תרגיל בית מס' Z (תרגיל מסכם)

- יש להגיש תרגיל זה אונליין בקובץ ZIP מהפורמט assZ_number1_number2.ZIP כאשר number1 ו-number2 הם מספרי תעודות הזהות של הסטודנטים המגישים (דוג' assZ_012345678_098765432.ZIP). על הקובץ להכיל:
  - חבילה/חבילות של קוד python 3 כנדרש עם כל התלויות המאפשרות למתודות של החבילות לרוץ (יש לשים לב לדרישות הספציפיות משום שעל הקוד לרוץ בסקריפט בדיקה מוכן מראש).
  - **<u>בנוסף מסמך מלווה מוקלד</u>** (ולא כתוב ביד וסרוק) בפורמט PDF (ניתן להכין ב-Word מסמך DOC או DOCX ואז לשמור כ-PDF). המסמך יכיל תשובות לשאלות התאורטיות לפה הדרישות, כמו הסבר קצר וברור של הבחירות שנעשו בעת כתיבת הקוד, קבצים, מחלקות ומתודות חשובות, וכו'.
- שאלות התכנות מתייחסות ל-data מסוים. במידה ואין הפניה מהעבודה ישירות, data זה זמין באתר הקורס בצורת קובץ או הפניה (הורידו את ה-data מחדש. הקבצים עודכנו לפני פרסום התרגיל).
- על התשובות התכנותיות להיות גמישות כך שיעבדו על כל data בפורמט של ה-data שלכם.
- יש לרשום את שם ות"ז המגישים בראש כל אחד מהקבצים בפתרון התרגיל (קובץ ה-PDF וכל אחד מקבצי הקוד בצורת comment).
- יש לנסות לכתוב קוד יעיל ככל האפשר במסגרת המגבלות.
- יש לכתוב שמות משמעותיים למחלקות, פונקציות ומשתנים.
- מותר להוסיף קבצי py נוספים על הקבצים שסופקו. יש להגיש גם אותם ב-zip. על הקבצים הנוספים להיות באותה ספריה כמו קבצי ה-py שצורפו לשאלה.
- יש לכתוב קוד יעיל ככל שתוכלו האפשר במסגרת המגבלות. ריצות בדיקה יוגבלו בזמן.
- יש לכתוב שמות משמעותיים למחלקות, פונקציות ומשתנים.
- במידה ויש הערות תיעוד בקוד: **באנגלית בלבד** (במסמך המלווה מותר להשתמש בעברית) ועליהן להאיר נקודות חשובות או סבוכות בקוד.
- הניקוד מתחלק בין התאורטי למעשי. הקוד והתוצאות של ריצות נחשבות לחלק המעשי. הסברים במסמך המלווה למה שעשיתם בחלק המעשי נחשבים בחלק הניקוד שמוקצה לחלק התאורטי.
- מותר להיעזר באינטרנט כדי להבין איך עובד אלגוריתם כזה או אחר או איך משתמשים ב-python בצורה יצירתית שמקלה את התרגיל. אסור להוריד חבילות או להעתיק קוד שפותר את התרגיל בשבילכם.
- מותר להתייעץ ב-AI. אסור לעשות Copy/Paste מקוד שנתן לכם AI בתור פתרון מלא (או כמעט מלא) של השאלה.
- לא לתת ל-AI לכתוב בשבילכם את המסמך המלווה. במידה והמסמך המלווה יכיל חומות מלל AI שלא מסביר את הקוד שלכם ולא עומד בדרישות זה יחשב לחוסר יושר אקדמי.
- במידה ובלית ברירה השתמשתם בכמות קטנה של קוד מועתק ממקור חיצוני (מהאינטרנט, ולא מסטודנטים אחרים) **<u>יש לציין זאת במפורש</u>** בהערה בקוד כמו גם במסמך המלווה (יש להבהיר מהו המקור, מה גודל הקטע המועתק, והאם נעשו שינויים בקוד המועתק). סטודנטים שיציינו שאילות קוד כאלה ציונם בתרגיל עשוי להיפגע אך לא יחשבו ככאלה שנהגו בחוסר יושרה אקדמי.
- יש להגיש את הפתרון עד למועד שנקבע <u>במודל (כלומר אתר הקורס) בלבד</u>. איחור לא מאושר עשוי לגרום לפסילת התרגיל.
- בלי קשר לסכום הנקודות שצברתם. לא ניתן לקבל יותר מ-100 בתרגיל.

