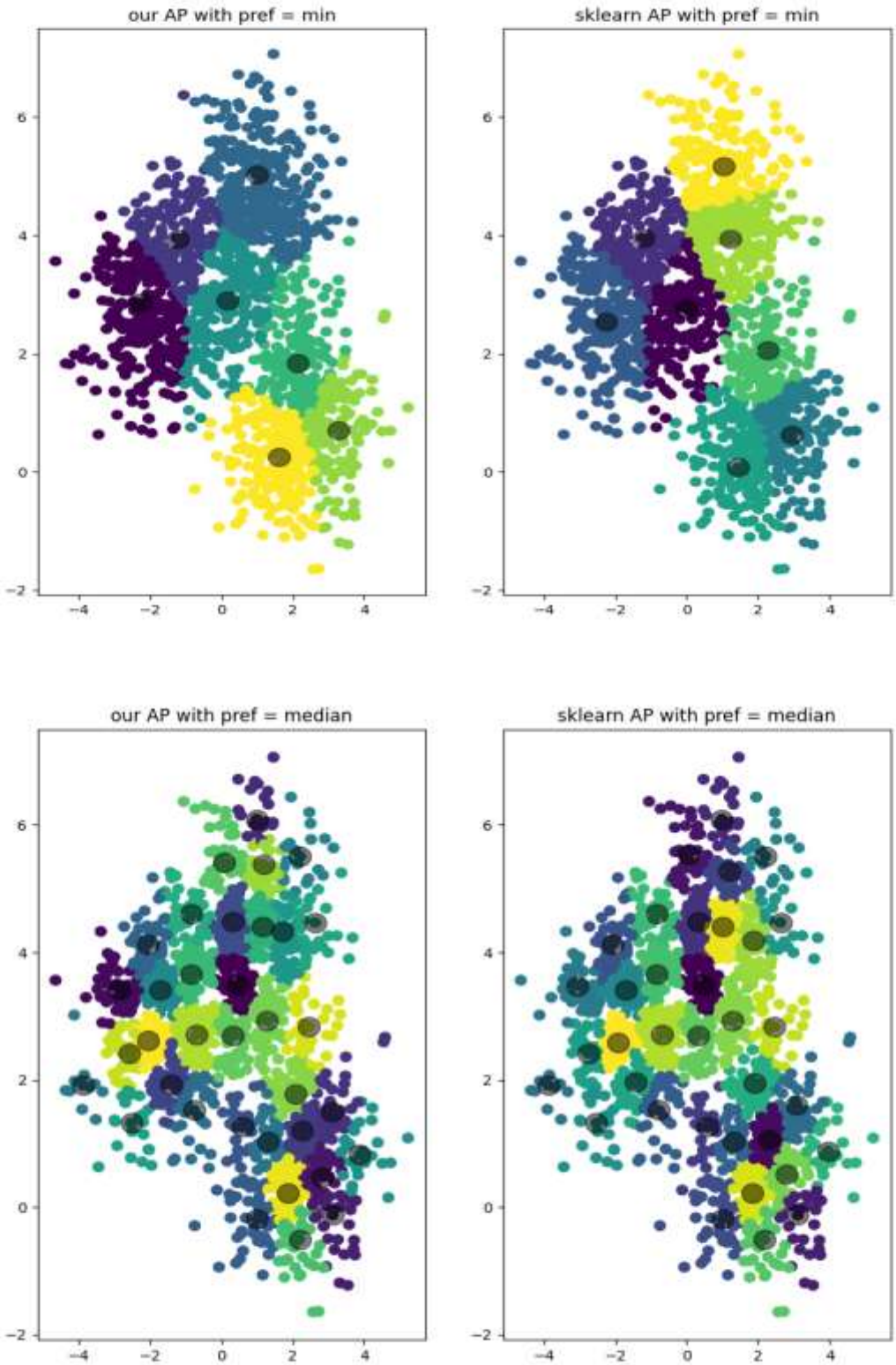
$$a(k,k) = \sum_{i' \neq k} \max\{0, r(i',k)\}$$

איטרציות אלה ממשיכות עד להתכנסות, שבה נקבעים באופן סופי ה-*exemplars*.
לחלופין, אם הריצה לא מתכנסת לפתרון, האלגוריתם יעצור לאחר מספר מקסימלי של איטרציות (מספר זה מתקבל כפרמטר).

