מסמך מלווה לפרויקט סיום בקורס כריית נתונים

---------------------------------------------------------------------------------

**חלק ראשון**

סעיף 1 –

ייצור הנתונים שיותאמו לclusters יתבצע באופן הבא:

1) בחירה אקראית של מספר הנקודות שבכל קבוצה כך שסכום הקבוצות יהיה שווה לn

2) K פעמים יתבצעו השלבים הבאים:

1. maxMumberInThisGroup- הגרלת מספר מ 5000 עד 10000 (שהוא ייצג את המספר המקסימלי שיכול להיות בקבוצה הזו) עבור כל מספר שהוגרל בסעיף קודם שהוגרל- יוגרלו dim מספרים מ maxMumberInThisGroupחלקי 2 ועד maxMumberInThisGroup וזה בעצם נקודה אחת מסך הנקודות
   1. סעיף זה יתבצע בפונקציית עזר **שמקבלת** את הK הרלוונטי ויוצאת נקודות ואת
   2. חזרה על סעיף 3 C פעמים

הסבר לסעיפים 2-3: סעיפים אלו התבצעו באופן הזה על מנת לאפשר לנקודות פוטנציאליות להיות באותו אשכול

סעיף 2 –

האלגוריתם שיתבצע הוא אשכול היררכי, שבו כל נקודה מתחילה כאשכול נפרד. במהלך החישוב, נבצע מספר לולאות: בלולאה הפנימית, נחשב את המרחקים בין כל נקודה לכל נקודה אחרת שלא באותו אשכול בהתחלה זה יהיה כולם עם כולם, ולאחר מכן, בשלבים יותר מתקדמים, נחשב ממוצע עבור כל אשכול כדי לייצג אותו כנקודה אחת.

לאחר מכן, נחשב את המרחק האוקלידי בין כל נקודת אשכול לנקודת אשכול אחרת. בסיום כל לולאה, נבחר את המרחק המינימלי בין האשכולות ונוודא שכל הנקודות יתאגדו לפי מיזוג על פי המרחק הקצר ביותר

לגבי ההחלטה על מספר האשכולות אם לא הוזן k מראש, נמשיך במיזוג האשכולות עד שכל הנקודות יהיו באותו אשכול.

הפונקציה h\_clustering תבצע clustering היררכי על נקודות ב-dim ממדים, כאשר הקלט הוא רשימת points שמכילה את הנקודות.

אם k לא הוזן, הפונקציה תמשיך עד שכל הנקודות יתאגדו לאשכול אחד.

dist הוא פונקציית המרחק (אם לא הוזן, נניח שמדובר במרחק אוקלידי), ו-clusts יהיה משתנה הפלט שבו יאוחסנו האשכולות.

סעיף 3 –

במסגרת המימוש של אלגוריתםK-Means , מקבלת את המשתנים הבאים:

dim מספר הממדים של כל נקודה.

מספר הקלאסטרים (או None אם לא מוגדר מראש). K

n מספר הנקודות.

points רשימת הנקודות.

Clusts רשימת הקלאסטרים שיכיל את הרשימה הסופית ובה כל תא מייצג אשכול

הפונקציה פועלת בשני מקרים שונים, בהתאם לערך של k.

כאשר k מוגדר:

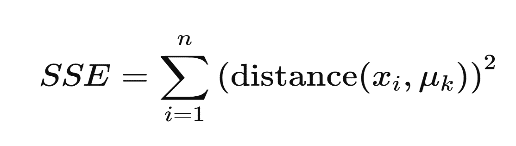
במקרה זה, אנחנו בוחרים באופן אקראי k נקודות מתוך מאגר הנקודות שלנו כמרכזים התחלתיים.

לאחר מכן, עבור כל נקודה, אנו מחשבים את המרחקים שלה לכל אחד מהמרכזים באמצעות מרחק אוקלידי.

הנקודות מסווגות לפי הקרבה למרכזים, ואז כל מרכז משתנה להיות ממוצע הנקודות שהוקצו לו. התהליך הזה חוזר על עצמו עד שהמרכזים מתייצבים.