נושאי התרגיל: Clustering
תאריך אחרון להגשה: 6/3/2025

<u>תרגיל סיום:</u> אלגוריתמי Clustering על נתונים מסוגים שונים ומגדלים שונים.
במשימה זו עליכם לממש כלים ממספר סוגים

1. כלים לייצור של דאטה רב-ממדי המורכב ממספר אשכולות.
2. כלי ל-Clustering היררכי של דאטה קטן יחסית
3. כלי K-Means ל-Clustering של דאטה במרחב אוקלידי שמתאים לזכרון הראשי
4. כלי BFR ל-Clustering של דאטה אוקלידי שיושב בדיסק הקשיח ומורכב מורכב מאשכולות שמתפלגים נורמלית בכל אחד מהצירים
5. כלי CURE ל-Clustering של דאטה אוקלידי שיושב בדיסק הקשיח והתפלגות האשכולות שלו אינה ידועה

כמה דברים חשובים בקשר לפרויקט

- במהלך המימוש תצטרכו לבצע כל מיני החלטות. במסמך המלווה שיצורף להגשה שלכם תצטרכו להסביר את ההחלטות האלה ולמה הקוד שלכם מבצע את המבוקש.
- במשימה זו נקודות ב-dim ממדים ייוצגו (לפחות כלפי חוץ) כ-tuples בגודל dim
  לדוגמא (2.2,1.3,-4.0) היא נקודה במרחב אוקלידי בן 3 ממדים.
- כאשר אלגוריתם clustering שאינו שומר את התוצאות לקובץ יסיים לרוץ הוא ישמור את את הנקודות בזיכרון כרשימה של רשימות. לדוגמא:
  [[(3, 4, 7), (4, 5, 6), (3, 4, 6), (5, 6, 7)], [(2, 4, 2), (2, 3, 3), (0, 2, 3), (0, 1, 4), (1, 2, 4), (3, 4, 4)]]
- רשימות של נקודות ו-clusters ישמרו בקבצים מפורמט CSV שהם קבצי טקסט בהם השדות מופרדים בפסיקים. ראו הגדרה כאן. כל נקודה תתפוס שורה. במידה ונשמר גם מספר ה-cluster בקובץ הוא ישמר בתור הערך האחרון בכל שורה.
  כלומר בלי מספר cluster השורות יראו כך:

  3,4,5
  2,4,2
  …

  ועם מספר cluster השורות יראו כך:

  3,4,5,0
  2,4,2,1
  …

חלק 1: (30 נקודות) סביבת Clustering ואלגוריתמים לשימוש בזיכרון הראשי.

בחלק זה יש לממש:

1. פונקציה המייצרת נתונים ושומרת אותם בקובץ CSV.
2. פונקציות clustering שעובדות בזיכרון הראשי.
3. פונקציות המקבלות קובץ CSV עם נתונים אלגוריתם Clustering שעובד בזיכרון הראשי ושומרת את הנתונים אחרי פעולת ה-Clustering בקובץ CSV חדש.

1) כתבו את הפונקציה

```
def generate_data(dim, k, n, out_path, points_gen=None, extras={})
```

הפונקציה generate_data תיצור קובץ CSV ב-path שנמצא במשתנה out_path אשר יכיל n נקודות ב-dim ממדים אשר מחולקות ל-k clusters. המטרה היא לא פשוט להדביק לנקודות k תוויות שונות, אלא ליצר נתונים כך שהם מורכבים ממספר clusters שאלגוריתם clustering יכול לזהות. הפונקציה יכולה לקבל פונקציית עזר point_gen אשר תיצור את הנקודות (או חלק מהן, לבחירתכם). אתם יכולים להחליט אילו פרמטרים לשלוח ל-points_gen ומה היא מחזירה. במידה ואתם רוצים להגדיר לפונקציה ארגומנטים נוספים אתם יכולים להוסיף אותם למילון extras. הסבירו במסמך המלווה איך הקוד שלכם מייצר בהצלחה נתונים מחולקים ל-clusters. הסבירו גם מהם הפרמטרים הנוספים במידה ויש כאלה. אם השתמשתם ב-point_gen, הסבירו מה היא עושה ואיך היא עובדת.