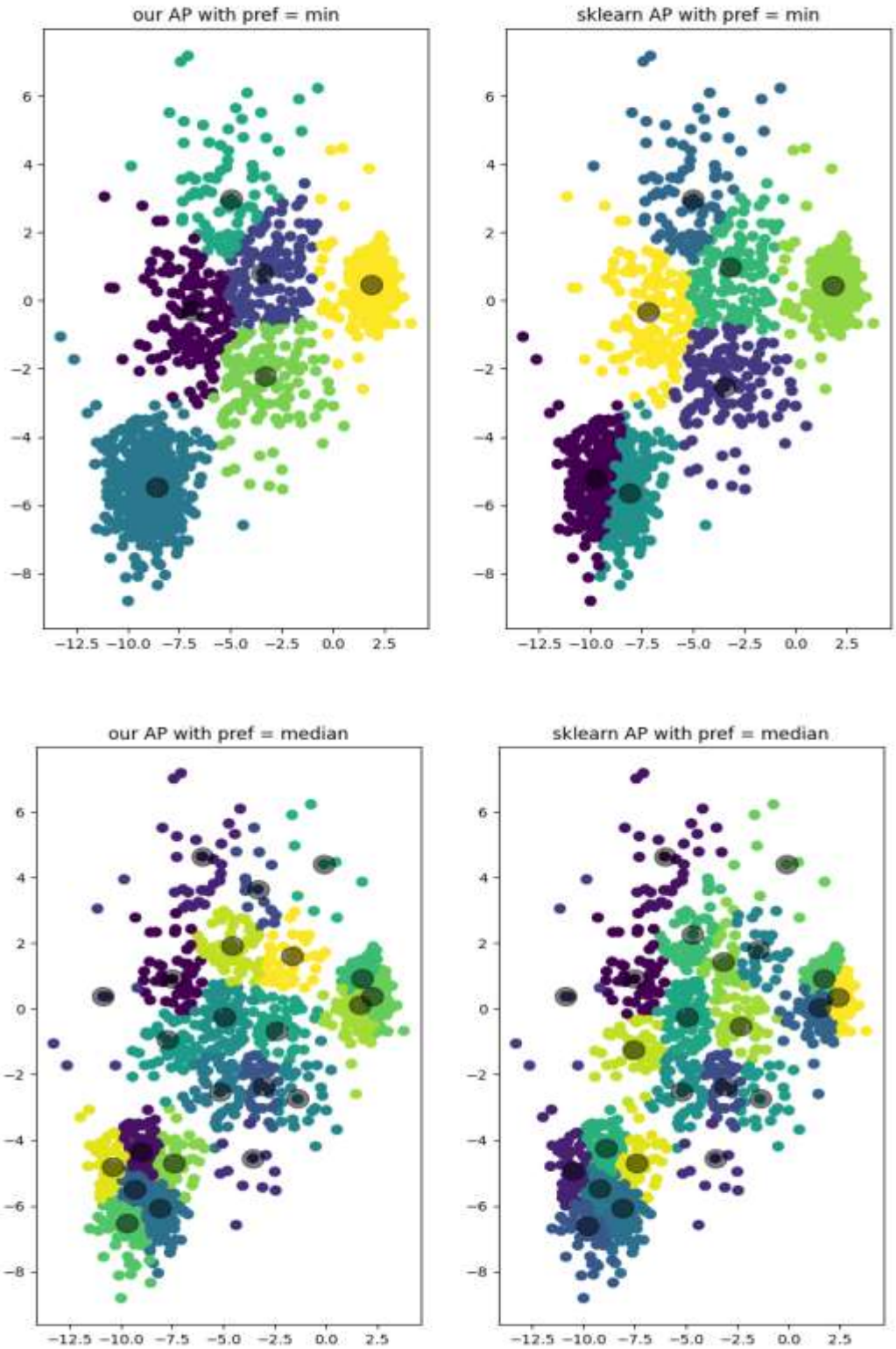
האלגוריתם מקבל פרמטר אחד חשוב – *preference*.
פרמטר זה משפיע על מספר ה-*exemplars* שייבחרו (כלומר על מספר ה-*clusters*):
preference נמוך (קרוב לערך המינימלי של ה-*similarity* של המדגם) יניב מספר נמוך של *exemplars*, ואילו *preference* גבוה יותר יניב מספר גדול יותר של *exemplars*.
כברירת מחדל, ה-*preference* שבו משתמש האלגוריתם הוא ה-*similarity* החציוני (*median*) של המדגם.

נתבונן בתוצאות ההרצה של המימוש שלנו לעומת מימוש הספרייה:

