# עבודה מספר 2 – מסמך אפיון

# מימוש אלגוריתם K-MEANS באמצעות SPARK

שחר עודד – 208388918

ניר רהב – 316275437

במסמך זה נעבור על אופן בניית הקוד שלנו שלב אחר שלב.

## תתי פונקציות במימוש:

תתי הפונקציות שהגדרנו ביצעו משימות נלוות לפונקציה הראשית שלנו, שלא רצינו להכניס למימוש הראשי מטעמי יעילות, מודולריות ונראות הקוד.

### def Normalize\_dataset(dataset):

* הפונקציה מופעלת ע"י הפונקציה הראשית ומקבלת dataset בתצורה מוכנה בspark כקובץ RDD.
* הפונקציה מקימה אובייקט MinMaxScaler
* הפונקציה מפרידה את הdata לקובץ פיצ'רים ולקובץ class, ומשתמשת בtransform על מנת לנרמל את קובץ הפיצ'רים.
* לאחר מכן הפונקציה תאחד את שני הרכיבים, ותמיר את הקובץ חזרה לצורת RDD
* לבסוף יוחזר קובץ RDD מפוצל שוב לקובץ פיצ'רים ולקובץ מטרה, רק הפעם באופן מבוזר (RDD)

### euclidean\_distance(a, b):

* הפונקציה מקבלת שני ווקטורים a,b ומחזירה את המרחק האוקלידי בינהם כחישוב פשוט, בתור float.

### closest\_centroid(row, centroids):

* תוך שימוש בפונקציה הקודמת הפונקציה מחזירה את האינדקס של המרכז הקרוב ביותר לשורה מסוימת בדאטה
* הקלט של הפונקציה הוא שורה בדאטה ורשימת המרכזים (ממומש כרשימה מטעמי נוחות)
* הפונקציה מייצרת רשימה של מרחקים של שורת הדאטה מכל אחד מהמרכזים ובאמצעות אינדוקס לרשימה המקורית מחזירה את האינדקס של המרכז הקרוב ביותר לשורה המבוקשת כמספר int.

## הפונקציה הראשית:

### kmeans\_spark(dataset\_path, k, convergence\_threshold=0.0001, num\_iterations=30, num\_experiments=10)

**הפונקציה הראשית במימוש, נפרק את עבודתה לפי שלבים:**

* הפונקציה מייבאת קובץ CSV באמצעות dataset\_path הניתן כקלט
* לאחר יצירת הקובץ כקובץ RDD הפונקציה תשלח אותו לפונקציה Normalize\_dataset(dataset) שהוגדרה לעיל, אשר תחזיר שני קבצי RDD: X,y.
* הפונקציה תיכנס ללולאה המרכזית שלה, הרצה כמספר ה num\_experimentsעל מנת לייצר מספר ערכים בסיום, ולבחון את הממוצע בינהם (ממוצע ציונים במטריקות שיוגדרו בהמשך).
* הפונקציה תבחר מתוך X מספר דגימות רנדומליות (k) ותייצר מהם מרכזים הנשמרים בתור רשימה של tuples מהצורה:
* במהלך הלולאות נגדיר תנאי עצירה הבודק את השינוי במיקום המרכזים לאורך האיטרציות בלולאה הפנימית (num\_iterations), לכן, נאתחל ערך ראשוני לשינוי בתור .
* ניכנס ללולאה הפנימית הרצה num\_iterations פעמים

**פעולת הmapper:**

* עבור פעולת הmapper הפונקציה תרצה לחשב מהו המרכז הקרוב ביותר לכל שורה בdata באמצעות הפונקציה closest\_centroid. הפונקציה מחזירה אובייקטים מהצורה הבאה:

*\*\* חידוד, הפיצ'רים נכנסו כרשימה לתוך הtuple.*

* על מנת לעזור בפעולת הreduce המתקרבת, האובייקטים לאחר מכן סודרו לצורה:

**פעולת הreducer:**

* בשלב הראשון תייצר הפעולה פלט מהצורה הבאה לכל אינדקס של מרכז:
* לאחר מכן יבוצע reduce by key על מנת לעדכן את הספירה כל כמות השורות תחת המרכז
* לאחר מכן באמצעות מתודה mapValues נחשב על כל אחד מהמרכזים האלו את הממוצע החדש, הוא הופך להיות המרכז החדש במקומו של המרכז הישן (אינדקס), באותו הפורמט שהיה בהתחלה:
* באמצעות סכימה של ההפרשים בקואורדינטות, נוצר ערך נומרי עבור המרחק הכולל של כל המרכזים מהמרכז שהוגדר באיטרציה הקודמת. אם המרחק קטן מהסף המוגדר בפונקציה – נסיים את הלולאה הפנימית ונעבור לשלב הבא, אחרת נחזור כמספר האיטרציות שהוגדר.

**סיום הניסוי (אחד מתוך num\_experiments):**

* מתוך הקלאסטרים ששויכו לכל שורה כאובייקט
* נשלוף את האינדקס של הקלאסטר ששויך לכל שורה, ונצרף אותו כפיצ'ר עבור הערכים השמורים של X.
* נפעיל calinski\_harabasz\_score על הדאטה ונשמור את התוצאה ברשימה שהוגדרה מראש
* נפעיל adjusted\_rand\_score על הדאטה ונשמור את התוצאה ברשימה שהוגדרה מראש

**סיום ריצת הפונקציה:**

* בסיום הריצה על num\_experiments ולאחר שכל תוצאות הניסויים נאספו ברשימות, נחשב את הממוצע וסטיית התקן של כך אחד מהמבחנים עבור הdata הנתון ועבור המספר k שנבחר.

## תוצאות:

התוצאה הסופית של הפונקציה נשמרת בטבלה לכל dataset ולכל מספר מוגדר של מרכזים. להלן התוצאות שנאספו:

A picture containing text, number, screenshot, font

Description automatically generated

\*\* יש לציין כי ריצת הפונקציה נכשלה מספר פעמים על זמן ריצה באמצעות Collab ולכן הטבלה הורכבה יחדיו במספר ריצות נפרדות