כתבו את הפונקציה

```
def load_points(in_path, dim, n=-1, points=[])
```

הפונקציה load_points תקרא מקובץ CSV ב-path שנמצא במשתנה in_path n נקודות ב-dim ממדים. כל נקודה תיקרא משורה בקובץ. במידה ויש יותר מ-dim נתונים בשורה load_points תהפוך את dim הראשונים לנקודה ותתעלם מהיתר. במידה ויש יותר מ-n שורות בקובץ load_points תקרא את n השורות הראשונות. במידה ו-n קיבל את ערך ברירת המחדל 1-, load_points תקרא את כל השורות בקובץ. הפונקציה תכניס לרשימה points את הנקודות שנשלפו מ-in_path.

2) כתבו את הפונקציה

```
def h_clustering(dim, k, points, dist, clusts=[])
```

הפונקציה h_clustering תבצע clustering היררכי bottom up של נקודות ב-dim ממדים. הנקודות נמצאות במשתנה points נקודות ב-dim ממדים. k הוא מספר ה-clusters הרצוי. במידה ו-k=None צריך להחליט לבד מתי להפסיק. השתמשו בגישה שנלמדה בכיתה כדי לבחור מתי להפסיק במקרה זה. dist היא פונקציית המרחק, במידה ו-dist=None מניחים מרחב אוקלידי (יהיה עליכם לכתוב פונקציה המחשבת מרחק אוקלידי). clusts הוא משתנה פלט אליו יוכנסו האשכולות כרשימה של רשימות של נקודות. הסבירו במסמך המלווה שלכם את ההחלטות שקיבלתם ואיך הוא עובד.

3) כתבו את הפונקציה

```
def k_means(dim, k, n, points, clusts=[])
```

הפונקציה k_means תקרא תבצע k-means clustering של נקודות ב-dim ממדים. הנקודות נמצאות במשתנה points. k הוא מספר ה-clusters הרצוי. במידה ו-k=None צריך להחליט מה ה-k הנכון. השתמשו בגישה שנלמדה בכיתה כדי לבחור מתי למצוא את k במקרה זה (זה יצריך מספר קריאות עם ערכי k שונים). clusts הוא משתנה פלט אליו יוכנסו האשכולות כרשימה של רשימות של נקודות. הסבירו במסמך המלווה שלכם את ההחלטות שקיבלתם ואיך הוא עובד.

4) כתבו את הפונקציה

```
def save_points(clusts, out_path, out_path_tagged)
```

הפונקציה save_points תיצור 2 קובצי CSV מרשימה של clusters במשתנה clusts. הקובץ ב-path שנמצא במשתנה out_path אשר יכיל את כל הנקודות ב-clusts בסדר אקראי. הקובץ ב-out_path_tagged יכיל את הנקודות לפי הסדר כשבכל שורה שדה נוסף המכיל את מספר ה-cluster (בין 0 ל-k-1 כאשר k הוא מספר ה-clusters ב-clusts).

חלק 2: (30 נקודות) אלגוריתמים ל-clustering של כמות נקודות גדולה.

1) מצאו דרך ליצור כמות גדולה של נקודות (גדולה מכדי להחזיקה בזיכרון) ולשמור אותה לקובץ CSV.
אתם יכולים להעזר בפונקציה generate_data מחלק 1.
הסבירו איך הקוד (ובעזרת איזו פונקציה) שלכם מבצע את המשימה.

2) כתבו את הפונקציה

bfr_cluster(dim, k, n, block_size, in_path, out_path)

הפונקציה bfr_cluster תבצע BFR clustering. הנקודות נמצאות בקובץ CSV אשר נמצא ב-path שבמשתנה in_path אשר יכיל לפחות n נקודות בלפחות dim ממדים. כל שורה מייצגת נקודה אחת. במידה ובשורה יש יותר מ-dim שדות יש לקחת את dim השדות הראשונים בשורה בתור dim הממדים של הנקודה. k הוא מספר ה-clusters הרצוי. במידה ו-k=None צריך להחליט מה ה-k הנכון. השתמשו בגישה שנלמדה בכיתה כדי לבחור מתי למצוא את k במקרה זה (זה יצריך מספר קריאות עם ערכי k שונים). block_size הוא מספר הנקודות שטוענים בכל שלב של BFR מהקובץ לזיכרון הראשי. n הוא מספר הנקודות עליו עושים clustering. בוחרים את n השורות הראשונות ב-in_path. out_path הוא קובץ CSV אליו ישמרו הנקודות ביחד עם ה-cluster אליו הן שייכות. הסבירו במסמך המלווה שלכם את ההחלטות שקיבלתם ואיך הוא עובד.