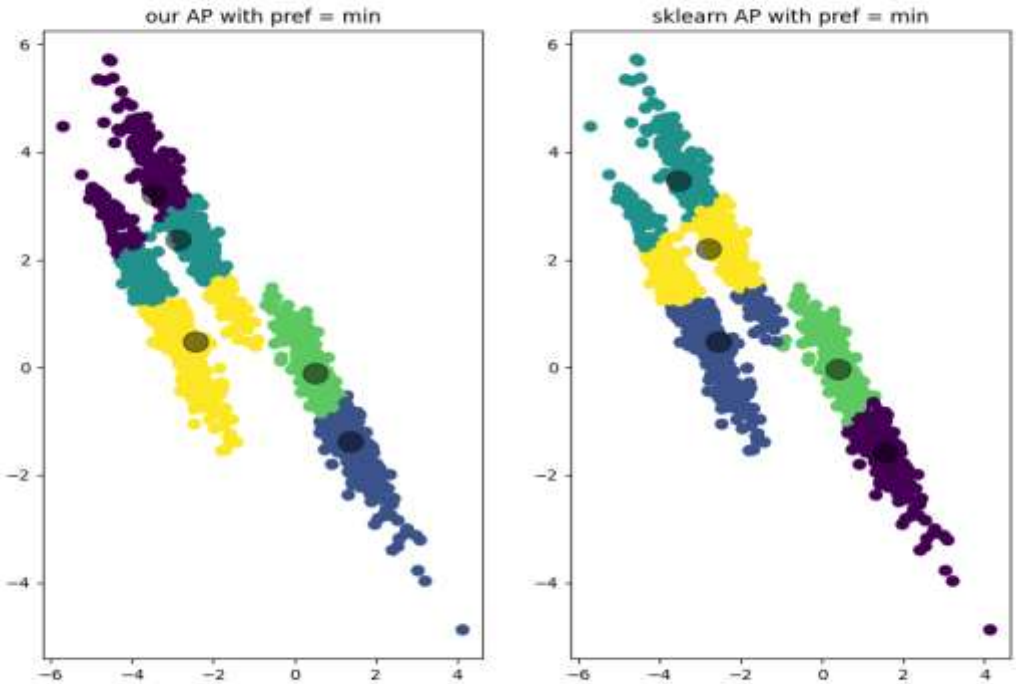
3.1

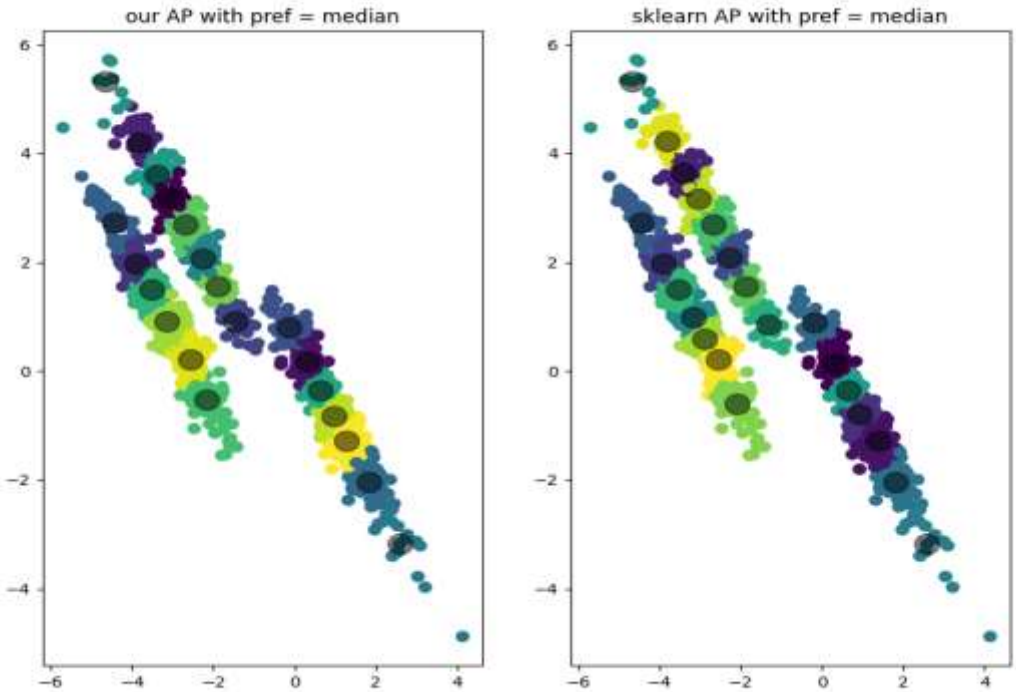

our AP with pref = min
sklearn AP with pref = min
our AP with pref = median
sklearn AP with pref = median