כאשר k הוא None:

במקרה זה, אנחנו מריצים את אלגוריתם K-Means מספר פעמים עבור ערכים שונים של k למשל, מ-1 עד k\_max

עבור כל ערך של k, אנו בודקים איזה K יש לו שגיאה יותר נמוכה, עבור כל קבוצה, נחשב את המרחקים בין כל נקודה במרכז שלה, ואז נרצה למצוא את הk שמניב את השגיאה הקטנה ביותר – SSE

בסיום, אנו בוחרים את הערך של k שמניב את התוצאה האופטימלית ומבצעים את אלגוריתם K-Means בפעם האחרונה עם הערך האופטימלי של k.  
בחרנו שבדיקת K אופטימלי תהיה בלולאה מk=2 עד עשירית מהK על מנת לצמצם פיזור יתר (למנוע מצבים שבה כל נקודה באשכול נפרד)

עבור סעיף 4 לא נדרשנו להסביר

---------------------------------------------------------------------------------

**חלק שני**

סעיף 1 –

סעיף 2 –

כשבנינו את האלגוריתם, המטרה שלנו הייתה לבצע אשכול תעזור לי בצורה יעילה תוך שמירה על ניהול זיכרון חכם והתאמה טובה של מרחקים בין נקודות. החלטנו להשתמש במרחק מהלנוביס כי הוא מאפשר לנו להתחשב בשונות המשתנים ולמנוע השפעה של הבדלים בקנה המידה של הנתונים.

לגבי אתחול קלאסטרים ראשוניים:

בהתחלה, בחרנו נקודות אקראיות כמרכזי קלאסטרים. החלטנו ללכת על גישה אקראית כי היא נותנת חלוקה ראשונית מגוונת יותר של הנתונים. וגם אין בעצם מצב שבו קלאסטר יהיה ריק מהסיבה שנקודה אחת בוודאות תהיה חלק מקלאסטר מסוים שהוא הקלאסטר שהיא נקודת המרכז שלו.

לגבי ייצוג קלאסטר בצורה חכמה:

כדי לייצג קלאסטר, השתמשנו בווקטור שמכיל מספר נקודות, סכום כל הנקודות, וסכום הריבועים שלהן. זה מאפשר לנו לחשב בקלות את המרכז והסטייה התקנית בהמשך.

לגבי איחוד קלאסטרים בצורה יעילה:

כשהבנו ששני קלאסטרים קרובים מספיק, השתמשנו בפונקציה לאיחוד קלאסטרים. הדרך הזו חוסכת לנו חישובים מיותרים ומשתמשת בנתונים שכבר חישבנו.

לגבי חישוב מרכז וסטיית תקן במהירות:

במקום לחשב כל פעם מחדש, אנחנו משתמשים בפונקציה לחישוב מרכז וסטיית תקן כדי לעדכן את המרכז והסטייה התקנית על בסיס הנתונים שנאספו עד כה.

לגבי מיזוג קלאסטרים לפי מרחק:

לפני שאנחנו מאחדים קלאסטרים, אנחנו בודקים את המרחק ביניהם. אם המרחק קטן מהסף שהגדרנו, אנחנו מחברים אותם כדי למנוע ריבוי קלאסטרים קטנים מדי.

לגבי הפונקציה הראשית - ניהול כל התהליך:

בפונקציה המרכזית הרכבנו את כל השלבים יחד:

1. טעינת נתונים בבלוקים - כדי לא להעמיס על הזיכרון, אנחנו טוענים את הנתונים במקטעים.
2. חישוב קלאסטרים ראשוניים - אם אין מספיק קלאסטרים מוכנים מראש, אנחנו יוצרים אותם מהנקודות הראשונות.
3. שיוך נקודות לקלאסטרים - כל נקודה מקבלת מרחק לכל קלאסטר ומצטרפת לזה שהכי קרוב אליה.
4. מיזוג קלאסטרים קרובים - אם קלאסטרים קרובים מדי, אנחנו מחברים אותם לפי תנאי הסף שקבענו.
5. שמירת התוצאה - בסוף התהליך, אנחנו כותבים את התוצאות לקובץ, כולל מזהה קלאסטר לכל נקודה.

---------------------------------------------------------------------------------

חלק 3 –

יצירת הקבצים נעשתה בעזרת פונקציית generate\_data מהחלק הראשון שמבצע תהליך של יצירת נתונים הכולל את השלבים הבאים:

1. חלוקת הנתונים באופן אקראי ל-K קבוצות, כך שסכום הקבוצות יהיה שווה ל-n.
2. עבור כל אשכול (סה"כ K פעמים), מבוצעים השלבים הבאים:

* בחירה באופן אקראי מספר מקסימלי של נקודות באותו אשכול (maxMumberInThisGroup) בטווח שבין 5000 ל-10000.
* עבור כל מספר שהוגרל, ייבחרו dim (ממד) מספרים בטווח שבין חצי מהמספר המקסימלי לבין המספר המקסימלי, כך שיתקבלו נקודות אקראיות שמתאימות לאותו האשכול.

המטרה היא ליצור את הנתונים באופן שמבטיח חלוקה אקראית אך עדיין שומרת על האשכולות, כך שהנקודות בתוכם לא יהיו מופרדות באופן טריוויאלי, אלא יתחלקו בצורה טבעית בתוך החלוקה שנבחרה.