3)

cure_cluster(dim, k, n, block_size, in_path, out_path)

הפונקציה cure_cluster תבצע CURE clustering. הנקודות נמצאות בקובץ CSV אשר נמצא ב-path שבמשתנה in_path אשר יכיל לפחות n נקודות בלפחות dim ממדים. כל שורה מייצגת נקודה אחת. במידה ובשורה יש יותר מ-dim שדות יש לקחת את dim השדות הראשונים בשורה בתור dim הממדים של הנקודה. k הוא מספר ה-clusters הרצוי. במידה ו-k=None צריך להחליט מה ה-k הנכון. השתמשו בגישה שנלמדה בכיתה כדי לבחור מתי למצוא את k במקרה זה (זה יצריך מספר קריאות עם ערכי k שונים). block_size הוא מספר הנקודות שטוענים בכל שלב של CURE מהקובץ לזיכרון הראשי (הוא גם חסם עליון למספר הנקודות שטוענים לזיכרון בשלב הדגימה של CURE). n הוא מספר הנקודות עליו עושים clustering. בוחרים את n השורות הראשונות ב-in_path.
out_path הוא קובץ CSV אליו ישמרו הנקודות ביחד עם ה-cluster אליו הן שייכות. הסבירו במסמך המלווה שלכם את ההחלטות שקיבלתם ואיך הוא עובד.

חלק 3: (40 נקודות) בדיקות והשוואות

בחלק זה עליכם להריץ את הקוד שלכם ולרשום את התוצאות שלכם בטבלאות.

ראשית יש ליצור קבצים

- צרו קובץ דאטה קטן עם 2 ממדים המחולק ל-5 אשכולות לפחות (מומלץ כ-1000 שורות סה"כ).
- צרו קובץ דאטה מספיק קטן בשביל עבודה בזכרון הראשי עם dim1 ממדים המחולק ל-k1 אשכולות. על הפרמטרים לקיים $dim1 > 3, k > 4$ וגם $dim1 + k1 > 10$.
- צרו עוד לפחות 2 קבצים אחרים כאלה בגדלים שונים עם dim, k שונים.
- מותר לכם ואף מומלץ להוריד בנוסף דאטה קיים כדי לנסות עליו את האלגוריתמים (יתכן שיהיה צורך בקוד שיעבוד על הפורמט).
- אל תגישו את קבצי הדאטה (מלבד הקטן). אתם יכולים להגיש לינקים לדרייב.

הסבירו איך יצרתם את הקבצים ודאגתם לזה שהם יהיו מחולקים לאשכולות. הסבירו גם איך דאגתם שהחלוקה לא תהיה טריוויאלית מדי. על ההגשה להכיל סקריפטים של python שיוצרים את הקבצים, והוראות ליצירתם.

כעת יש לבצע הרצות של clustering היררכי ואלגוריתם k-means:

אתם צריכים לכתוב מהו המדד שלכם (או המדדים שלכם) לאיכות של החלוקה לאשכולות. מה גורם לכם להעריך את ה-clustering כמוצלח יותר או פחות ומה גורם לאלגוריתמי ה-clustering לבחור k במידה ולא סיפקתם להם את k מראש. את חלק מהדברים האלה כבר היה צריך לכתוב בחלק 1. לא חובה לחזור על הכל בפירוט שוב, אבל חשוב שזה יהיה כתוב איפשהו.

על הקובץ הראשון הריצו את אלגוריתמי h_clustering ו-k-means.
הראו כיצד המדד המתאים משתנה כאשר k משתנה:
הנה דוגמא לטבלה כזו (יש לשקף מה בדיוק מודדים עבור ערכי k שונים):

File name	Algorithm	k=2	k=3	K=4	K=5	K=6	K=7	K=8
f1.csv	Hierarchical							
f1.csv	k-means							

הוסיפו טבלה כזו גם עבור לפחות קובץ קלט אחד מהגדולים יותר. ציינו עבור כל אחד מהקבצים שלכם האם האלגוריתמים שלכם בוחרים ב-k הנכון.