3.2

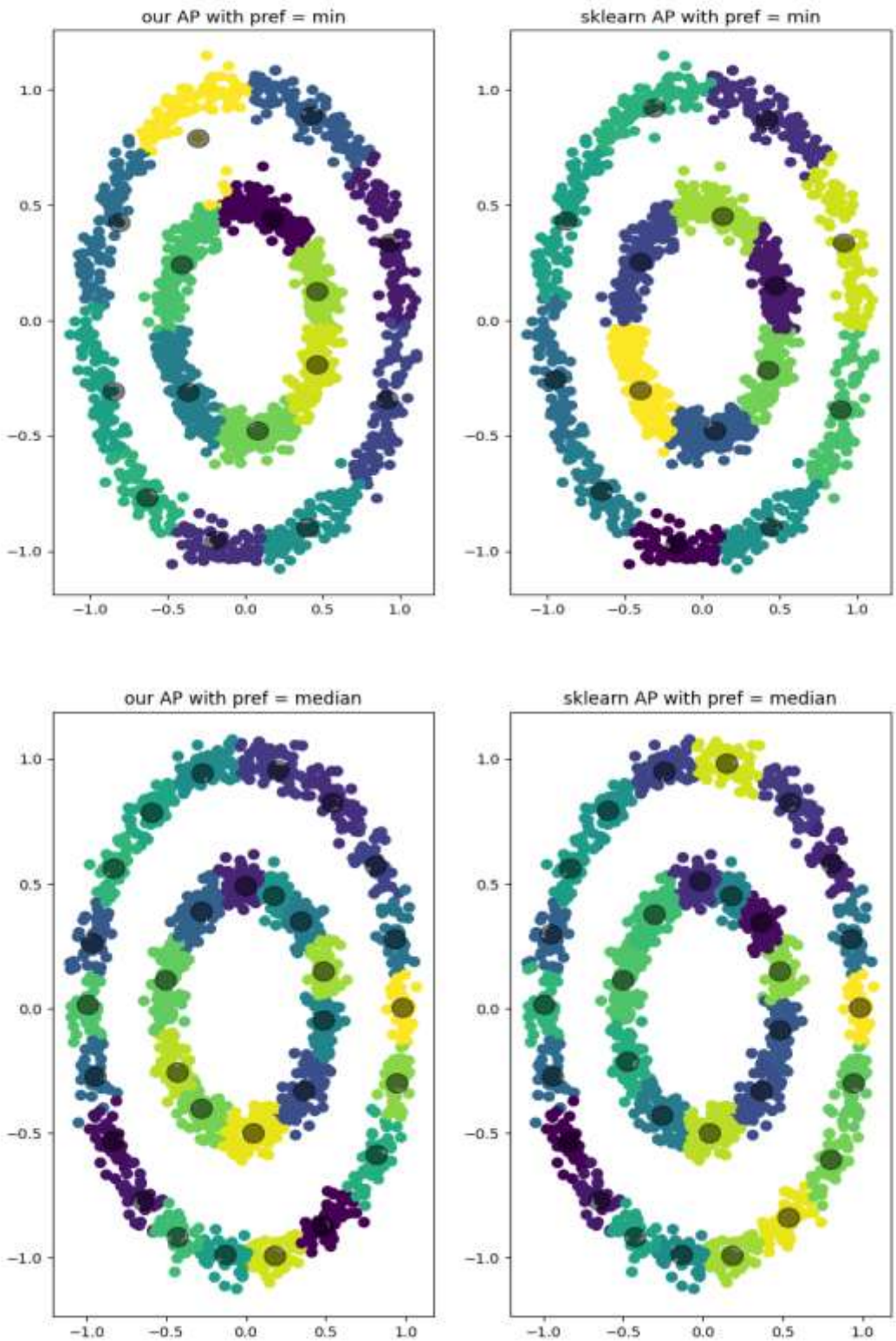

our AP with pref = min
sklearn AP with pref = min
our AP with pref = median
sklearn AP with pref = median
6
4
2
0
−2
−4
−6
−8
−12.5 −10.0 −7.5 −5.0 −2.5 0.0 2.5

3.3

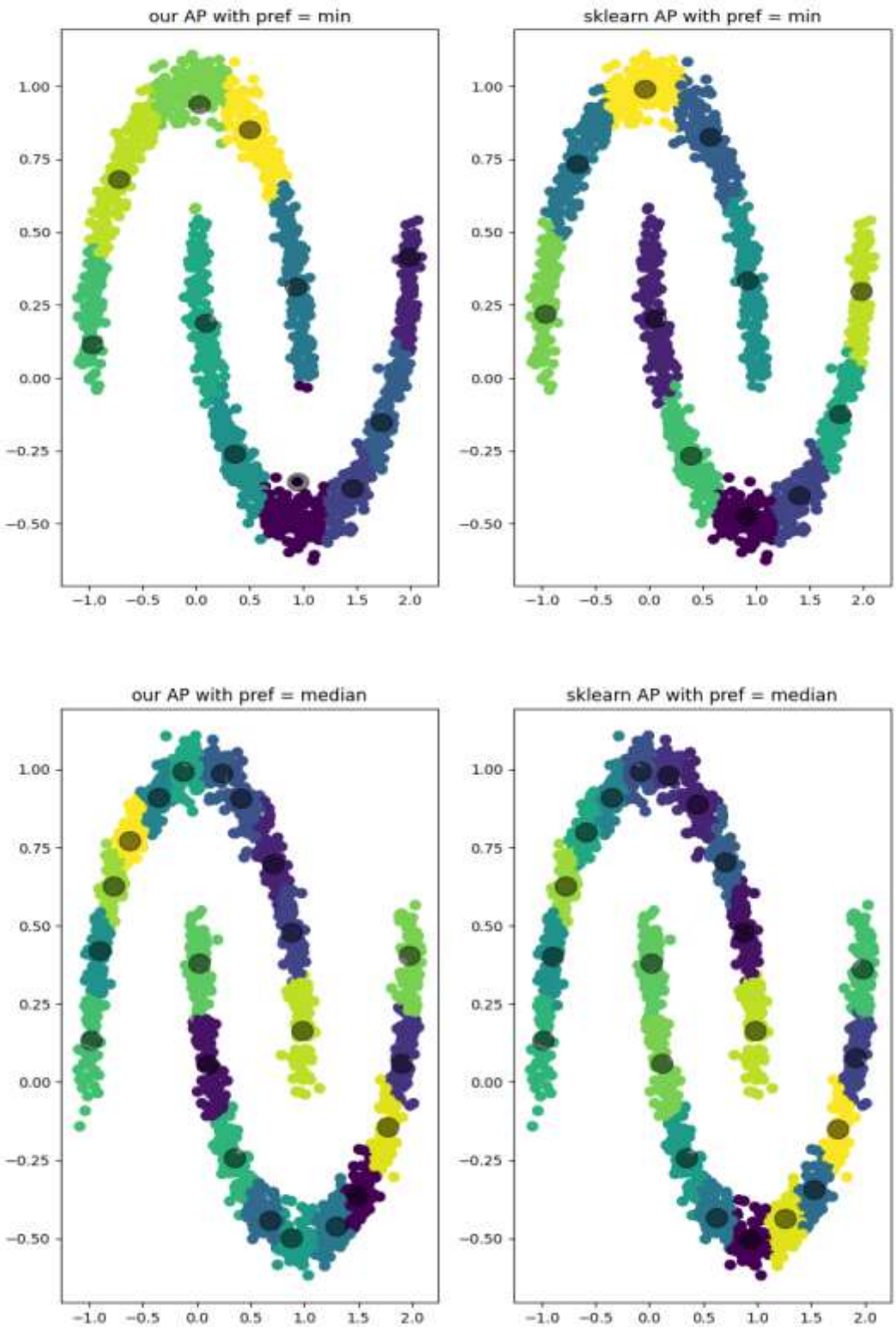

our AP with pref = min
sklearn AP with pref = min


our AP with pref = median
sklearn AP with pref = median

3.4


our AP with pref = min
sklearn AP with pref = min
our AP with pref = median
sklearn AP with pref = median
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0

3.5

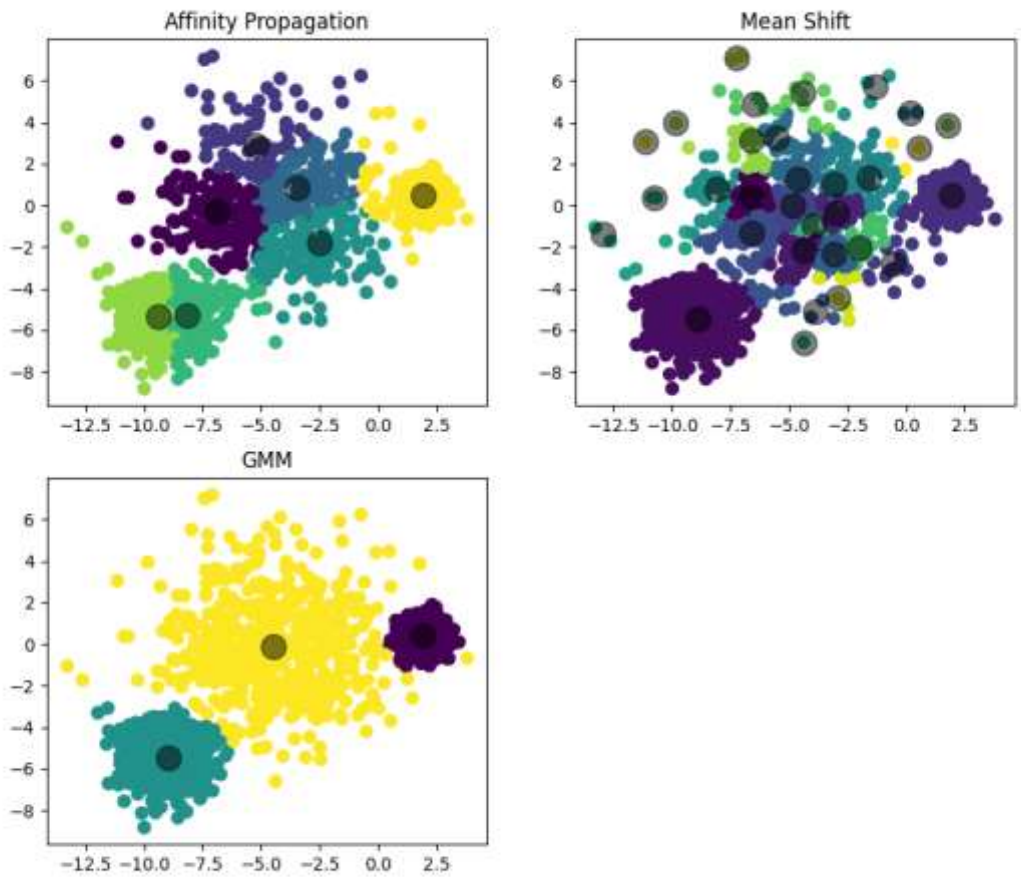

our AP with pref = min
sklearn AP with pref = min
our AP with pref = median
sklearn AP with pref = median
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

<u>ניתוח התוצאות:</u>
ניתן לראות כי בכל התוצאות הערך של הפרמטר $preference$ השפיע באופן משמעותי על מספר ה-$clusters$. בתוצאות בהן ה-$preference$ היה מינימלי האלגוריתם בחר מספר קטן בהרבה של $clusters$ מאשר בתוצאות בהן ה-$preference$ נקבע להיות ה-$median$. הסיבה לכך היא שה-$preference$ קובע את האלכסון של מטריצת ה-$similarity$, כלומר מתקיים כי באתחול האלגוריתם $s(i,i) = preference$. ככל שה-$preference$ קטן, גם הסיכוי שנקודה תיבחר להיות $exemplar$ קטן. כאשר ה-$preference$ נקבע להיות ה-$median$, לכל הנקודות יש סיכוי שוויני יותר להיבחר כ-$exemplars$.

בנוסף ניתן לראות כי התוצאות שהניב המימוש שלנו כמעט זהות לתוצאות של המימוש של הספרייה גם במספר ה-$clusters$ וגם במיקומם.