בדיקת דיוק:
מאחר ואת הקבצים אתם יצרתם ה-clustering ה-"נכון" אמור להיות ידוע לכם, ושמור בקובץ נפרד. כעת אתם יכולים לבדוק באיזו מידה האלגוריתמים שלכם מסווגים את הנקודות בנתונים שלכם ל-clusters ה-"נכונים". שימו לב שאם יצרתם clusters קרובים ולא מאוד קלים לפענוח זה טבעי מאוד שהאלגוריתמים לא תמיד יגיעו לדיוק של 100%, ואין שום בעיה עם זה. הטבלה יכולה להיראות כך:

File name	Algorithm	k0	k1	k2	k3	k4	k5
f2.csv	Hierarchical	99.11%	97.22%	97.79%	...	...	...
f2.csv	k-means	99.21%	97.12%	98.98%	...	...	...

File name	Algorithm	k0	k1	k2	k3	k4	k5	k6	k7	k8
f3.csv	Hierarchical	75.22%	...	...	...	...	...	...	...	...
f3.csv	k-means	85.32%	...	...	...	...	...	...	...	...

משמעות המספרים בטבלאות שלמעלה היא האחוז מבין האיברים ששויכו ל-cluster אז אמורים להיות חלק ממנו. שימו לב שלא בהכרח אלגוריתם ה-clustering שלכם יבחר את אותו סדר של clusters כמו בקובץ המקורי (כלומר המקורי יכול להיות מתויג כעת כ-$k3$). חוסר תשומת לב כזה עלול לגרום לכם בטעות לחשוב שהאלגוריתם שלכם פספס כמעט לחלוטין clusters שלמים כאשר הוא רק נתן להם שם אחר. אז שימו לב כשאתם משווים. $k1$

כאשר אתם מייצרים טבלאות דאגו שיהיה לכם קובץ אחד עם clustering מתויג נכון כפי שיצרתם את הנתונים (כלומר ללא אלגוריתם clustering), וקובץ אחד עם התשובות שנתן אלגוריתם ה-clustering שאתם רוצים להעריך את הדיוק שלו. עכשיו הריצו קוד שיעבור על 2 הקבצים וישווה בין המיקומים של הנקודות לפי שתי החלוקות ל-clusters.

<u>הערה:</u> מכיוון שאינכם נדרשים להגיש את הקוד שמבצע את ההשוואה בין התוצאות שלכם לבין ה-clustering ה-"נכון", אלא רק לדווח על התוצאות, אתם בהחלט יכולים להשתמש ב-AI או במקורות חיצוניים ללא מגבלה ביצירת הקוד הזה.

<u>בדיקות לגבי חלק 2 (קבצים גדולים):</u>

<u>ראשית יש ליצור קבצים</u>

- צרו 2 קובצי דאטה בגודל של לפחות 10 ג'יגה בייט כל אחד (בעצם 4 קבצים אם שומרים גם את המתויגים לפי cluster). שמרו אותם בדיסק הקשיח שלכם. עליהם תעבדו.
- קובץ מספר 1: מותאם לעבודה עם BFR.
- קובץ מספר 2: מותאם לעבודה עם Cure.
- שני הקבצים יהיו עם dim ממדים ומחולקים ל-k אשכולות. על הפרמטרים לקיים $dim > 5, k > 5$.
- אל תגישו את הקבצים אבל הגישו סקריפטים של python שיוצרים את הקבצים, והוראות להפעלת הסקריפטים.
- מאוד חשוב ליצור את הקבצים שלכם ולא להעתיק 1 ל-1 את דרך יצירת הקבצים של סטודנטים אחרים.
  - הנתונים של BFR הם תמיד די דומים אבל עם סטיות תקן שונות ומרכזים שונים. משחק עם dim ו-k כמו גם עם דרך בחירת סטיות התקן והמרכזים תהיה ההבדל בין הקבצים.
  - מותר לכם להתיעץ ב-AI כדי לחשוב על רעיונות בשביל יצירת clusters לאלגוריתם Cure. זכרו תמיד לוודא שההצעות הגיוניות.

<u>בדיקת דיוק:</u>

צרו טבלאות כמו שעשיתם עבור בדיקות הדיוק של חלק 1 והעריכו איך עובדים שני האלגוריתמים על 2 הקבצים (אם יצרתם את הקבצים נכון אז BFR אמור פחות להצליח על הדאטה שנוצר עבור Cure).