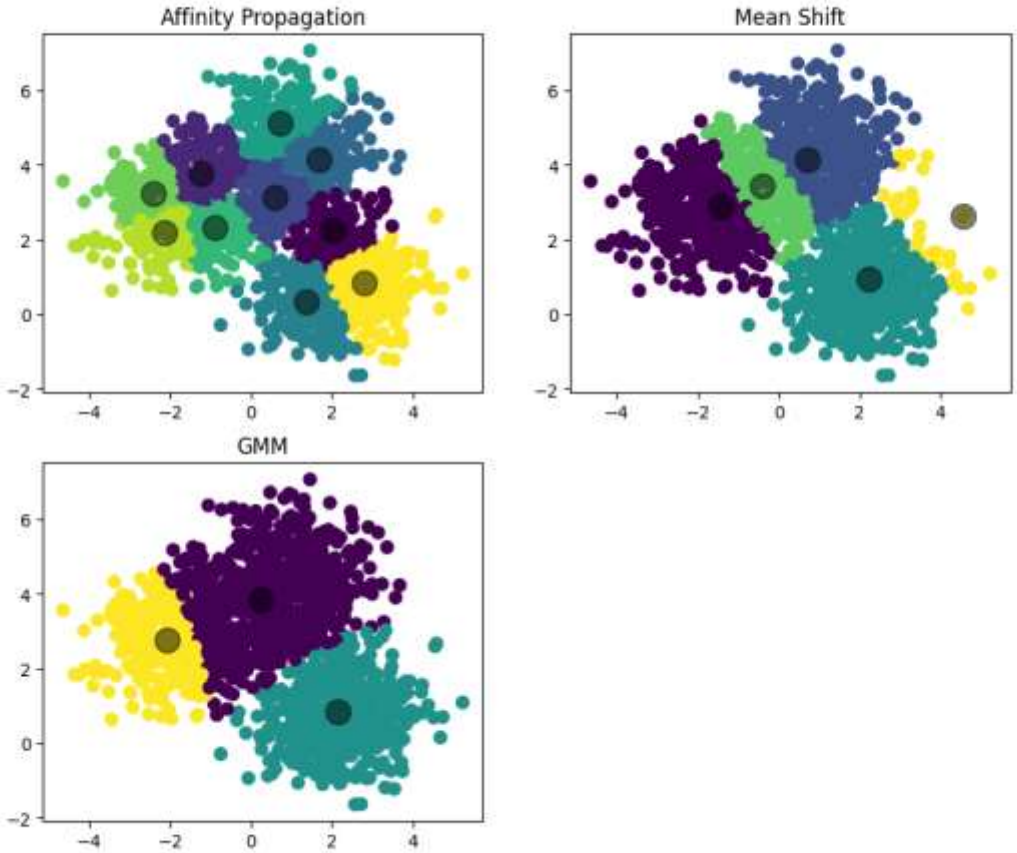
<u>השוואה בין האלגוריתמים השונים</u>

1) שלושה גאוסיאנים בעלי שונות בסדרי גודל שונים



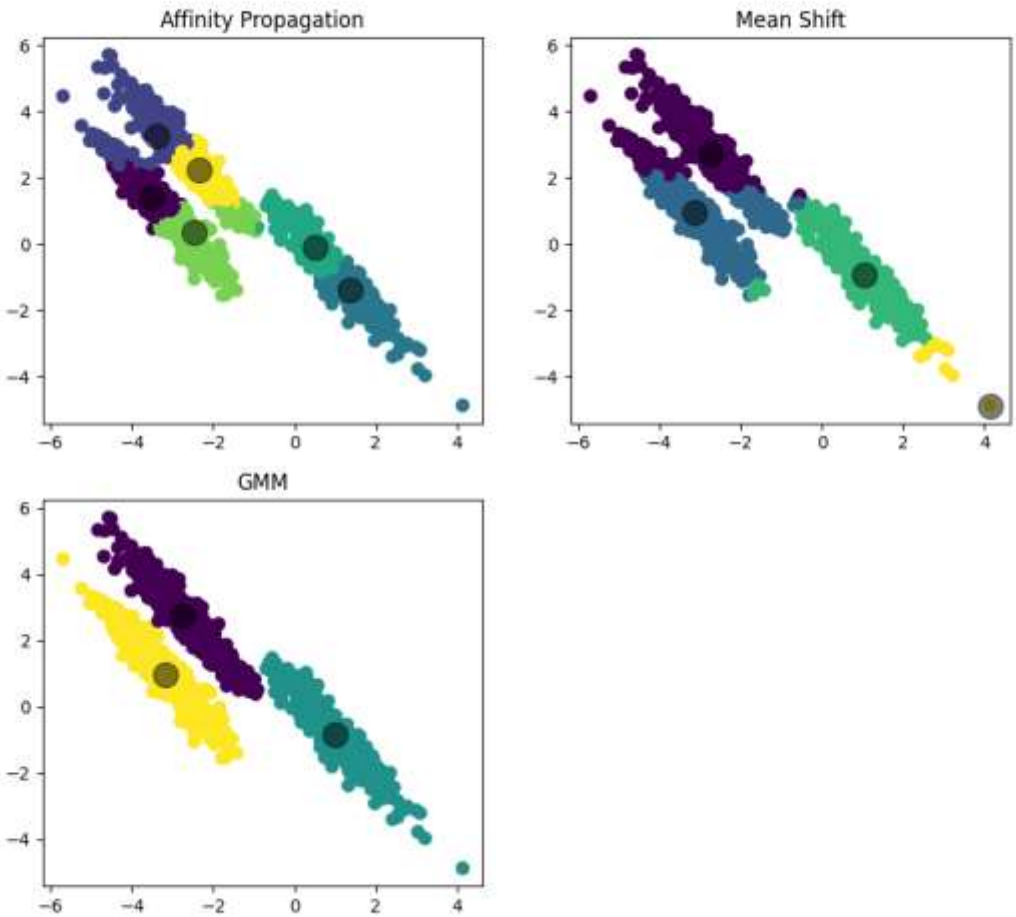
עבור מדגם זה ניתן לראות כי GMM קיבל את התוצאה הטובה ביותר, כאשר הוא קיבל כפרמטר להתאים 3 גאוסיאנים. לעומת זאת, שני האלגוריתמים האחרים נתנו פרשנות אחרת למדגם.
ניתן לראות כי Mean Shift הצליח לזהות שני clusters בצורה יחסית ברורה בצדדים, אך במרכז התוצאה שהתקבלה היא מספר רב של clusters כאשר הרבה מהם בעצם נקודות בודדות במדגם. תוצאה זאת הגיונית מכיוון שהנקודות במרכז קצת יותר רחוקות אחת מהשנייה ולכן יכול להיות שערך ה-bandwidth היה קטן מדי ($bandwidth = 1.1$).
ניתן לראות כי Affinity Propagation כן עשה clustering, אך ההתייחסות שלו למדגם שונה מזאת של Mean Shift ו-GMM. ניתן לראות ה-clusters כולם באותו הגודל, כלומר האלגוריתם מצא חלוקה שווה של המדגם.

2) שלושה גאוסיאנים בעלי שונות בסדר גודל דומה



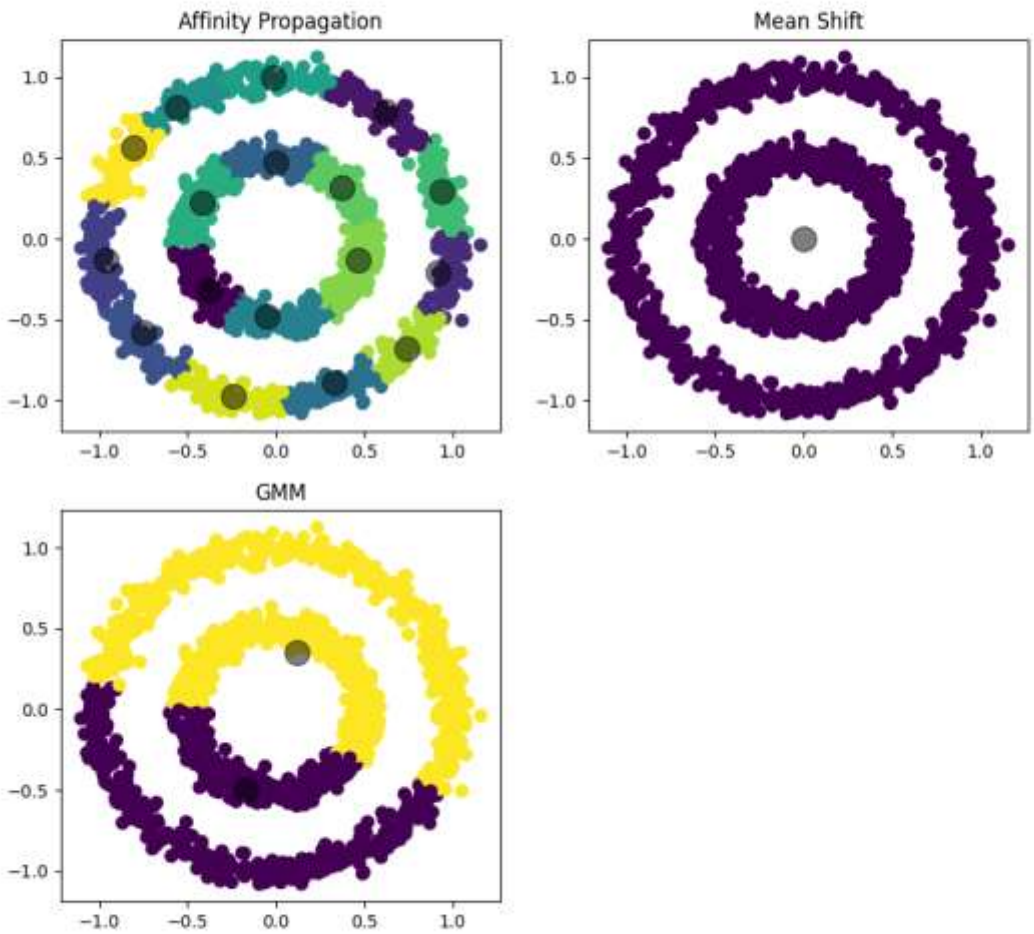
עבור מדגם זה ניתן לראות ממבט ראשון ש-*GMM* נתן חלוקה יחסית טובה אך בכל זאת לא זיהה בצורה מדויקת את שלושת הגאוסיאנים (ה-*cluster* הצהוב נקטע). אפשר לראות ש-*Mean Shift* נתן תוצאה יחסית מדויקת. ניתן לראות שלושה *clusters* שמייצגים בצורה טובה את רוב המדגם, כאשר ה-*cluster* הצהוב ניתן לפירוש כשגיאה של האלגוריתם בעקבות נקודה מבודדת שסווגה כמרכז. פעם נוספת ניתן לראות כי *Affinity Propagation* נתן מספר רב יחסית של *clusters* בגודל שווה, אך ה-*clustering* שהוא ביצע לא זיהה את הגאוסיאנים.

3) שלושה גאוסיאנים אשר עברו טרנספורמציה לינארית



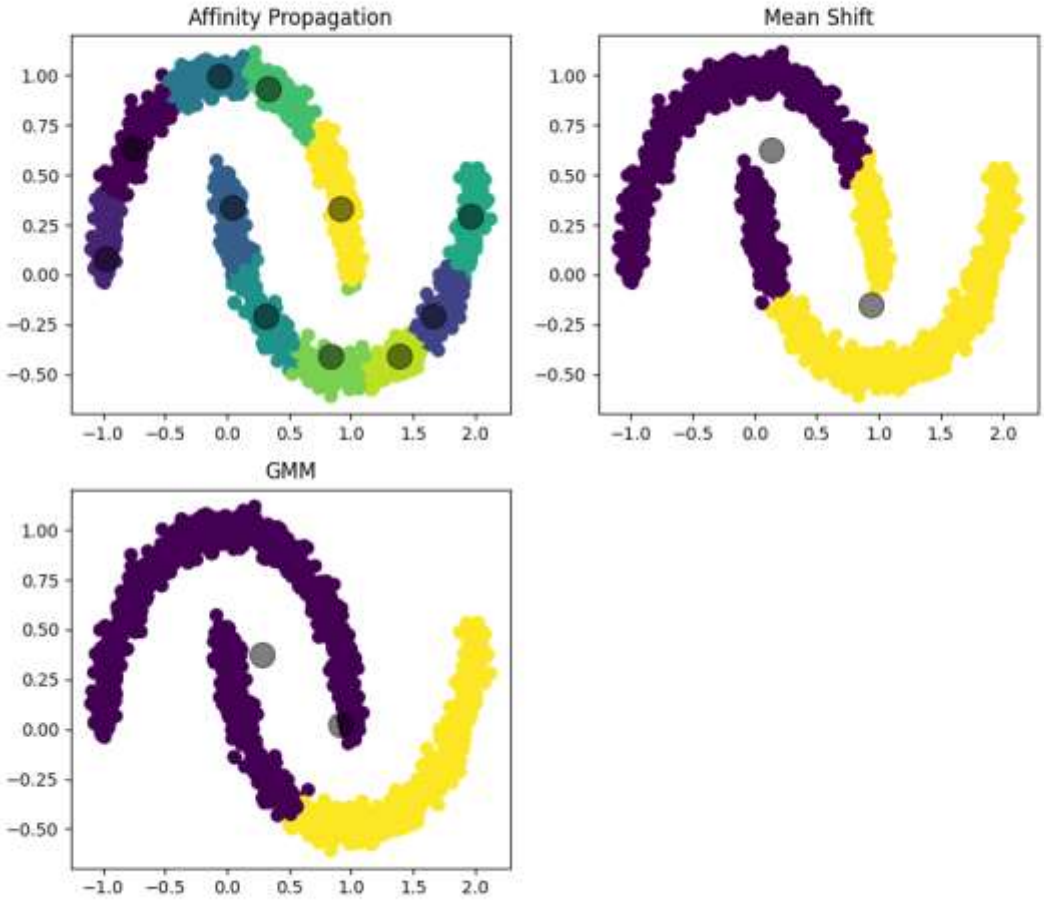
ניתן לראות כי *GMM* זיהה בהצלחה את שלושת הגאוסיאנים שעברו טרנספורמציה.
פעם נוספת *Mean Shift* נתן *cluster* שלא לצורך כתוצאה מנקודה מרוחקת במדגם. אך למעט זאת, נתן תוצאה יחסית טובה. ניתן לראות כי *Affinity Propagation* נתן תוצאה פחות טובה ולא הצליח להבדיל בין המקבצים השונים.

4) שני מעגלים אחד בתוך השני



ניתן לראות כי עבור מדגם זה קיבלנו תוצאות שונות מאוד בין אלגוריתם אחד לשני.
GMM – האלגוריתם לא ידע להפריד בין שני המעגלים, ובמקום זאת "העביר קו" די שרירותי בין נקודות המדגם ובכך חילק את המדגם לשני *clusters* שווים.
Mean Shift – האלגוריתם התכנס ל-*cluster* אחד. זוהי תוצאה מעניינת מכיוון שבאתחול האלגוריתם כל נקודה מהמדגם הייתה מרכז בפני עצמה. במהלך הריצה כל המרכזים "זזו" לכיוון המרכז המשותף של המעגלים כך שבסוף הריצה כל המרכזים היו מספיק קרובים אחד לשני על מנת לאחדם למרכז אחד.
Affinity Propagation – האלגוריתם חילק כל מעגל ל-*clusters* בצורת קשתות קטנות באותו הגודל. ניתן לראות שלא קיים *cluster* שבו נמצאות נקודות משני המעגלים, כך שבמובן מסוים האלגוריתם כן הצליח להפריד בין המעגלים. לדעתנו זוהי התוצאה הטובה ביותר עבור מדגם זה.

5) שני "ירחים" משולבים אחד בשני



ניתן לראות כי *GMM* לא הצליח להבדיל בין שני הירחים ונתן חלוקה לא טובה. זוהי תוצאה צפויה עבור מדגם זה מאחר שהוא שונה מאוד מ-*Gaussian mixture*.
ניתן לראות כי *Mean Shift* הצליח להתכנס לשני *clusters* שזוהי תוצאה טובה. ניתן לראות כי האלגוריתם מיקם את המרכזים בנקודות הקרובות ביותר בין קצה ירח אחד לאמצע ירח שני.
ניתן לראות כי *Affinity Propagation* נתן מספר גדול של *clusters*, אך כל *cluster* שייך לירח אחד בלבד (פרט לשגיאה קטנה).